**Assigned: 21 Feb 07**
**Topic: What is a computer? (Part II: Philosophy)**

**Required:**

Searle, John R. (1990), "Is the Brain a Digital Computer?", Proceedings and Addresses of the American Philosophical Association 64(3) (November): 21–37.

**Strongly recommended:**

Shagrir, Oron (2006), "Why We View the Brain as a Computer", Synthese 153(3) (December): 393–416.

**Recommended:**

Piccinini, Gualtiero  (2003), "The Mind as Neural Software: Functionalism, Computationalism, and Computational Functionalism", paper read at the APA Pacific Division (March 2004).

Rapaport, William J. (in press),  "Searle on Brains as Computers", American Philosophical Association Newsletter on Computers and Philosophy.

# IS THE BRAIN A DIGITAL COMPUTER?

John R. Searle
University of California/Berkeley

*Presidential Address delivered before the Sixty-fourth Annual Pacific Division Meeting of the American Philosophical Association in Los Angeles, California, March 30, 1990.*

## I. Introduction, Strong AI, Weak AI and Cognitivism.

There are different ways to present a Presidential Address to the APA; the one I have chosen is simply to report on work that I am doing right now, on work in progress. I am going to present some of my further explorations into the computational model of the mind. [1]

The basic idea of the computer model of the mind is that the mind is the program and the brain the hardware of a computational system. A slogan one often sees is "the mind is to the brain as the program is to the hardware." [2]

Let us begin our investigation of this claim by distinguishing three questions:

1. Is the brain a digital computer?
2. Is the mind a computer program?
3. Can the operations of the brain be simulated on a digital computer?

I will be addressing 1 and not 2 or 3. I think 2 can be decisively answered in the negative. Since programs are defined purely formally or syntactically and since minds have an intrinsic mental content, it follows immediately that the program by itself cannot constitute the mind. The formal syntax of the program does not by itself guarantee the presence of mental contents. I showed this a decade ago in the Chinese Room Argument (Searle, 1980). A computer, me for example, could run the steps of the program for some mental capacity, such as understanding Chinese, without understanding a word of Chinese. The argument rests on the simple logical truth that syntax is not the same as, nor is it by itself sufficient for, semantics. So the answer to the second question is obviously "No".

The answer to 3. seems to me equally obviously "Yes", at least on a natural interpretation. That is, naturally interpreted, the question means: Is there some description of the brain such that under that description you could do a computational simulation of the operations of the brain. But since according to Church's thesis, anything that can be given a precise enough characterization as a set of steps can be simulated on a digital computer, it follows trivially that the question has an affirmative answer. The operations of the brain can be simulated on a digital computer in the same sense in which weather systems, the behavior of the New York stock market or the pattern of airline flights over Latin America can. So our question is not, "Is the mind a program?" The answer to that is, "No". Nor is it, "Can the brain be simulated?" The answer to that is, "Yes". The question is, "Is the brain a digital computer?" And for purposes of this discussion I am taking that question as equivalent to: "Are brain processes computational?"

One might think that this question would lose much of its interest if question 2 receives a negative answer. That is, one might suppose that unless the mind is a program, there is no interest to the question whether the brain is a computer. But that is not really the case. Even for those who agree that programs by themselves are not constitutive of mental phenomena, there is still an important question: Granted that there is more to the mind than the syntactical operations of the digital computer; nonetheless, it might be the case that mental states are *at least* computational states and mental processes are computational processes operating over the formal structure of these mental states. This, in fact, seems to me the position taken by a fairly large number of people.

I am not saying that the view is fully clear, but the idea is something like this: At some level of description brain processes are syntactical; there are so to speak, "sentences in the head". These need not be sentences in English or Chinese, but perhaps in the "Language of Thought" (Fodor, 1975). Now, like any sentences, they have a syntactical structure and a semantics or meaning, and the problem of syntax can be separated from the problem of semantics. The problem of semantics is: How do these sentences in the head get their meanings? But that question can be discussed independently of the question: How does the brain work in processing these sentences? A typical answer to that latter question is: The brain works as a digital computer performing computational operations over the syntactical structure of sentences in the head.

Just to keep the terminology straight, I call the view that all there is to having a mind is having a program, Strong AI, the view that brain processes (and mental processes) can be simulated computationally, Weak AI, and the view that the brain is a digital computer, Cognitivism.

This paper is about Cognitivism, and I had better say at the beginning what motivates it. If you read books about the brain (say Shepherd (1983) or Kuffler and Nicholls (1976)) you get a certain picture of what is going on in the brain. If you then turn to books about computation (say Boolos and Jeffrey, 1989) you get a picture of the logical structure of the theory of computation. If you then turn to books about cognitive science, (say Pylyshyn, 1985) they tell you that what the brain books describe is really the same as what the computability books were describing. Philosophically speaking, this does not smell right to me and I have learned, at least at the beginning of an investigation, to follow my sense of smell.

## II. The Primal Story

I want to begin the discussion by trying to state as strongly as I can why Cognitivism has seemed intuitively appealing. There is a story about the relation of human intelligence to computation that goes back at least to Turing's classic paper (1950), and I believe it is the foundation of the Cognitivist view. I will call it the Primal Story:

> We begin with two results in mathematical logic, the Church-Turing thesis (sometimes called Church's thesis) and Turing's theorem. For our purposes, the Church-Turing thesis states that for any algorithm there is some Turing machine that can implement

> that algorithm. Turing's theorem says that there is a Universal Turing Machine which can simulate any Turing Machine. Now if we put these two together we have the result that a Universal Turing Machine can implement any algorithm whatever.

But now, what made this result so exciting? What made it send shivers up and down the spines of a whole generation of young workers in artificial intelligence is the following thought: Suppose the brain is a Universal Turing Machine.

Well, are there any good reasons for supposing the brain might be a Universal Turing Machine? Let us continue with the Primal Story.

> It is clear that at least some human mental abilities are algorithmic. For example, I can consciously do long division by going through the steps of an algorithm for solving long division problems. It is furthermore a consequence of the Church-Turing thesis and Turing's theorem that anything a human can do algorithmically can be done on a Universal Turing Machine. I can implement, for example, the very same algorithm that I use for long division on a digital computer. In such a case, as described by Turing (1950), both I, the human computer, and the mechanical computer are implementing the same algorithm; I am doing it consciously, the mechanical computer nonconsciously. Now it seems reasonable to suppose there might also be a whole lot of mental processes going on in my brain nonconsciously which are also computational. And if so, we could find out how the brain works by simulating these very processes on a digital computer. Just as we got a computer simulation of the processes for doing long division, so we could get a computer simulation of the processes for understanding language, visual perception, categorization, etc.

"But what about semantics? After all, programs are purely syntactical." Here another set of logico-mathematical results comes into play in the Primal Story.

> The development of proof theory showed that within certain well known limits the semantic relations between propositions can be entirely mirrored by the syntactic relations between the sentences that express those propositions. Now suppose that mental contents in the head are expressed syntactically in the head, then all we would need to account for mental processes would be computational processes between the syntactical elements in the head. If we get the proof theory right the semantics will take care of itself; and that is what computers do: they implement the proof theory.

We thus have a well defined research program. We try to discover the programs being implemented in the brain by programming computers to implement the same programs. We do this in turn by getting the mechanical computer to match the performance of the human computer (i.e. to pass the Turing Test) and then getting the psychologists to look for evidence that the internal processes are the same in the two types of computer.

Now in what follows I would like the reader to keep this Primal Story in mind--notice especially Turing's contrast between the conscious implementation of the program by the human computer and the nonconscious implementation of programs, whether by the brain or by the mechanical computer; notice furthermore the idea that we might just *discover* programs running in nature, the very same programs that we put into our mechanical computers.

If one looks at the books and articles supporting Cognitivism one finds certain common assumptions, often unstated, but nonetheless pervasive. *First*, it is often assumed that the only alternative to the view that the brain is a digital computer is some form of dualism. The idea is that unless you believe in the existence of immortal Cartesian souls, you must believe that the brain is a computer. Indeed, it often seems to be assumed that the question whether the brain is a physical mechanism determining our mental states and whether the brain is a digital computer are the same question. Rhetorically speaking, the idea is to bully the reader into thinking that unless he accepts the idea that the brain is some kind of computer, he is committed to some weird antiscientific views. Recently the field has opened up a bit to allow that the brain might not be an old fashioned von Neumann style digital computer, but rather a more sophisticated kind of parallel processing computational equipment. Still, to deny that the brain is computational is to risk losing your membership in the scientific community.

*Second*, it is also assumed that the question whether brain processes are computational is just a plain empirical question. It is to be settled by factual investigation in the same way that such questions as whether the heart is a pump or whether green leaves do photosynthesis were settled as matters of fact. There is no room for logic chopping or conceptual analysis, since we are talking about matters of hard scientific fact. Indeed I think many people who work in this field would doubt that the title of this paper poses an appropriate philosophic question at all. "Is the brain really a digital computer?" is no more a philosophical question than "Is the neurotransmitter at neuro-muscular junctions really acetylcholene?"

Even people who are unsympathetic to Cognitivism, such as Penrose and Dreyfus, seem to treat it as a straightforward factual issue. They do not seem to be worried about the question what sort of claim it might be that they are doubting. But I am puzzled by the question: What sort of fact about the brain could constitute its being a computer?

*Third*, another stylistic feature of this literature is the haste and sometimes even carelessness with which the foundational questions are glossed over. What exactly are the anatomical and physiological features of brains that are being discussed? What exactly is a digital computer? And how are the answers to these two questions supposed to connect? The usual procedure in these books and articles is to make a few remarks about 0's and 1's, give a popular summary of the Church-Turing thesis, and then get on with the more exciting things such as computer achievements and failures. To my surprise in reading this literature I have found that there seems to be a peculiar philosophical hiatus. On the one hand, we have a very elegant set of mathematical results ranging from Turing's theorem to Church's thesis to recursive function theory. On the other hand, we have an impressive set of electronic devices which we use every day. Since we have such advanced mathematics and such good electronics, we assume that somehow somebody must have done the basic philosophical work of connecting the mathematics to the electronics. But as far as I can tell that is not the case. On the contrary, we are in a peculiar situation where there is little theoretical agreement among the practitioners on such absolutely fundamental questions as, What exactly is a digital computer? What exactly is a symbol? What exactly is a computational process? Under what physical conditions exactly are two systems implementing the same program?

## III. The Definition of Computation

Since there is no universal agreement on the fundamental questions, I believe it is best to go back to the sources, back to the original definitions given by Alan Turing.

According to Turing, a Turing machine can carry out certain elementary operations: It can rewrite a 0 on its tape as a 1, it can rewrite a 1 on its tape as a 0, it can shift the tape 1 square to the left, or it can shift the tape 1 square to the right. It is controlled by a program of instruction and each instruction specifies a condition and an action to be carried out if the condition is satisfied.

That is the standard definition of computation, but, taken literally, it is at least a bit misleading. If you open up your home computer you are most unlikely to find any 0's and 1's or even a tape. But this does not really matter for the definition. To find out if an object is really a digital computer, it turns out that we do not actually have to look for 0's and 1's, etc.; rather we just have to look for something that we could *treat as* or *count as* or *could be used to* function as 0's and 1's. Furthermore, to make the matter more puzzling, it turns out that this machine could be made out of just about anything. As Johnson-Laird says, "It could be made out of cogs and levers like an old fashioned mechanical calculator; it could be made out of a hydraulic system through which water flows; it could be made out of transistors etched into a silicon chip through which an electrical current flows. It could even be carried out by the brain. Each of these machines uses a different medium to represent binary symbols--the positions of cogs, the presence or absence of water, the level of the voltage and perhaps nerve impulses" (Johnson Laird, 1988, p. 39).

Similar remarks are made by most of the people who write on this topic. For example, Ned Block (Block, 1990), shows how we can have electrical gates where the 1's and 0's are assigned to voltage levels of 4 volts and 7 volts respectively. So we might think that we should go and look for voltage levels. But Block tells us that 1 is only "conventionally" assigned to a certain voltage level. The situation grows more puzzling when he informs us further that we did not need to use electricity at all but we could have used an elaborate system of cats and mice and cheese and make our gates in such a way that the cat will strain at the leash and pull open a gate which we can also treat as if it were an 0 or 1. The point, as Block is anxious to insist, is "the irrelevance of hardware realization to computational description. These gates work in different ways but they are nonetheless computationally equivalent" (p. 260). In the same vein, Pylyshyn says that a computational sequence could be realized by "a group of pigeons trained to peck as a Turing machine!" (Pylyshn, 1985, p. 57)

But now if we are trying to take seriously the idea that the brain is a digital computer, we get the uncomfortable result that we could make a system that does just what the brain does out of pretty much anything. Computationally speaking, on this view, you can make a "brain" that functions just like yours and mine out of cats and mice and cheese or levers or water pipes or pigeons or anything else provided the two systems are, in Block's sense, "computationally equivalent". You would just need an awful lot of cats, or pigeons or waterpipes, or whatever it might be. The proponents of Cognitivism report this result with sheer and unconcealed delight. But I think they ought to be worried about it, and I am going to try to show that it is just the tip of a whole iceberg of problems.

## IV. First Difficulty: Syntax is not Intrinsic to Physics

Why are the defenders of computationalism not worried by the implications of multiple realizability? The answer is that they think it is typical of functional accounts that the same function admits of multiple realizations. In this respect, computers are just like carburetors and thermostats. Just as carburetors can be made of brass or steel, so computers can be made of an indefinite range of hardware materials.

But there is a difference: The classes of carburetors and thermostats are defined in terms of the production of certain *physical* effects. That is why, for example, nobody says you can make carburetors out of pigeons. But the class of computers is defined syntactically in terms of the *assignment* of 0's and 1's. The multiple realizability is a consequence not of the fact that the same physical effect can be achieved in different physical substances, but that the relevant properties are purely syntactical. The physics is irrelevant except insofar as it admits of the assignments of 0's and 1's and of state transitions between them.

But this has two consequences which might be disastrous:

1. The same principle that implies 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.

2. Worse yet, syntax is not intrinsic to physics. The ascription of syntactical properties is always relative to an agent or observer who treats certain physical phenomena as syntactical.

Now why exactly would these consequences be disastrous?

Well, we wanted to know how the brain works, specifically how it produces mental phenomena. And it would not answer that question to be told that the brain is a digital computer in the sense in which stomach, liver, heart, solar system, and the state of Kansas are all digital computers. The model we had was that we might discover some fact about the operation of the brain which would show that it is a computer. We wanted to know if there was not some sense in which brains were *intrinsically* digital computers in a way that green leaves intrinsically perform photosynthesis or hearts intrinsically pump blood. It is not a matter of us arbitrarily or "conventionally" assigning the word "pump" to hearts or "photosynthesis" to leaves. There is an actual fact of the matter. And what we were asking is, "Is there in that way a fact of the matter about brains that would make them digital computers?" It does not answer that question to be told, yes, brains are digital computers because everything is a digital computer.

On the standard textbook definition of computation,

1. For any object there is some description of that object such that under that description the object is a digital computer.

2. For any program there is some sufficiently complex object such that 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 which is isomorphic with the formal structure of Wordstar. But if the wall is implementing Wordstar then if it is a big enough wall it is implementing any program, including any program implemented in the brain.

I think the main reason that the proponents do not see that multiple or universal realizability is a problem is that they do not see it as a consequence of a much deeper point, namely that the "syntax" is not the name of a physical feature, like mass or gravity. On the contrary they talk of "syntactical engines" and even "semantic engines" as if such talk were like that of gasoline engines or diesel engines, as if it could be just a plain matter of fact that the brain or anything else is a syntactical engine.

I think it is probably possible to block the result of universal realizability by tightening up our definition of computation. Certainly we ought to respect the fact that programmers and engineers regard it as a quirk of Turing's original definitions and not as a real feature of computation. Unpublished works by Brian Smith, Vinod Goel, and John Batali all suggest that a more realistic definition of computation will emphasize such features as the causal relations among program states, programmability and controllability of the mechanism, and situatedness in the real world. But these further restrictions on the definition of computation are no help in the present discussion because the really deep problem is that syntax is essentially an observer relative notion. The multiple realizability of computationally equivalent processes in different physical media was not just a sign that the processes were abstract, but that they were not intrinsic to the system at all. They depended on an interpretation from outside. We were looking for some facts of the matter which would make brain processes computational; but given the way we have defined computation, there never could be any such facts of the matter. We can't, on the one hand, say that anything is a digital computer if we can assign a syntax to it and then suppose there is a factual question intrinsic to its physical operation whether or not a natural system such as the brain is a digital computer.

And if the word "syntax" seems puzzling, the same point can be stated without it. That is, someone might claim that the notions of "syntax" and "symbols" are just a manner of speaking and that what we are really interested in is the existence of systems with discrete physical phenomena and state transitions between them. On this view we don't really need 0's and I's; they are just a convenient shorthand. But, I believe, this move is no help. A physical state of a system is a computational state only relative to the assignment to that state of some computational role, function, or interpretation. The same problem arises without 0's and 1's because notions such as computation, algorithm and program do not name intrinsic physical features of systems. Computational states are not *discovered within* the physics, they are *assigned* to the physics.

This is a different argument from the Chinese Room Argument and I should have seen it ten years ago but I did not. The Chinese Room Argument showed that semantics is not intrinsic to syntax. I am now making the separate and different point that syntax is not intrinsic to physics. For the purposes of the original argument I was simply assuming that the syntactical characterization of the computer was unproblematic. But that is a mistake.

There is no way you could discover that something is intrinsically a digital computer because the characterization of it as a digital computer is always relative to an observer who assigns a syntactical interpretation to the purely physical features of the system. As applied to the Language of Thought hypothesis, this has the consequence that the thesis is incoherent. There is no way you could discover that there are, intrinsically, unknown sentences in your head because something is a sentence only relative to some agent or user who uses it as a sentence. As applied to the computational model generally, the characterization of a process as computational is a characterization of a physical system from outside; and the identification of the process as computational does not identify an intrinsic feature of the physics, it is essentially an observer relative characterization.

This point has to be understood precisely. I am not saying there are *a priori* limits on the patterns we could discover in nature. We could no doubt discover a pattern of events in my brain that was isomorphic to the implementation of the vi program on this computer. But to say that something is *functioning as* a computational process is to say something more than that a pattern of physical events is occurring. It requires the assignment of a computational interpretation by some agent. Analogously, we might discover in nature objects which had the same sort of shape as chairs and which could therefore be used as chairs; but we could not discover objects in nature which were functioning as chairs, except relative to some agents who regarded them or used them as chairs.

## V. Second Difficulty: The Homunculus Fallacy is Endemic to Cognitivism

So far, we seem to have arrived at a problem. Syntax is not part of physics. This has the consequence that if computation is defined syntactically then nothing is intrinsically a digital computer solely in virtue of its physical properties. Is there any way out of this problem? Yes, there is, and it is a way standardly taken in cognitive science, but it is out of the frying pan and into the fire. Most of the works I have seen in the computational theory of the mind commit some variation on the homunculus fallacy. The idea always is to treat the brain as if there were some agent inside it using it to compute with. A typical case is David Marr (1982) who describes the task of vision as proceeding from a two-dimensional visual array on the retina to a three-dimensional description of the external world as output of the visual system. The difficulty is: Who is reading the description? Indeed, it looks throughout Marr's book, and in other standard works on the subject, as if we have to invoke a homunculus inside the system in order to treat its operations as genuinely computational.

Many writers feel that the homunculus fallacy is not really a problem, because, with Dennett (1978), they feel that the homunculus can be "discharged". The idea is this: Since the computational operations of the computer can be analyzed into progressively simpler units, until eventually we reach simple flip-flop, "yes-no", "1-0" patterns, it seems that the higher-level homunculi can be discharged with progressively stupider homunculi, until finally we reach the bottom level of a simple flip-flop that involves no real homunculus at all. The idea, in short, is that recursive decomposition will eliminate the homunculi.

It took me a long time to figure out what these people were driving at, so in case someone else is similarly puzzled I will explain an example in detail: Suppose that we have a computer that multiplies six times eight to get forty-eight. Now we ask "How does it do it?" Well, the answer might be that it adds six to itself seven times. [3] But if you ask "How does it add six to itself seven times?", the answer might be that, first, it converts all of the numerals into binary notation, and second, it applies a simple algorithm for operating on binary notation until finally we reach the bottom level at which the only instructions are of the form, "Print a zero, erase a one." So, for example, at the top level our intelligent homunculus says "I know how to multiply six times eight to get forty-eight". But at the next lower-level he is replaced by a stupider homunculus who says "I do not actually know how to do multiplication, but I can do addition." Below him are some stupider ones who say "We do not actually know how to do addition or multiplication, but we know how to convert decimal to binary." Below these are stupider ones who say "We do not know anything about any of this stuff, but we know how to operate on binary symbols." At the bottom level are a whole bunch of homunculi who just say "Zero one, zero one". All of the higher levels reduce to this bottom level. Only the bottom level really exists; the top levels are all just *as-if*.

Various authors (e.g. Haugeland (1981), Block (1990)) describe this feature when they say that the system is a syntactical engine driving a semantic engine. But we still must face the question we had before: What facts intrinsic to the system make it syntactical? What facts about the bottom level or any other level make these operations into zeros and ones? *Without a homunculus that stands outside the recursive decomposition, we do not even have a syntax to operate with.* The attempt to eliminate the homunculus fallacy through recursive decomposition fails, because the only way to get the syntax intrinsic to the physics is to put a homunculus in the physics.

There is a fascinating feature to all of this. Cognitivists cheerfully concede that the higher levels of computation, e.g. "multiply 6 times 8" are observer relative; there is nothing really there that corresponds directly to multiplication; it is all in the eye of the homunculus/beholder. But they want to stop this concession at the lower levels. The electronic circuit, they admit, does not really multiply 6X8 as such, but it really does manipulate 0's and 1's and these manipulations, so to speak, add up to multiplication. But to concede that the higher levels of computation are not intrinsic to the physics is already to concede that the lower levels are not intrinsic either. So the homunculus fallacy is still with us.

For real computers of the kind you buy in the store, there is no homunculus problem, each user is the homunculus in question. But if we are to suppose that the brain is a digital computer, we are still faced with the question "And who is the user?" Typical homunculus questions in cognitive science are such as the following: "How does the visual system compute shape from shading; how does it compute object distance from size of retinal image?" A parallel question would be, "How do nails compute the distance they are to travel in the board from the impact of the hammer and the density of the wood?" And the answer is the same in both sorts of case: If we are talking about how the system works intrinsically neither nails nor visual systems compute anything. We as outside homunculi might describe them computationally, and it is often useful to do so. But you do not understand hammering by supposing that nails are somehow intrinsically implementing hammering algorithms and you do not understand vision by supposing the system is implementing, e.g., the shape from shading algorithm.

## VI. Third Difficulty: Syntax Has No Causal Powers

Certain sorts of explanations in the natural sciences specify mechanisms which function causally in the production of the phenomena to be explained. This is especially common in the biological sciences. Think of the germ theory of disease, the account of photosynthesis, the DNA theory of inherited traits, and even the Darwinian theory of natural selection. In each case a causal mechanism is specified, and in each case the specification gives an explanation of the output of the mechanism. Now if you go back and look at the Primal Story it seems clear that this is the sort of explanation promised by Cognitivism. The mechanisms by which brain processes produce cognition are supposed to be computational, and by specifying the programs we will have specified the causes of cognition. One beauty of this research program, often remarked, is that we do not need to know the details of brain functioning in order to explain cognition. Brain processes provide only the hardware implementation of the cognitive programs, but the program level is where the real cognitive explanations are given. On the standard account, as stated by Newell for example, there are three levels of explanation, hardware, program, and intentionality (Newell calls this last level, the knowledge level), and the special contribution of cognitive science is made at the program level.

But if what I have said so far is correct, then there is something fishy about this whole project. I used to believe that as a causal account the cognitivist's theory was at least false; but I now am having difficulty formulating a version of it that is coherent even to the point where it could be an empirical thesis at all. The thesis is that there is a whole lot of symbols being manipulated in the brain, 0's and 1's flashing through the brain at lightning speed and invisible not only to the naked eye but even to the most powerful electron microscope, and it is these which cause cognition. But the difficulty is that the 0's and 1's as such have no causal powers at all because they do not even exist except in the eyes of the beholder. The implemented program has no causal powers other than those of the implementing medium because the program has no real existence, no ontology, beyond that of the implementing medium. Physically speaking there is no such thing as a separate "program level".

You can see this if you go back to the Primal Story and remind yourself of the difference between the mechanical computer and Turing's human computer. In Turing's human computer there really is a program level intrinsic to the system and it is functioning causally at that level to convert input to output. This is because the human is consciously following the rules for doing a certain computation, and this causally explains his performance. But when we program the mechanical computer to perform the same computation, the assignment of a computational interpretation is now relative to us, the outside homunculi. And there is no longer a level of intentional causation intrinsic to the system. The human computer is consciously following rules, and this fact explains his behavior, but the mechanical computer is not literally following any rules at all. It is designed to behave exactly as if it were following rules, and so for practical, commercial purposes it does not matter. Now Cognitivism tells us that the brain functions like the commercial computer and this causes cognition. But without a homunculus, both commercial computer and brain have only patterns and the patterns have no causal powers in addition to those of the implementing media. So it seems there is no way Cognitivism *could* give a causal account of cognition.

However there is a puzzle for my view. Anyone who works with computers even casually knows that we often do in fact give causal explanations that appeal to the program. For example, we can say that when I hit this key I got such and such results because the machine is implementing the vi program and not the emacs program; and this looks like an ordinary causal explanation. So the puzzle is, how do we reconcile the fact that syntax, as such, has no causal powers with the fact that we do give causal explanations that appeal to programs? And, more pressingly, would these sorts of explanations provide an appropriate model for Cognitivism, will they rescue Cognitivism? Could we for example rescue the analogy with thermostats by pointing out that just as the notion "thermostat" figures in causal explanations independently of any reference to the physics of its implementation, so the notion "program", might be explanatory while equally independent of the physics.

To explore this puzzle let us try to make the case for Cognitivism by extending the Primal Story to show how the Cognitivist investigative procedures work in actual research practice. The idea, typically, is to program a commercial computer so that it simulates some cognitive capacity, such as vision or language. Then, if we get a good simulation, one that gives us at least Turing equivalence, we hypothesize that the brain computer is running the same program as the commercial computer, and to test the hypothesis we look for indirect psychological evidence, such as reaction times. So it seems that we can causally explain the behavior of the brain computer by citing the program in exactly the same sense in which we can explain the behavior of the commercial computer. Now what is wrong with that? Doesn't it sound like a perfectly legitimate scientific research program? We know that the commercial computer's conversion of input to output is explained by a program, and in the brain we discover the same program, hence we have a causal explanation.

Two things ought to worry us immediately about this project. First, we would never accept this mode of explanation for any function of the brain where we actually understood how it worked at the neurobiological level. Second, we would not accept it for other sorts of system that we can simulate computationally. To illustrate the first point, consider for example, the famous account of "What the Frog's Eye Tells the Frog's Brain" (Lettvin, et al. 1959 in McCulloch, 1965). The account is given entirely in terms of the anatomy and physiology of the frog's nervous system. A typical passage, chosen at random, goes like this:

> 1. Sustained Contrast Detectors
>
> An unmyelinated axon of this group does not respond when the general illumination is turned on or off. If the sharp edge of an object either lighter or darker than the background moves into its field and stops, it discharges promptly and continues discharging, no matter what the shape of the edge or whether the object is smaller or larger than the receptive field. (p. 239).

I have never heard anyone say that all this is just the hardware implementation, and that they should have figured out which program the frog was implementing. I do not doubt that you could do a computer simulation of the frog's "bug detectors". Perhaps someone has done it. But we all know that once you understand how the frog's visual system actually works, the "computational level" is just irrelevant.

To illustrate the second point, consider simulations of other sorts of systems. I am, for example, typing these words on a machine that simulates the behavior of an old fashioned mechanical typewriter. [4] As simulations go, the word processing program simulates a typewriter better than any AI program I know of simulates the brain. But no sane person thinks: "At long last we understand how typewriters work, they are implementations of word processing programs." It is simply not the case in general that computational simulations provide causal explanations of the phenomena simulated.

So what is going on? We do not in general suppose that computational simulations of brain processes give us any explanations in place of or in addition to neurobiological accounts of how the brain actually works. And we do not in general take "X is a computational simulation of Y" to name a symmetrical relation. That is, we do not suppose that because the computer simulates a typewriter that therefore the typewriter simulates a computer. We do not suppose that because a weather program simulates a hurricane, that the causal explanation of the behavior of the hurricane is provided by the program. So why should we make an exception to these principles where unknown brain processes are concerned? Are there any good grounds for making the exception? And what kind of a causal explanation is an explanation that cites a formal program?

Here, I believe, is the solution to our puzzle. Once you remove the homunculus from the system, you are left only with a pattern of events to which someone from the outside could attach a computational interpretation. Now the only sense in which the specification of the pattern by itself provides a causal explanation is that if you know that a certain pattern exists in a system you know that some cause or other is responsible for the pattern. So you can, for example, predict later stages from earlier stages. Furthermore, if you already know that the system has been programmed by an outside homunculus, you can give explanations that make reference to the intentionality of the homunculus. You can say, e.g., this machine behaves the way it does because it is running vi. That is like explaining that this book begins with a bit about happy families and does not contain any long passages about a bunch of brothers, because it is Tolstoy's *Anna Karenina* and not Dostoevsky's *The Brothers Karamazov*. But you cannot explain a physical system such as a typewriter or a brain by identifying a pattern which it shares with its computational simulation, because the existence of the pattern does not explain how the system actually works *as a physical system*. In the case of cognition the pattern is at much too high a level of abstraction to explain such concrete mental (and therefore physical) events as the occurrence of a visual perception or the understanding of a sentence.

Now, I think it is obvious that we cannot explain how typewriters and hurricanes work by pointing to formal patterns they share with their computational simulations. Why is it not obvious in the case of the brain?

Here we come to the second part of our solution to the puzzle. In making the case for Cognitivism we were tacitly supposing that the brain might be implementing algorithms for cognition, in the same sense that Turing's human computer and his mechanical computer implement algorithms. But it is precisely that assumption which we have seen to be mistaken. To see this, ask yourself what happens when a system implements an algorithm. In the human computer the system consciously goes through the steps of the algorithm, so the process is both causal and logical; logical, because the algorithm provides a set of rules for deriving the output symbols from the input symbols; causal, because the agent is making a conscious effort to go through the steps. Similarly in the case of the mechanical computer, the whole system includes an outside homunculus, and with the homunculus the system is both causal and logical; logical, because the homunculus provides an interpretation to the processes of the machine; and causal, because the hardware of the machine causes it to go through the processes. But these conditions cannot be met by the brute, blind nonconscious neurophysiological operations of the brain. In the brain computer there is no conscious intentional implementation of the algorithm as there is in the mechanical computer either, because that requires an outside homunculus to attach a human computer, but there can't be any nonconscious implementation as there is in the computational interpretation to the physical events. The most we could find in the brain is a pattern of events which is formally similar to the implemented program in the mechanical computer, but that pattern, as such, has no causal powers to call its own and hence explains nothing.

In sum, the fact that the attribution of syntax identifies no further causal powers is fatal to the claim that programs provide causal explanations of cognition. To explore the consequences of this, let us remind ourselves of what Cognitivist explanations actually look like. Explanations such as Chomsky's account of the syntax of natural languages or Marr's account of vision proceed by stating a set of rules according to which a symbolic input is converted into a symbolic output. In Chomsky's case, for example, a single input symbol, S, is converted into any one of a potentially infinite number of sentences by the repeated application of a set of syntactical rules. In Marr's case, representations of a two dimensional visual array are converted into three dimensional "descriptions" of the world in accordance with certain algorithms. Marr's tripartite distinctions between the computational task, the algorithmic solution of the task and the hardware implementation of the algorithm has (like Newell's distinctions) become famous as a statement of the general pattern of the explanation.

If you take these explanations naively, as I do, it is best to think of them as saying that it is just as if a man alone in a room were going through a set of steps of following rules to generate English sentences or 3D descriptions, as the case might be. But now, let us ask what facts in the real world are supposed to correspond to these explanations as applied to the brain. In Chomsky's case, for example, we are not supposed to think that the agent consciously goes through a set of repeated applications of rules; nor are we supposed to think that he is unconsciously thinking his way through the set of rules. Rather the rules are "computational" and the brain is carrying out the computations. But what does that mean? Well, we are supposed to think that it is just like a commercial computer. The sort of thing that corresponds to the ascription of the same set of rules to a commercial computer is supposed to correspond to the ascription of those rules to the brain. But we have seen that in the commercial computer the ascription is always observer relative, the ascription is made relative to a homunculus who assigns computational interpretations to the hardware states. Without the homunculus there is no computation, just an electronic circuit. So how do we get computation into the brain without a homunculus? As far as I know, neither Chomsky nor Marr ever addressed the question or even thought there was such a question. But without a homunculus there is no explanatory power to the postulation of the program states. There is just a physical mechanism, the brain, with its various real physical and physical/mental causal levels of description.

## VII. Fourth Difficulty: The Brain Does Not Do Information Processing

In this section I turn finally to what I think is, in some ways, the central issue in all of this, the issue of information processing. Many people in the "cognitive science" scientific paradigm will feel that much of my discussion is simply irrelevant and they will argue against it as follows:

> There is a difference between the brain and all of these other systems you have been describing, and this difference explains why a computational simulation in the case of the other systems is a mere simulation, whereas in the case of the brain a computational simulation is actually duplicating and not merely modeling the functional properties of the brain. The reason is that the brain, unlike these other systems, is an *information processing* system. And this fact about the brain is, in your words, "intrinsic". It is just a fact about biology that the brain functions to process information, and since we can also process the same information computationally, computational models of brain processes have a different role altogether from computational models of, for example, the weather. So there is a well defined research question: "Are the computational procedures by which the brain processes information the same as the procedures by which computers process the same information?

What I just imagined an opponent saying embodies one of the worst mistakes in cognitive science. The mistake is to suppose that in the sense in which computers are used to process information, brains also process information. To see that that is a mistake, contrast what goes on in the computer with what goes on in the brain. In the case of the computer, an outside agent encodes some information in a form that can be processed by the circuitry of the computer. That is, he or she provides a syntactical realization of the information that the computer can implement in, for example, different voltage levels. The computer then goes through a series of electrical stages that the outside agent can interpret both syntactically and semantically even though, of course, the hardware has no intrinsic syntax or semantics: It is all in the eye of the beholder. And the physics does not matter provided only that you can get it to implement the algorithm. Finally, an output is produced in the form of physical phenomena which an observer can interpret as symbols with a syntax and a semantics.

But now contrast that with the brain. In the case of the brain, none of the relevant neurobiological processes are observer relative (though of course, like anything, they can be described from an observer relative point of view) and the specificity of the neurophysiology matters desperately. To make this difference clear, let us go through an example. Suppose I see a car coming toward me. A standard computational model of vision will take in information about the visual array on my retina and eventually print out the sentence, "There is a car coming toward me". But that is not what happens in the actual biology. In the biology a concrete and specific series of electro-chemical reactions are set up by the assault of the photons on the photo receptor cells of my retina, and this entire process eventually results in a concrete visual experience. The biological reality is not that of a bunch of words or symbols being produced by the visual system, rather it is a matter of a concrete specific conscious visual event; this very visual experience. Now, that concrete visual event is as specific and as concrete as a hurricane or the digestion of a meal. We can, with the computer, do an information processing model of that event or of its production, as we can do an information processing model of the weather, digestion or any other phenomenon, but the phenomena themselves are not thereby information processing systems.

In short, the sense of information processing that is used in cognitive science, is at much too high a level of abstraction to capture the concrete biological reality of intrinsic intentionality. The "information" in the brain is always specific to some modality or other. It is specific to thought, or vision, or hearing, or touch, for example. The level of information processing which is described in the cognitive science computational models of cognition, on the other hand, is simply a matter of getting a set of symbols as output in response to a set of symbols as input.

We are blinded to this difference by the fact that the same sentence, "I see a car coming toward me", can be used to record both the visual intentionality and the output of the computational model of vision. But this should not obscure from us the fact that the visual experience is a concrete event and is produced in the brain by specific electro-chemical biological processes. To confuse these events and processes with formal symbol manipulation is to confuse the reality with the model. The upshot of this part of the discussion is that in the sense of "information" used in cognitive science, it is simply false to say that the brain is an information processing device.

## VI. Summary of the Argument

This brief argument has a simple logical structure and I will lay it out:

1. On the standard textbook definition, computation is defined syntactically in terms of symbol manipulation.

2. But syntax and symbols are not defined in terms of physics. Though symbol tokens are always physical tokens, "symbol" and "same symbol" are not defined in terms of physical features. Syntax, in short, is not intrinsic to physics.

3. This has the consequence that computation is not discovered in the physics, it is assigned to it. Certain physical phenomena are assigned or used or programmed or interpreted syntactically. Syntax and symbols are observer relative.

4. It follows that you could not *discover* that the brain or anything else was intrinsically a digital computer, although you could assign a computational interpretation to it as you could to anything else. The point is not that the claim "The brain is a digital computer" is false. Rather it does not get up to the level of falsehood. It does not have a clear sense. You will have misunderstood my account if you think that I am arguing that it is simply false that the brain is a digital computer. The question "Is the brain a digital computer?" Is as ill defined

as the questions "Is it an abacus?", "Is it a book?", "Is it a set of symbols?", "Is it a set of mathematical formulae?"

5. Some physical systems facilitate the computational use much better than others. That is why we build, program, and use them. In such cases we are the homunculus in the system interpreting the physics in both syntactic and semantic terms.

6. But the causal explanations we then give do not cite causal properties different from the physics of the implementation and the intentionality of the homunculus.

7. The standard, though tacit, way out of this is to commit the homunculus fallacy. The homunculus fallacy is endemic to computational models of cognition and cannot be removed by the standard recursive decomposition arguments. They are addressed to a different question.

8. We cannot avoid the foregoing results by supposing that the brain is doing "information processing". The brain, as far as its intrinsic operations are concerned, does no information processing. It is a specific biological organ and its specific neurobiological processes cause specific forms of intentionality. In the brain, intrinsically, there are neurobiological processes and sometimes they cause consciousness. But that is the end of the story. [5]

## Footnotes

[1] For earlier explorations see Searle (1980) and Searle (1984).

[2] This view is announced and defended in a large number of books and articles many of which appear to have more or less the same title, e.g. *Computers and Thought* (Feigenbaum and Feldman, eds. 1963), *Computers and Thought* (Sharples et al, 1988), *The Computer and the Mind* (Johnson-Laird, 1988), *Computation and Cognition* (Pylyshyn, 1985), "The Computer Model of the Mind" (Block, 1990, forthcoming), and of course, "Computing Machinery and Intelligence" (Turing, 1950).

[3] People sometimes say that it would have to add six to itself *eight* times. But that is bad arithmetic. Six added to itself eight times is fifty-four, because six added to itself zero times is still six. It is amazing how often this mistake is made.

[4] The example was suggested by John Batali.

[5] I am indebted to a remarkably large number of people for discussions of the issues in this paper. I cannot thank them all, but special thanks are due to John Batali, Vinod Goel, Ivan Havel, Kirk Ludwig, Dagmar Searle and Klaus Strelau.

## References

Block, Ned (1990), forthcoming "The Computer Model of the Mind".

Boolos, George S. and Jeffrey, Richard C. (1989). *Computability and Logic*. Cambridge University Press, Cambridge.

Dennett, Daniel C. (1978). *Brainstorms: Philosophical Essays on Mind and Psychology*. MIT Press, Cambridge, Mass.

Feigenbaum, E.A. and Feldman, J. eds. (1963). *Computers and Thought*. McGraw-Hill Company, New York and San Francisco.

Fodor, J. (1975). *The Language of Thought*. Thomas Y. Crowell, New York.

Haugeland, John, ed. (1981). *Mind Design*. MIT Press, Cambridge, Mass.

Johnson-Laird, P.N. (1988). *The Computer and the Mind*. Harvard University Press, Cambridge, Mass.

Kuffler, Stephen W. and Nicholls, John G. (1976). *From Neuron to Brain*. Sinauer Associates, Sunderland, Mass.

Lettvin, J.Y., Maturana, H.R., McCulloch, W.S., and Pitts, W.H. (1959). "What the Frog's Eye Tells the Frog's Brain". *Proceedings of the Institute of Radio Engineers*, 47, 1940-1951, reprinted in McCulloch, 1965, pp. 230-255.

Marr, David (1982). *Vision*. W.H. Freeman and Company, San Francisco.

McCulloch, Warren S. (1965). *The Embodiments of Mind*. MIT Press, Cambridge, Mass.

Pylyshyn, Z. (1985). *Computation and Cognition*. MIT Press, Cambridge, Mass.

Searle, John R. (1980). "Minds, Brains and Programs", *The Behavioral and Brain Sciences*. 3, pp. 417-424.

Searle, John R. (1984). *Minds, Brains and Science*. Harvard University Press, Cambridge, Mass.

Sharples, M., Hogg, D., Hutchinson, C., Torrance, S., and Young, D. (1988). *Computers and Thought*. MIT Press, Cambridge, Mass. and London.

Shepherd, Gordon M. (1983). *Neurobiology*. Oxford University Press, New York and Oxford.

Turing, Alan (1950). "Computing Machinery and Intelligence." *Mind*, 59, 433-460.

Synthese (2006) 153:393–416
DOI 10.1007/s11229-006-9099-8

ORIGINAL PAPER

# Why we view the brain as a computer

**Oron Shagrir**

Received: 6 July 2006 / Accepted: 8 August 2006 /
Published online: 20 October 2006

**Abstract** The view that the brain is a sort of computer has functioned as a theoretical guideline both in cognitive science and, more recently, in neuroscience. But since we can view every physical system as a computer, it has been less than clear what this view amounts to. By considering in some detail a seminal study in computational neuroscience, I first suggest that neuroscientists invoke the computational outlook to explain regularities that are formulated in terms of the information content of electrical signals. I then indicate why computational theories have explanatory force with respect to these regularities: in a nutshell, they underscore correspondence relations between formal/mathematical properties of the electrical signals and formal/mathematical properties of the represented objects. I finally link my proposal to the philosophical thesis that content plays an essential role in computational taxonomy.

**Keywords** Computation · Content · Information · Explanation

A central working hypothesis in cognitive and brain sciences is that the brain is a sort of a computer. But what, exactly, does it mean to say that an organ or a system such as the brain is a computer? And why do scientists take a computational approach to brain and cognitive function? In addressing these questions, I will put forward a revisionary account of computation that makes two radical claims. First, that everything can be conceived as a computer, and that to be a computer is not a matter of fact or discovery, but a matter of perspective. And second, that representational content plays an essential role in the individuation of states and processes into computational types.

As a friend of brain and cognitive science, and a proponent of the computational approach, one of my objectives is to show that there is no conflict between the view

O. Shagrir (✉)
Departments of Philosophy and Cognitive Science,
The Hebrew University of Jerusalem,
Jerusalem 91905, Israel
e-mail: shagrir@cc.huji.ac.il

I advance here and the foundations of computational cognitive and brain science. If anything, it helps explain why the computational approach has been fruitful.

The paper is organized as follows. In Sect. 1 I review the notorious problem of physical computation, that is, the problem of distinguishing computing physical systems, such as desktops and brains, from physical systems, such as planetary systems, digestive systems, and washing machines, that do not compute. After enumerating the conditions for being a computer that have been adduced in the literature, I conclude that some of the distinguishing features have to do with the way we conceive the systems in question. In this sense, being a computer is, at least in part, a matter of perspective.

Why do we assume the computational outlook when we study the brain, but not when we study other systems? In seeking an answer to this question I examine (in Sect. 2) a study in computational neuroscience (Shadmehr & Wise, 2005). I observe that we apply computational theories when we want to explain how the brain performs a *semantic* task, i.e., a task specified in terms of representational content, and the (computational) explanation consists in postulating an information-processing *mechanism*. From this I conclude that we adopt the computational approach because we seek to explain how a semantic task can be carried out, and computational explanations are able to do this. But what is the source of this explanatory force? Why are computing mechanisms able to explain semantic tasks? I suggest (in Sect. 3) that the explanatory force of a computing mechanism derives from its correspondence to mathematical relations between the represented objects and states. In the last section (Sect. 4), I discuss several objections to my account, and argue that the individuation of a computing mechanism makes an essential reference to these mathematical relations, and, hence, to certain aspects of representational content.

## 1 The problem of physical computation: what does distinguish computers from other physical systems?

Computer models are often used to study, simulate, and predict the behavior of dynamical systems. In most cases, we do not view the modeled system, e.g., the solar system, as a computer. When studying the brain, however, our approach is different. In this case, in addition to using a computer model to simulate the system under investigation, i.e., the brain, we also take the modeled system itself to compute, viewing its dynamical processes as computing processes. Why is this so? What is it about the brain that makes us consider it, along with desktops and pocket calculators, a species of computer, when it doesn't even occur to us to accord that status to solar systems, stomachs and washing machines?

When we talk about computation in the context of mind and brain, we have to distinguish between two categories of computing. One is associated with certain everyday activities, e.g., multiplying 345 by 872. This sort of calculation is done “in-the-head,” often with the aid of a pencil and paper. I call it *mental calculation* to signify that the computing agent is largely aware of the process and “controls” it. In characterizing mental calculation, we take the contrast class to be other mental processes, such as dreaming, mental imagery, deliberating, falling in love, and so forth, and demarcating calculation here does not raise any particular problem. The other category is that of *cerebral computation*. When we say that the brain is a computer, we are assuming that the processes underlying many mental phenomena—mental calculation, dreaming, and so on—are themselves computations. In characterizing cerebral computation, the contrast class is different. Since we take brain processes to be electrical, chemical and biological, the question we should be asking is what makes them computational. What distinguishes these processes from all the other physical, chemical and biological processes, e.g., planetary movements, digestive processes and wash cycles, which are not conceived as computations? And, more generally, what is it that makes some *physical* processes, but not others, computations?

It should be emphasized that I do not seek a precise and definitive answer to the question of what it is that makes us deem a physical process computation. There are, undeniably, fuzzy cases such as look-up tables and infinite-time machines. The puzzle about physical computation does not arise merely because we do not have a precise definition of physical computation, but because we have a hard time coming up with a definition that distinguishes even the most obvious cases of computation from non-computation. There are conditions deemed—correctly, in my opinion—necessary for something to be considered a computer. But astonishingly, these conditions, individually and jointly, fail to differentiate even a single clear-cut case of a computer from a clear-cut case of a non-computer.

Two conditions are often invoked in characterizing computation. The first is that computation is a species of information-processing, or as the saying goes, "no computation without representation" (Fodor, 1981, p. 122; Pylyshyn, 1984, p. 62). What is meant by 'information' or 'representation' here? Our first reaction is that the notion of representation assumed by computation must be restricted. After all, every physical system can be interpreted as representing something. We can, for instance, take planetary systems, stomachs and washing machines to compute the solutions of the mathematical equations that describe their operations; that is, we can construe their states as representing certain numerical values. The question, then, is whether there is a kind of representation that unequivocally distinguishes computing systems from at least some non-computing systems.

One proposal is that suitable representations are those whose content is observer-independent. On this criterion, representations whose content is "derived" are excluded from the computational domain.[1] Our cognitive states are usually classified as representations of the former sort: whether I believe that Bush is a good president is said to be independent of what others think or what they take me to believe.[2] By contrast, the aforementioned construal of states of washing machines as representing numbers does not seem to satisfy the criterion, since this interpretation is observer-dependent: it is we who ascribe to them this representational force.

But this proposal draws the line in the wrong place. On the one hand, it is by no means implausible that planetary systems, stomachs and washing machines have observer-independent representational powers. Planetary systems might carry information about the Big Bang, stomachs about what one has eaten recently, and washing machines about what one has worn recently.[3] On the other hand, digital electronic systems, e.g., desktops, the paradigm cases of computing systems, operate on symbols whose content is, indisputably, observer-dependent. That the states of Deep Junior

[1] The distinction is suggested, e.g., by Dretske (1988), who uses the terms 'natural' and 'conventional,' and Searle (1992), who speaks of the 'intrinsic' and 'non-intrinsic.'

[2] But see Davidson (1990) and Dennett (1971) for a different view.

[3] See Dretske (1988, Chapter 3). Dretske further distinguishes information from (natural) representation, which might improve things somewhat, but cannot account for computing systems that operate on conventional signs.

represent possible states of chessboards is an interpretation we have ascribed to them; we could just as well have ascribed to them very different content.

Another proposal starts from the premise that a computation operates solely on a symbol system, i.e., a system of representations with combinatorial syntax and semantics.[4] Hence, representations that can differentiate computations from non-computations are those that constitute such systems. This criterion seems more plausible than the preceding one, since it appears applicable to all digital electronics, and, arguably, to mental representations as well. But it is far too strict. First, there are all sorts of *analog computers* that range over representations that are not symbolic. Second, the current trend in brain and cognitive science is to view the brain as a computer whose computational processes operate on representations that have no combinatorial structure. This approach is now widespread in the fields of connectionism, neural computation, and computational neuroscience.[5] Whether it is empirically adequate is not at issue here: we are *not* inquiring into whether the computational architecture of the brain is "classical" or "connectionist." We are, rather, inquiring into whether we can describe the brain as a computer even if it turns out that it does not operate on a combinatorial system of representations.

Thus the no-computation-without-representation condition does not advance our investigation at this point. Every system can be seen as representing something. Attempting to distinguish between different sorts of representations turns out to be irrelevant in the computational context. Some computations range over representations whose content is observer-independent, some over representations whose content is observer-dependent. Some computations range over representations whose structure is combinatorial, some over other sorts of representations.

An oft-invoked constraint is what Fodor (1980) calls the "formality condition": a process is computational only if it is formal. But what does formality entail? It has been associated with a number of features, particularly mechanicalness, abstractness, and algorithmicity. Mechanicalness is generally understood in one of two ways. The sense more common in the philosophy of science is that of a mechanism, namely, a causal process underlying a certain phenomenon or behavior: "Mechanisms are entities and activities organized such that they realize of regular changes from start or setup conditions to finish or termination conditions" (Craver, 2002, p. 68). Physical computation is surely mechanical in this sense: it is a causal process that involves changes in entities from initial conditions (inputs) to termination conditions (outputs). But so are the other physical processes, such as planetary movements, digestive processes and wash cycles, which are non-computing.

Another sense of mechanicalness is the logical sense. A process is mechanical in this sense if it is blind to the specific content of the symbols over which it operates. The way the rules of inference function in axiomatic systems is, perhaps, the paradigm case, "the outstanding feature of the rules of inference being that they are purely formal, i.e., refer only to the outward structure of the formulas, not to their meaning, so that they could be applied by someone who knew nothing about mathematics, or by

[4] See, e.g., Newell and Simon (1976), Chomsky (1980), and Fodor and Pylyshyn (1988).

[5] See, e.g., Rumelhart, McLelland, and PDP Research Group (1986), Smolensky (1988), Churchland and Sejnowski (1992), and Churchland and Grush (1999). A specific example is considered in the next section, a propos discussion of the work of Shadmehr and Wise.

a machine" (Gödel, 1933, p. 45).[6] Mechanicalness in this sense is often associated with mental calculation, and is indeed central to the definition of a formal system.[7] But in the context of physical computation, the condition, though satisfied, is not helpful: it does not distinguish computing from non-computing systems. Almost every physical process is mechanical in this sense. Planetary movements, digestive processes and wash cycles proceed regardless of the content one might ascribe to their intermediate states.

A second feature of formality is *abstractness*. This feature refers to a description of the system, that is formulated in terms of an abstract—i.e., mathematical, logical, or "syntactical"—language, and often called a "program."[8] It is often further required that the physical system implement this abstract description, which means, roughly, that its physical states and operations "mirror" the states and operations of the abstract description.[9] Now I agree that the computational description of a system is formulated in terms of some logical or mathematical language, and I have no quarrel with the fact that we can view the system as implementing this program. The problem is that this condition does not advance our goal of distinguishing computing from non-computing systems. After all, planetary movements, digestive processes, and wash cycles are also described by a set of mathematical equations, known as the laws of nature. And, much like digital computers, they can be seen as "implementing" these equations, in the sense that their physical states mirror or otherwise correspond to the states of the describing equations.

A third feature of formality is *algorithmicity*. This feature can be understood as amounting to no more than the combination of the features of mechanicalness and abstractness.[10] But it can also be understood in a stronger sense, as a structural constraint on computing. So understood, a physical computing system is formal in that it does not implement just any abstract structure, for instance, a set of differential equations, but rather, the implemented structure must be of a special sort: a Turing machine, a finite-state automaton, or perhaps some other kind of "digital" or "discrete" process.[11]

[6] This sense of being mechanical is stressed by Fodor: "Formal operations are the ones that are specified without reference to such semantic properties of representations as, for example, truth, reference and meaning" (1980, p. 309).

[7] Cf. Gödel: "Turing's work gives an analysis of the concept of 'mechanical procedure' (alias 'algorithm' or 'computation procedure' or 'finite combinatorial procedure'). This concept is shown to be equivalent with that of a 'Turing machine'. A formal system can simply be defined to be any mechanical procedure for producing formulas, called provable formulas" (in his Postscript to Gödel, 1934, *Collected Works*, vol. 1, pp. 369–370).

[8] The appeal to abstraction is hinted at in Gödel's reference to the "outward structure of the formulas." It is made more explicit by Fodor, who defines computation as "mappings from symbols under syntactic description to symbols under syntactic description" (1994, p. 8).

[9] The notion of implementation is controversial. Roughly, a physical system implements an abstract structure if its physical states "mirror" the operations of the abstract structure in the sense that there is some correspondence between the physical states and the abstract states. For example, if a physical state P corresponds to an abstract state A, and a physical state Q corresponds to an abstract state B, and if P always brings about Q, i.e., the conditional is counterfactual supportive, then A always brings about B. For refinements of this idea see Chalmers (1996) and Scheutz (2001).

[10] In my opinion, computational neuroscientists, e.g., Shadmehr and Wise (see next section), tend to take algorithmicity in this weaker sense.

[11] This feature of formality is stressed in Haugeland (1981). The claim that algorithms are captured by Turing machines is also made in Gödel's comment (note 7 above), and is generally referred to as the Church–Turing thesis. Note, however, that what I am calling "algorithmicity" is close to Gödel's notion

But the algorithmicity condition faces two major difficulties. For one thing, appearances to the contrary, it is satisfied by any physical system. At some level of description, everything is algorithmic: every physical process can be seen as a discrete state-transition process. This point is argued for compellingly by Putnam (1988) and Searle (1992), who further maintain that any physical system implements any "algorithm" whatsoever.[12] It may well be that Putnam and Searle rely on an excessively liberal notion of implementation.[13] But even if they do, their point is well taken. As I have suggested elsewhere (Shagrir, 2001), even under the stricter understandings of implementation, very simple physical devices simultaneously implement very different automata. This indicates, at the very least, that every physical system simultaneously implements several algorithms, even if not every algorithm.

A more serious difficulty is that the algorithmicity condition seems to be inadequate: it draws the line between computing and non-computing descriptions in the wrong place. As we noted, there are analog computers whose processes are considered to be non-algorithmic.[14] It is also possible to conceive of ideal physical systems that are in some sense "digital," yet compute non-recursive functions, i.e., functions that cannot be computed by means of an algorithm.[15] And thirdly, there is the important class of neural networks, artificial and biological, that can be viewed as computing, even though their dynamics are not "digital" in any obvious sense. These dynamics are described by "energy" equations, of the sort that describe many other dynamical systems, including spin glass systems whose particles align in the same direction.[16] Of course, we can count these neural computations as algorithmic, but then, the other such systems must also be deemed algorithmic.

We find ourselves in a conundrum: any physical system computes something, or even many things, and every physical process can be seen as computation. The familiar constraints fail to differentiate computing processes from non-computing processes. While computing is information-processing, and is mechanical and abstract, these features can just as well be said to characterize any physical process. And while some computing processes are algorithmic, in the structural sense outlined above, some computing processes are non-algorithmic.

What are we to make of this? Churchland, Koch, and Sejnowski (1990) state, correctly I think, that "whether something is a computer has an interest-relative

Footnote 11 continued
of a finite procedure. On the relationship between the notions of "finite procedure" and "mechanical procedure," see my "Gödel on Turing on Computability" (2006).

[12] Putnam provides a proof for the claim that every physical system that satisfies certain minimal conditions implements every finite state automaton. Searle claims that every physical process can be seen as executing any computer program.

[13] See Chalmers (1996) and Scheutz (2001).

[14] These machines are algorithmic in the sense that their processes can be approximated by a Turing machine. The problem with this sense of algorithmicity is that it encompasses every physical system, including planetary systems, stomachs and washing machines. But see also the well-known exception presented in Pour-El and Richards (1981).

[15] See Hogarth (1992, 1994). Shagrir and Pitowsky (2003) argue that while the postulated computing machine is digital, in that it consists of two communicating Turing machines, it is not "finite," since it can carry out infinitely many steps in a finite time span. Hence, it does not refute the Church–Turing thesis.

[16] For an extensive discussion of this point, see my "Two Dogmas of Computationalism" (Shagrir, 1997). An artificial network whose dynamics are given by "energy equations" is described in Shagrir (1992). On network models of the brain, see the next section. For a detailed discussion of the relations between the theory of neural networks and statistical mechanics, see Amit (1989).

component, in the sense that it depends on whether someone has an interest in the device's abstract properties and in interpreting its states as representing states of something else" (p. 48). But I wouldn't go as far as Searle (1992) who argues that "computation is not discovered in the physics" (p. 225), that "syntax is not intrinsic to physics" (p. 208), and that "there is no way that computational cognitive science could ever be a natural science, because computation is not an intrinsic feature of the world. It is assigned relative to observers" (p. 212). True, we do not discover that the brain is a computer, but decide to so describe it.[17] But it does not follow, without additional assumptions, that there is nothing here to discover. First, the claim that computation is observer-relative does not imply that what the mind/brain represents is observer-relative. All we have said is that some computations, e.g., desktops, are defined over representations whose content is observer-relative—not that all are. Second, the claim that computing is observer-relative does not entail that the truth-value of the syntactic descriptions is observer-relative. Syntactic descriptions are true (or false) abstract descriptions of what the system does; in this sense, the physical system really implements these abstract structures. The "interest-relative component" is our decision to describe a system in these terms.

Searle's conclusion is, therefore, premature. That being a computer is a matter of perspective does not entail that computational cognitive science (neuroscience) has no empirical content. In particular, it is consistent with their *discovering* (a) the representational contents of the brain—which entities it represents, and (b) the operations that are performed over these representations. It might well be, therefore, that cognitive (and brain) science seeks to discover "the computational structure of the brain": the implemented abstract structure that is defined over the mental (or cerebral) representations.[18]

Searle arrives at his daring conclusion because he thinks that if being a computer is a matter of perspective, then the claim that the brain is a computer "does not get up to the level of falsehood. It does not have a clear sense" (p. 225). Indeed, we have described the brain as a computer without a clear sense of what we are talking about. But it does not follow that our talk is meaningless, just that we still have to unpack the notion of cerebral computation. Given that applying the computational outlook involves some interest-relative component, our task is to clarify what motivates us to apply the computational approach when studying brain and cognitive functions. We have to figure out why we apply it to some systems, e.g., the brain, and not others, e.g., planets or washing machines. The next step in our investigation is to consider this question in the context of a seminal study of computation.

## 2 Why to apply the computational approach: a case study

*The Computational Neurobiology of Reaching and Pointing*, by Shadmehr and Wise, offers a comprehensive treatment of the motor-control problem.[19] I will focus on part II (Chapters 9–14), where Shadmehr and Wise theorize about how the brain might

[17] See also Smith (Smith, 1996, 75 ff.).

[18] Whether computational cognitive science/neuroscience is a natural science in the "naturalistic" sense that computational types make no essential reference to "mental" or "semantic" items is discussed in length in Sect. 4.

[19] I learned about Shadmehr's work at the 2004 Washington University Computational Modeling and Explanation in Neuroscience workshop, both from Frances Egan, who discussed it from a

compute the vector difference between the location of the end-effector (i.e., the hand and objects it controls) and the target location. The discussion is divided into three parts: computing the end-effector location (Chapters 9–10), computing the target location (Chapter 11), and computing the difference vector between the end-effector and target locations (Chapters 12–14). I will not go into the empirical content of the theory in any detail; my aim is to see what we can learn about the philosophical issues at hand from this scientific study.

Like many scientists, Shadmehr and Wise do not provide a comprehensive account of what they mean by "computation." In a section of the introductory chapter entitled "Why a Computational Theory?," they rely mainly on Marr's framework:[20]

> In his often-quoted work on vision, David Marr described three levels of understanding CNS [central nervous system] functions: the level of a *computational theory*, which clarifies the problem to be solved as well as the constraints that physics imposes on the solution; the level of an *algorithm*, which describes a systematic procedure that solves the problem; and the level of *implementation*, which involves the physical realization of the algorithm by a neural network. A computational-level theory thus explains some of what a complex system does and how it *might* work. (pp. 3–4)

There are, however, striking differences between the approach advocated by Marr and the approach taken by Shadmehr and Wise, both with respect to the methodology of investigation and to the characterization of the three levels. Marr's methodology, famously, is top-down, moving from the computational level to the algorithmic, and ultimately, implementation. Computational theory "clarifies the problem" by identifying "the constraints that physics imposes on the solution", where "physics" denotes the physical environment. Shadmehr and Wise, on the other hand, rarely appeal to constraints from the physical environment.[21] In imposing constraints on the problem and its solution, they appeal, more often, to data from evolutionary biology, experimental psychology, robotics, and most of all, neuroscience.[22] Let me bring two examples to illustrate the role played by neurobiological constraints in computational theories.

Shadmehr and Wise's theory about the computation of the target location relies on the three-layer neural net model presented in Zipser and Andersen (1988). The inputs in the model are two sets of cells, one encoding information about eye position (orientation) relative to the head, another encoding the location of the stimulus (target) on the retina, i.e., its retinotopic location. The output is the location of the target in

Footnote 19 continued
philosophical viewpoint, and from Shadmehr's former student, Kurt Thoroughman, who discussed it from an empirical viewpoint.

[20] See Marr (1982), Chapter 1.

[21] It should be noted, however, that Marr's focus on the physical environment is not exclusive, and he also invokes neurobiological data as a constraint on solutions, and that Shadmehr and Wise do, to some extent, take the agent's environment into account. But there is a significant difference in the weight they give these factors.

[22] Evolutionary constraints are addressed in the first part of the book; see also the discussion of the evolutionary advantages of fixation-centered coordinates (p. 185). For behavioral data, see, e.g., Sect. 10.1.2 (pp. 160–162), where the results from behavioral experiments are taken to suggest that the CNS aligns visual and proprioceptive cues to produce an estimate of hand location. For cues from virtual robotics, e.g., in computing the forward kinematics problem, see Sect. 9.6 (pp. 148–151).

a head-centered set of coordinates.[23] The model rests on earlier electrophysiological results by Andersen, Essick, and Siegel (1985), who found three classes of cells in the PPC (in area 7a) of the monkey: (1) cells that respond to eye position only (15% of the sampled cells); (2) cells that are not sensitive to eye orientation (21%), but have an activity field in retinotopic coordinates; and (3) cells that combine information from retinotopic coordinates with information about eye orientation (57%).[24] Based on these constraints, Zipser and Andersen constructed the aforementioned three-layer model so as to have the two sets of inputs in the model correspond to the first two classes of PPC cells, and the hidden layer of the (trained) network correspond to the third class of PPC cells.

The second example pertains to the coordinate system relative to which the computation is carried out: it can be either body-centered, e.g., relative to the head or shoulder, or fixation-centered. These different systems become relevant with respect to the coordinate system in which the CNS represents the end-effector location.[25] The assumption underlying the Zipser and Andersen model is that the CNS uses a head-centered coordinate system. However, more recent data collected by Buneo, Jarvis, Batista, and Andersen (2002) suggests that in area 5d of the PPC, neuronal activity encodes target and hand locations in fixation-centered coordinates.[26] Based on this data, Shadmehr and Wise adjusted their computational theory, adopting visual coordinates, with the fovea as the point of origin.

It is also notable that Shadmehr and Wise differ from Marr in the way they characterize and distinguish between the three levels. Consider the distinction between the computational and algorithmic levels. For Marr, the computational theory “clarifies the problem to be solved,” whereas the algorithmic level “describes a systematic procedure that solves the problem.” For Shadmehr and Wise, the objective of the computational theory is to explain how the system *might* solve the problem, which, they say, amounts to ascertaining the “systematic procedure that solves the problem,” namely, the “the process of computing” (p. 147). This is, of course, precisely what Marr sees as the objective of the algorithmic level. It would thus seem that Shadmehr and Wise do not recognize any significant difference between the levels. As far as they are concerned, to theorize at the computational level is to conjecture, on the basis of all available data, as to how the problem *might* be solved, whereas the ‘algorithmic level’ refers to way it *is* solved.[27]

Or consider the distinction between the algorithmic and implementation levels. Marr reserves the terms ‘computation’ and ‘algorithm’ for a cognitive, e.g., visual, system, and the term ‘implementation’ for their realization in the brain. Shadmehr and Wise draw no such distinction. On the one hand, it is clear that Shadmehr and Wise take the brain itself to be a computer. Indeed, the phrase “the CNS computes” occurs dozens, if not hundreds, of times throughout the book, for example, in the general thesis that “according to the model presented in this book, in order to control

[23] Shadmehr and Wise, pp. 193–197. Shadmehr and Wise diverge from Zipser and Andersen mainly in locating the outputs in a fixation-centered coordinate system, i.e., a visual frame with the fovea as its point of origin (see below).

[24] Shadmehr and Wise, pp. 188–192.

[25] See Shadmehr and Wise, pp. 209–212.

[26] See Shadmehr and Wise, pp. 212–216.

[27] Shadmehr and Wise thus conclude the section “Why a Computational Theory?” with the statement that the book “presents one plausible, if incomplete, framework for understanding reaching and pointing movements” (p. 4).

a reaching movement, the CNS computes the difference between the location of a target and the current location of the end effector" (p. 143). On the other hand, by "algorithm," Shadmehr and Wise do not refer to a sequential, digital, discrete, or controlled process, but to something altogether different. They almost always refer to a multi-directional, parallel, spontaneous, and often analog process, and always describe it by means of a neural network model.[28]

Keeping these differences in mind, let us focus on the objectives of computational theories. According to Shadmehr and Wise, the aim of a computational theory is to *explain* "some of what a complex system does and how it *might* work." This statement raises two questions: (a) What exactly is the explanandum of a computational theory: what are the computational problems that the brain solves? And (b) What exactly is a computational explanation: how does a computational theory explain how these problems are—or might be—solved? I address them in turn.

Two observations about the nature of computational problems come to mind immediately. First, the problem is formulated in terms of regularities: what is being computed is an input–output *function*. Thus in Chapter 9 Shadmehr and Wise consider "the problem of computing end-effector location from sensors that measure muscle lengths or joint angles, a computation called **forward kinematics**" (p. 143). In Chapter 11 they advance a theory, based on the Zipser–Andersen model, that seeks to explain how the CNS computes the target location from information about eye orientation and the target's retinotopic location. And in Chapter 12 they consider the problem of computing the vector difference from information about the locations of the end-effector and the target.

The second observation I want to make is that these regularities are specified in *semantic* terms. By this I mean more than that the inputs and outputs are representational states. I mean, in addition, that the function being computed is specified in terms of the content of these representations: *information* about joint angles, hand and target location, eye orientation and so forth. For example, information about the "end-effector location" is computed from information about "muscle lengths or joint angles": "the CNS computes the difference between the location of a target and the current location of the end effector" (p. 143). However, in calling these terms semantic I do not imply that they are intentional. In computational neuroscience, it is far from clear that the semantic concepts that *scientists* invoke, e.g., representation, information and encoding, refer to much more than stimuli–response causal relations, i.e., a way to interpret neural activity in cells as a response to certain environmental stimuli. The point is simply that regularities in the brain are formulated in these semantic terms.

I also do not suggest that the semantic terms are introduced once we apply the computational approach. To the contrary, assigning content and information to brain/mental states is often *prior* to taking the computational outlook. This is perhaps obvious in the case of computational models of "higher-level" cognitive phenomena. But it is also the case in computational neuroscience. To see this, let us take a closer look at the electrophysiological studies of Andersen et al. (1985). In these experiments

[28] Shadmehr and Wise also advocate a "non-classical," neural network approach in robotics. They write, e.g., that "rather than a symbolic computer program to align "proprioception" with "vision," the imaginary engineer might use one neural network based on feedforwarded connections to map proprioception to vision (forward kinematics) and another network to map vision to proprioception (inverse kinematics). Both of these networks would perform what is called a *function-approximation* computation" (p. 150).

Andersen and his colleagues record PPC neuronal activity from awake monkeys trained in various visuospatial tasks, e.g., fixating on a small point at different eye positions. They then label one group of neurons as "eye-position cells" interpreting their activity as "coding" a range of horizontal or vertical eye positions. They state, more generally, that the "brain receives visual information" and that "at least nine visual cortical fields ... contain orderly representations of the contralateral visual field" (p. 456). Undertaking their computational model, Zipser and Andersen (1988) do not introduce new semantic concepts but build on the interpretation and terminology of the prior electrophysiological studies.[29]

Taken together, we can say that the problems the brain computes are certain regularities or functions that are specified in terms of the representational content of the arguments (inputs) and values (outputs). I call these problems *semantic tasks*. The assertion that it is semantic tasks that are being computed is consistent with other computational studies of the brain.[30] It is also consistent with the computing technology tradition that calculators compute mathematical functions such as addition and multiplication, chess machines compute the next move on the chess-board, and robots compute motor-commands.[31] And it is consistent with much of the philosophical tradition.[32]

Let us now turn to the explanation itself. Perusal of Shadmehr and Wise suggests two features that are associated with computational explanations. First, explanation of a semantic task consists in revealing the (computing) process that mediates between the "initial conditions" (e.g., states that represent the joint angles) and the "termination conditions" (e.g., states that represent the hand location). In this sense, a computational explanation is a sort of *mechanistic explanation*: it explains how the computational problem is solved by revealing the mechanism, or at least a potential mechanism, that solves the problem. The second feature is that the computing process invoked by the explanation satisfies the conditions on computation mentioned in the first section: it is both information-processing and formal, whether the latter is taken to connote mechanicalness or abstractness.

The information-processing character of the process is evident throughout the book. On a single page, we find several statements to this effect: "the *process of computing* motor commands depends crucially on *information* provided by sensory systems"; "the coordinate frame used by the CNS for *representing* hand location reflects this dominance" ; "the idea that your CNS *encodes* hand location in a vision-based coordinates intuitive enough" (p. 147, emphasis added). The mechanical character of computing processes is also abundantly evident in the book, particularly in descriptions of computing processes in engineered robots. And the abstractness of computing

[29] I am grateful to an anonymous referee who encouraged me to add these comments.

[30] See, e.g., Marr (1982), and Lehky and Sejnowski (1988), whose computational theories seek to explain, e.g., how the brain extracts information about an object's shape from information about shading.

[31] E.g., Shadmehr and Wise's discussion of how to approach the problem of forward kinematics in robotics engineering (pp. 148–151): "the engineer's computer program can be said to *align* the mappings of gripper location in *visual and proprioceptive coordinates*" (pp. 148–149).

[32] E.g., Fodor (1994, Chapter 1) states that computational mechanisms are invoked to explain intentional regularities.

processes is manifest in the fact that mathematical language and neural net models are used to describe them.[33]

To gain insight into computational explanation, let us examine a real computational theory, the computational theory at the center of part II of Shadmehr and Wise, which seeks to explain how the brain computes the difference between the target and the current location of the hand. The first step is to divide the problem into sub-problems, each of which is also a *semantic task* that calls for explanation.[34] Shadmehr and Wise divide the problem into three parts: computing hand location from information about joint-angles; computing target location from information about eye-orientation and the retinotopic location of the target, and computing the difference vector from the locations of the hand and target.

The second step is to characterize the sub-tasks in abstract terms, that is, to describe input–output relations by means of, say, mathematical equations. I will focus on the computational problem of the second phase—computing target location from information about eye-orientation and the retinotopic location of the target. This problem is described by the vectorial transformation $[R] + [xR] \rightarrow [Cr]$, where [R] is a vector of the numerical activation values of cells that encode the stimulus (target) location in terms of a retinotopic coordinate system, [xR] stands for the values of electrical discharges that encode eye-location in head-centered coordinates, and [Cr] stands for those that encode the target location in head-centered coordinates.[35]

The third step is to clarify the relations between the abstract inputs, in our case, [R] and [xR], and the output [Cr]. To do this, Shadmehr and Wise appeal to the Zipser–Andersen model, which is a three-layer neural network trained to accomplish the task. The network consists of 64 input units that stand for the visual input [R] and another 32 input units that stand for the eye-position [xR]. Two output representations were used, both of which are meant to represent the stimulus location in head-centered position [Cr]. The number of units in the hidden layer is not specified. The discharge $p_j$ of each cell $j$ in the second and third layers, is a logistic function $1/(1 + e^{-\text{net}})$.[36] After

[33] In the "Why a Computational Theory?" section, Shadmehr and Wise repeatedly associate computational theories with mathematical terms and entities, declaring, e.g., that numbers "enable computations and, therefore, computational theories" (p. 3).

[34] Obviously, the "steps" here serve to highlight the logical structure of the explanation; I am not claiming that the explanation actually proceeds by means of these "steps."

[35] Shadmehr and Wise, p. 194. Two points should be noted. First, this vectorial transformation fits the Zipser–Andersen model, and does not reflect Shadmehr and Wise's preference for fixation-centered coordinates. Second, $[R] + [xR]$ signifies a vectorial combination that is, in fact, non-linear.

As to the computational problems tackled in the other phases, the forward kinematic problem is described in terms of a vector that represents the trigonometric relations between the joint angles and the end-effector location in a shoulder-based coordinate system; see pp. 151–157, and the supplementary information on Shadmehr's website: http://www.bme.jhu.edu/~reza/book/kinematics.pdf. The computational problem tackled in the last phase, computing the difference vector, is described by the equation $X_{\text{dv}} = X_{\text{t}} - X_{\text{ee}}$, where $X_{\text{dv}}$ is the difference vector (target location with respect to end-effector), $X_{\text{t}}$ is the vector representing the target location in fixation-centered coordinates, and $X_{\text{ee}}$ is the vector representing the end-effector location in fixation-centered coordinates.

[36] Net $= \Sigma w_{ij} \cdot p_i$, where $p_i$ is the discharge of cell $i$, and $w_{ij}$ is the weight from $i$ to $j$. Since the network is feed-forward, for each cell, $j$, $w_{ij}$ is defined only for (all) cells in the preceding layer. The initial weights are arbitrary: Zipser and Andersen train the net to find the exact mathematical relations between the input $[R] + [xR]$ and the output [Cr] "by itself."

a training period, in which the weights are modified by means of back-propagation, the network exhibits the behavior $[\mathrm{R}] + [\mathrm{xR}] \rightarrow [\mathrm{Cr}]$.[37]

The fourth and crucial step is to make use of the model to explain the semantic task. The Zipser–Andersen model is meant to explain two semantic features. It explains, first, the pattern of behavior in area 7a, i.e., how cells of the "third group" combine information from retinotopic coordinates with information about eye orientation. These cells are represented in the model by the units in the hidden layer. Analyzing the behavior of the hidden units, Zipser and Andersen found that these cells combine information about the inputs, much like the cells in area 7a. In particular, they behave in accordance with the equation $p_i = (k_i^T e + b_i) \exp(-(r - r_i)^T (r - r_i)/2\sigma^2)$, where $e$ stands for eye-orientation with respect to two angular components, $k_i$ and $b_i$ represent the cell's gain and bias parameters, $r_i$ is the center of the activity field in retinotopic coordinates, and $\sigma$ describes the width of the Gaussian.[38] A second explanatory goal of the model is to show that there can be, at least in principle, cells coding eye-position-independent location of the stimulus. This goal is achieved by demonstrating that the output units extract from the information in the hidden units a coding of the target location in head-centered coordinates. The existence of such "output" cells in the brain, however, is debatable and has not been unequivocally demonstrated.

The last stage in the explanation is to combine the three "local" computational theories into an "integrated" computational theory that explains how the more general problem is solved. This is done via a neural net model that combines the three networks that describe how the sub-tasks are carried out.[39]

We are now in a better position to say why we apply the computational approach to some systems, e.g., brains, but not others, e.g., planetary systems, stomachs and washing machines. One reason is that computational theories seek to explain semantic tasks. We apply the computational approach when our goal is to explain a semantic pattern manifested by the system in question: to explain how the CNS produces certain motor commands, how the visual system extracts information about shape from information about shading, and how Deep Junior generates the command "move the queen to D-5." We do not view planetary systems, stomachs and washing machines as computers because the tasks they perform—moving in orbits, digesting and cleaning—are not defined in terms of representational content. Again, we could view them as computing semantic tasks, in which case we might have applied the computational approach. But we don't: we aim to explain the semantic tasks that desktops, robots and brains perform, but not the semantic tasks that planetary systems, stomachs and washing machines might have performed.[40]

[37] For Zipser and Andersen's presentation of the model, see Zipser and Andersen (1988), Fig. 4, p. 681. For a sketch of a model for forward kinematics, see see Shadmehr and Wise (2005) Fig. 9.3 on p. 149. For a sketch of a model for the third phase (computing the difference vector), see Fig. 12.5 on p. 212. For a detailed mathematical description, see Sect. 12.4 (p. 216 ff.) and Shadmehr's website: http://www.bme.jhu.edu/~reza/book/recurrent_networks.pdf

[38] For further details, see Shadmehr and Wise, pp. 193–197.

[39] A sketch of the network is provided by Shadmehr and Wise in Fig. 12.11 on p. 223.

[40] There is the question of why we view the mind/brain as performing semantic tasks. Answering this question is beyond the analysis of computation: assigning semantic tasks is not part of the computational explanation, but is made prior to taking the computational approach (though the computational study might reveal that the semantic task being solved is not the one we initially attributed to the mind/brain).

A second reason is that computational theories do explain the pertinent semantic tasks that these systems perform. When we look at the Zipser–Andersen model, we *understand*, at least in principle, how the semantic task is carried out. We understand how the CNS might solve the semantic task of converting information about eye-orientation and the target's retinotopic location into information about the target's location in body-centered coordinates. This is not a trivial achievement. It is far from clear that other sorts of explanation, e.g., physical, chemical or biological, have this or any explanatory force with respect to semantic tasks.

So we have made some progress, but our analysis is not yet complete. To better understand why we apply the computational approach we should clarify what is the source of its explanatory force with respect to semantic tasks: we have to better understand the contribution of computational theories in explaining how semantic tasks are carried out.

## 3 On the explanatory force of computational theories

Where do we stand? We noted in the first section that to view something as a computer is to describe its processes as operating on representations, and as being formal, understood as both mechanistic and abstract. But since everything can be described this way, we concluded that computation is a matter of perspective, at least in part. We then asked why we adopt the computational attitude to some systems and not others. Why do we take brains and desktops, but not planetary systems, stomachs and washing machines, to be computers? Surveying the work of Shadmehr and Wise has provided at least a partial answer. The computational approach is an explanatory strategy that seeks to explain a system's execution of semantic tasks. Computational theories in neuroscience explain how the CNS accomplishes tasks such as solving the forward kinematics problem of extracting the end-effector location from information about muscle lengths and joint angles.

We now need to account for the fact that the computational approach gives us what we want, that is, they can explain how semantic tasks are carried out. Let me sharpen the point in need of clarification a bit more. Consider a semantic task of the type $F \rightarrow G$, e.g., converting information about eye-orientation and the retinotopic location of the target ($F$) into information about the target location in body-centered coordinates ($G$).

Let $[\mathrm{R}]+[\mathrm{xR}] \rightarrow [\mathrm{Cr}]$ signify the computing mechanism described in the Zipser–Andersen model, where $[\mathrm{R}]+[\mathrm{xR}]$ is the computational description of the state $F$, and [Cr] that of G. Let us also assume that $N_0 \rightarrow N_n$ is a "low-level" neural mechanism underlying the semantic task, where $N_0$ is, e.g., the electric discharge of certain cells in area 7a correlated with $F$, and $N_n$ that which is correlated with $G$.

Now, when we look at the electrophysiological studies of Andersen et al. (1985) we see that they not only point to the pertinent neurological properties, $N_i$. They already formulate, at least partly, the semantic task $F \rightarrow G$ that is being performed: "many of the neurons can be largely described by the product of a gain factor that is a function of the eye position and the response profile of the visual receptive field. This operation produces an eye position-dependent tuning for locations in head-centered coordinate space" (p. 456).[41] Nevertheless, Andersen et al. (1985) do not provide an explanation for how this semantic operation is produced. A possible explanation is provided only later on by Zipser and Andersen (1988), and, following them, by Shadmehr and Wise (2005), in terms of the computational model.

So what is it about computational models that makes them explanatory with respect to semantic tasks? And what is the explanatory force of the computational mechanism above and beyond its neurobiological counterpart? Before I present my own answers, let me address, very briefly, some answers that have been suggested by others.

One answer cites multiple realization. The claim here is that a semantic task can be implemented in different physical mechanisms, some of which are non-biological.[42] But this answer is not satisfying. First, it is far from obvious that with respect to multiple realization there is any difference between the computational and the neurological. On the one hand, it might well be that different species apply different computational mechanisms in performing a given semantic task. On the other, there is no evidence that the semantic tasks we seek to explain are multiply-realizable in creatures of the same species, e.g., humans.[43] Second, multiple realization does not seem to account for the explanatory force of computational models, e.g., the Zipser–Andersen model, in neuroscience. The explanatory force of the model does not stem from the possibility of being applied to other remote creatures. Its force is in explaining even particular cases of semantic regularities, e.g., how *my* CNS comes to represent the location of a target—that keyboard, say.

Fodor has argued that computational processes, which he views as "mappings from symbols under syntactic description to symbols under syntactic description" (1994, p. 8), are truth-preserving: "It is characteristic of mental processes they [psychological laws] govern that they tend to preserve semantic properties like truth. Roughly, if you start out with a true thought, and you proceed to do some thinking, it is very often the case that the thoughts that the thinking leads you to will also be true" (1994, p. 9). I agree with Fodor that computational processes "preserve semantic properties," but I do not think that he correctly accounts for the explanatory import of this feature. First, if the syntactic processes are truth-preserving, their neural implementations must be truth-preserving too. But this leaves open the question of why these processes, under neural descriptions, do not suffice for explanatory purposes, even though they *are* truth-preserving. Second, Fodor's account is confined to "classical" processes, whereas we are interested in the explanatory power of computational models in general. For the question we seek to answer is *not* that of the empirical adequacy of the classical model of mind.[44] Rather, we are trying to understand why computational models, classical or non-classical, have explanatory power with respect to semantic tasks.

[41] I note again that the task described by Andersen et al. (1985) parallels the transformation between the input and hidden units in Zipser and Andersen (1988).

[42] See, e.g., Fodor (1974) and Putnam (1975). For a version of this argument that highlights complexity considerations see Putnam (1973); in particular, the peg-hole example. For a response, see Sober (1999).

[43] For a detailed discussion of these points with reference to Zipser and Andersen (1988), see Shagrir (1998).

[44] This is, however, the question Fodor addresses. He thus writes: "This emphasis upon the syntactical character of thought suggests a view of cognitive processes in general—including, for example, perception, memory and learning—as occurring in a languagelike medium, a sort of 'language of thought' " (p. 9).

According to Sejnowski, Koch, and Churchland, "mechanical and causal explanations of chemical and electrical signals in the brain are different from computational explanations. The chief difference is that a computational explanation refers to the information content of the physical signals and how they are used to accomplish a task" (1988, p. 1300). Indeed, Zipser and Andersen explain the semantic task $F \rightarrow G$ by referring to $F$, that is, they explain how cells in state $N_n$ encode information, $G$, about the target location in head-centered coordinates, by referring to the information content, $F$, of cells in state $N_0$ in area 7a, i.e., that the electrical signals of some of these cells encode information about stimulus retinotopic-location and that the electrical signals of other cells encode information about eye-orientation. Thus referring to the information content of electrical signals is a central ingredient of computational explanations. The question, still, is why invoke, *in addition*, the computing mechanism to explain the semantic transformation. Why describing the mediating mechanism between $N_0$ and $N_n$ in terms of the computational model, and not in terms of chemical and electrical signals in the brain?

Shadmehr and Wise suggest that a computational theory is an incomplete framework for how the brain "*might* solve these and other problems". A computational theory simply "helps to understand–in some detail—at least one way that a system *could* solve the same problem" (p. 4). In doing this, they serve to bridge between neuroscientists, who study brain function, and engineers, who design and construct mechanical devices,[45] and they promote further research, e.g., additional electrophysiological experiments, by pointing to a set of alternative hypotheses as to how the brain might accomplish the task.[46] On this picture, the explanatory advantage of computational theories is similar to that of mathematical models in other sciences: being more abstract, they underscore regularities that neurological, not to mention molecular, descriptions obscure.[47] But this does not mean that computational descriptions have extra explanatory qualities with respect to semantic tasks that strict neurobiological descriptions lack.

I do not underestimate the role of mathematical models in science in general, and in neuroscience in particular. But I want to insist that computational models have additional and unique explanatory role with respect to semantic tasks. Let me sketch what I take this explanatory role to be. A computational theory in neuroscience aims to explain how the brain extracts information content $G$ from another, $F$. We *know* from electrophysiological experiments that G is encoded in the electrical signals of a neural state $N_n$, and $F$ in those of $N_0$. Let us also assume that we can track chemical and electrical processes that mediate between $N_0$ and $N_n$. What we still want to know is *why* the information content of the electrical signals of $N_n$ is $G$, e.g., target location in head-centered coordinates, and not, say, the target's color. The question arises since the encoded information $G$ are not "directly" related to the target, but is mediated through a representational state that encodes F. The question, in other words, is why a chemical and electrical process that starts in a brain state $N_0$ that encodes $F$, and terminates in another brain state, $N_n$, yields the information content G. After all, this causal process $N_0 \rightarrow N_n$ is not sensitive to what is being represented, but to chemical and electrical properties in the brain.

[45] Shadmehr and Wise, pp. 2–3.

[46] This was suggested by a referee of this article.

[47] A more sophisticated version of this picture, in terms of levels of organization in the nervous system, is presented in Churchland and Sejnowski (1992).

A computational theory explains why it is $G$ that is extracted from $F$ by pointing to correspondence relations between mathematical or formal properties of the representing states—what we call computational structure—and mathematical or formal properties of the represented states. The idea is that by showing that $N_0$ encodes $F$, and that the mathematical relations between $N_0$ and $N_n$ correspond to those of the "worldly" represented states that are encoded as $F$ and $G$, we have shown that $N_n$ encodes $G$.

Let me explicate this point by means of three examples. The first is fictional. The brown–cow cell transforms information about brown things and about cow things into information about brown–cow things. The cell receives electrical inputs from two channels: it receives 50–100 mV from the "brown-channel" just in case there is a brown thing in the visual field and 0–50 mV otherwise, and it receives 50–100 mV from the "cow-channel" just in case there is a cow thing out there and 0–50 mV otherwise (I assume that the information is about the same object). The cell emits 50–100 mV whenever there is a brown cow in the visual field, and 0–50 mV otherwise. How does this cell encode information about brown cows: how does it extract information about brown cows from information about brown things and information about cow things?

The computational explanation is obvious enough. First, we describe the cell as an AND-gate that receives and emits 1's and 0's; these abstracts from emission/reception of 50–100 mV and 0–50 mV. This is the computational structure of the cell. Second, we refer to the information content of the inputs, which are brown things and cow things. And, third, we note that the AND-gate corresponds to a mathematical relation between the represented objects, that is, it corresponds to the *intersection* of the set of brown things and the set of cow things. Taken together, we understand why the information content of emission 50–100 mV is brown-cows: emitting 50–100 mV is an AND result of receiving 50–100 mV, and the AND-operation corresponds to the intersection of the sets of represented objects. Thus given that receiving 50–100 mV from each output channel represents brown things and cow things, emitting 50–100 mV must represent brown–cow things.

Marr's computational theory of early visual processes explains how cells extract information about object-boundaries from retinal photoreceptors that measure light intensities.[48] Marr and Hildreth suggest that the activity of the former cells, known as edge-detectors, can be described in terms of zero crossings in the mathematical formula $(\nabla^2 G) * I(x, y)$, where $I(x, y)$ is the array of light intensities (retinal image), $*$ is a convolution operator, $G$ is a Gaussian that blurs the image, and $\nabla^2$ the Laplacian operator $(\partial^2/\partial x^2 + \partial^2/\partial y^2)$ that is sensitive to sudden intensity changes in the image. Thus at one level, this formula describes a certain mathematical relation between the electrical signals of the edge-detectors and those of the photoreceptors that constitute the retinal image, $I(x, y)$. But this alone does not explain why the activity of the edge-detectors corresponds to object boundaries and other discontinuities in surface properties.[49] The explanation is completed when we note that these mathematical properties correspond to certain mathematical properties of the represented states, for example, of sudden changes in light reflection along boundaries of objects. Had the reflection laws been very different, the mathematical formula would still have

[48] Marr (1982, Chapter 2).

[49] Thus Marr (1982, p. 68) states that, at this point, it is better not to use the word 'edge', since it "has a partly physical meaning—it makes us think of a real physical boundary, for example—and all we have discussed so far are the zero values of a set of roughly band-pass second-derivative filters" (p. 68).

described the same cellular activity, but the cells would no longer carry information about object boundaries.[50]

Let us return to the Zipser–Andersen model. Analyzing the activity of the units in the intermediate layer, Zipser and Andersen found that these units behave like cells in area 7a that combine information about the target's retinotopic location with information about eye-orientation. It was then found that the mathematical description of the activity is $(k_i^T e + b_i)\exp(-(r - r_i)^T(r - r_i)/2\sigma^2)$, which might explain how these cells encode the combined information. The explanation refers to the content of the first two groups of cells in area 7a, and, in particular, that the parameter $e$ stands for eye-orientation with respect to two angular components, and $r_i$ is the center of the activity field in retinotopic coordinates. But it also establishes the findings of the earlier electrophysiological studies: that the described mathematical relations between the electrical signals correspond to certain mathematical relations between the represented states: for example, that the described mathematical relation $(r - r_i)$ of cellular activity (in the second group) corresponds (roughly) to the distance ("in numbers") of the retinotopic location of the stimulus from the receptive field.[51] Without this and the other correspondence relations between mathematical properties, we could have not explained why the cellular activity described by the mathematical formula conveys combined information about eye-orientation and the retinotopic location of the target.

The gist of my account, then, is as follows. When we describe something as a computer we apply a certain explanatory strategy. We apply this strategy when we seek to explain how a complex system performs a semantic task. The explanatory force of this strategy arises from identifying correspondence relation between the computational structure of the system and certain mathematical properties of the states and objects that are being represented.

## 4 Discussion

The following discussion, in the form of objections and replies, will further elaborate on my position, and situate it in the context of the philosophical discussion about computation and content.

*Objection #1*: It seems that you uphold some version of a "semantic" view of computation, but it is not clear what it amounts to.

*Reply*: Piccinini (2004) has recently characterized the semantic view of computation in terms of the no computation without representation condition. My view is certainly "semantic" in this sense, and as I said, this condition has been widely adopted.[52] But I think that there is another, more significant distinction to be made.

[50] As Marr (p. 68) puts it, if we want to use the word 'edges', we have to say why we have the right to do so. The computational theory justifies this use since it underscores the relations between the structure of the image and the structure of the real world outside.

[51] See also Grush (2001, pp. 160–162) who highlights, in his analysis of Zipser and Andersen (1988) the relations between what he calls the algorithm-semantic interpretation (which I take to be the mathematical properties of the cells) and the environmental-semantic interpretation (which I take to be the objects/states/properties that are being represented).

[52] Thus Piccinini rightly says that "the received view" is that there is no computation without representation. A notable exception is Stich (1983). I discuss Piccinini's objection to the no-computation-without representation condition in objection 6.

Because some—many, in fact—of those who accept that there is no computation without representation also hold computation to be "non-semantic" in the sense that the specification of states and processes as computational types makes no essential reference to semantic properties, including representational content. Indeed, this is, I would say, a more accurate statement of the received view about computation. That is, on the received view, computation is formal, in virtue of which it is—notwithstanding the no computation without representation constraint—non-semantic.[53] There are, however, those who maintain that computation is "semantic" in the sense that computational individuation does make essential reference to representational content. This view has been advanced by Burge (1986) and others with respect to computational theories in cognitive science.[54] I belong to this latter "semantic" camp, though my view is distinctive in crucial respects (see below).

*Objection #2*: You seem to be contradicting yourself. On the one hand, you say that computation is "semantic" in the stronger, individuative, sense. On the other, you uphold the view that computational structure is formal. You thus have to clarify how computational individuation that makes essential reference to content can be compatible with the formality condition.

*Reply*: I accept the idea that computation is formal, in the senses of mechanicalness and abstractness. A computational description is formulated in abstract terms, and what it describes is a causal process that is insensitive to representational content. However, it does not follow from these features that computation is non-semantic. It is consistent with mechanicalness and abstractness that semantic elements play an essential role in the individuation of the system's processes and states into computational types. They play an essential role in picking out "the" computational structure from the set of the abstract structures that a mechanism implements.

To better understand how content constrains computational identity, consider again the brown–cow cell. Assume that receiving/emitting 0–50 mV can be further analyzed: the cell emits 50–100 mV when it receives over 50 mV from each input channel, but it turns out that it emits 0–25 mV when it receives under 25 mV from each input channel, and 25–50 mV otherwise. Now assign "1" to receiving/emitting 25–100 mV and "0" to receiving/emitting 0–25 mV. Under this assignment the cell is an OR-gate. This means that the brown–cow cell simultaneously implements, at the very same time, and by means of the very same electrical activity, two different formal structures. One structure is given by the AND-gate and another by the OR-gate.[55]

Now, each of these abstract structures is, potentially, computational. But only the AND-gate is the computational structure of the system with respect to its being a brown-cow cell, namely, performing the semantic task of converting information about brown things and cow things into information about brown cow things. What determines this, that is, picks out this AND-structure as the system's computational structure, given that OR-structure abstract from the very same discharge? I have

[53] Fodor is the most explicit advocate of this view: "I take it that computational processes are both *symbolic* and *formal*. They are symbolic because they are defined over representations, and they are formal because they apply to representations, in virtue of (roughly) the *syntax* of the representations.... Formal operations are the ones that are specified without reference to such semantic properties of representations as, for example, truth, reference and meaning" (1980, p. 64).

[54] See also Kitcher (1988), Segal (1989, 1991), Davies (1991), Morton (1993), Peacocke (1994, 1999), Shagrir (2001). I have also argued for this "semantic" view in the context of computer science (Shagrir, 1999).

[55] A more complex example is presented in detail in Shagrir (2001).

suggested that it is content that makes the difference: discharges that are greater than 50mV correspond to certain types of content (of cows, browns, and brown cows) and the discharges that are less than 50 mV corresponds to another (their absence). Thus the identity conditions of the process, *when conceived as computational*, are determined, at least partly, by the content of the states over which it is defined.

*Objection #3*: This semantic account of computation cannot be correct. We know that we can assign many different interpretations to the same computational (abstract) structure, e.g., a computer program. This demonstrates that differences in content do *not* determine computational identity, for the computational identity can be the same even where there is a difference in content.

*Reply*: I am not saying that computational taxonomy takes every aspect of content into account; in fact, I do not think it takes specific content into account at all. Rather, it takes into account only mathematical properties of the represented objects; these features of content that have been called 'mathematical content' by Egan (1995) and 'formal content' by Sher (1996).[56] Take the brown–cow cell whose computational structure is given by the AND-gate. It is apparent that the computational identity of the cell would have been the same had the '1' and '0' has been given a different interpretation, e.g., that it is a black-dog cell. Still, these interpretations do have something in common. Their mathematical content is the same: the AND-gate corresponds to the same set-theoretic property in the visual field, i.e., that of the intersection of two sets.

*Objection #4*: But why should we care about how the computational structure is picked out? What matters is that we can *identify* this formal structure without appealing to semantic properties at all, i.e., identify it as an abstraction from the physical properties of the system.

*Reply*: I agree that the computational structure of a physical system is an abstraction from its physical or neurological properties, and as such, is "non-semantic." It is evident that the AND-operation describing the behavior of the brown–cow cell is an abstraction from the cell's electrical activity, viz., that its discharge is 50–100 mV if the stimulus in each input channel exceeds 50 mV, and 0–50 mV otherwise. I also agree that we can arrive at this very same formal description by abstracting from physical properties. I have emphasized, moreover, that Shadmehr and Wise often arrive at computational structure via constraints from the neurobiological level. My point about computation being semantic, therefore, is *not* methodological or epistemic; it is not about the way we arrive at the computational structure, about top-down versus bottom-up.

I am claiming, rather, that (a) we take an abstract (formal) structure to be computational when it plays a role in explaining how a system carries out a pertinent semantic task, and that (b) the explanatory role of this abstract structure consists in its being identified by reference to certain semantic features, i.e., mathematical content. We may find out, one way or another, that a certain mathematical description of a cell's behavior is an AND-operation, which is an abstraction from the cell's electrical activity. However, this mathematical description, on its own, is not computational. Physicists often describe systems in mathematical terms, but no one takes

[56] Sher presents the notion of formal content in her analysis of logical consequence. She argues that it is the formal, e.g., set-theoretic, features of the objects they denote, that make certain relations "logical" ; for a full account, see Sher (1991). Egan introduces the notion of mathematical content in the context of computational theories of vision. I follow her in this regard, but emphasize, like Sher, that formal properties are higher-order mathematical structures.

such descriptions to be computational. We take the description to be computational only in the context of explaining a semantic task, in this case, explaining the behavior of the cell as a brown–cow cell. The explanatory power of the AND-operation is due to its satisfying certain semantic constraints, viz., that it correlates with the intersection of two sets. And having satisfied these constraints is essential for its being identified as the computational structure of this cell. The OR-operation, which is simultaneously implemented by the cell, is not identified as the computational structure of the brown–cow cell; for, unlike the AND-operation, the OR-operation does not satisfy the relevant semantic constraints, and thus cannot serve to explain the behavior of the cell as a brown–cows detector.

*Objection #5*: Your semantic account cannot be correct simply because there can be computation without representation. In his "Computation without Representation" (forthcoming), Piccinini argues that "although for practical purposes the internal states of computers are usually ascribed content by an external semantics, this need not be the case and is unnecessary to individuate their computational states and explain their behavior."

*Reply*: In this paper I have focused on the computational approach in neuroscience and cognitive science. In these disciplines, even a glimpse at ongoing research suffices to make apparent the intimate connection between computation and representation. In other contexts, the connection may be less obvious, but I believe that careful scrutiny of any process described as computation will reveal the centrality of representations. I agree with Piccinini's assertion that the "mathematical theory of computation can be formulated without assigning any interpretation to the strings of symbols being computed." But while this means that the mathematical properties of computations can be studied without reference to semantic values, it does not entail that the processes themselves, qua computations, are not identified by their semantic values.

*Objection #6*: A semantic view of computation undermines the whole objective of computational theories of cognition, which is to explain content in non-semantic terms. As Piccinini puts it, an important motivation for many supporters of the computational theory of mind [CTM] is that "it offers (a step towards) a naturalistic explanation of mental content" (2004: 376). This objective is incompatible with a semantic view of computation, for "one problem with naturalistic theories of content that appeal to computational properties of mechanisms is that, when conjoined with the semantic view of computational individuation, they become circular" (Piccinini, forthcoming). By adopting the semantic view of computation, in other words, "we are back to explaining contents, which is what we were hoping to explain in the first place" (2004, p. 377).

*Reply*: I agree that CTM is entrenched in our philosophical landscape: many philosophers maintain that computational theories explain in non-semantic terms phenomena that are formulated in semantic terms. But in my opinion this philosophical outlook is flawed. Computational explanations are "semantic" in two ways: they refer to the specific content of the initial states, and they invoke a computing mechanism (structure) which is individuated by reference to (mathematical) content. I thus also agree that "naturalistic theories of content that appeal to computational properties of mechanisms ... become circular." But I also note that a semantic view of computation is consistent with a naturalistic approach to content. A theory of content that does not appeal to computation, namely, a causal or teleological theory, might well "naturalize" computation-in-the-brain, e.g., by accounting for the pertinent contents in non-semantic terms.[57]

*Objection #7*: Your semantic notion is only one way of accounting for computation. But there could be others that do not appeal to semantic features. You considered only non-semantic accounts, according to which computations are abstractions from internal physical properties. But there could be other, broader accounts, which take into consideration not only internal physical properties, but distal stimuli and responses as well.[58]

*Reply*: I have provided an analysis of the concept of computation as it is applied to physical systems, and as it functions in neuroscience. The analysis, if correct, reveals that when we apply the computational strategy, we individuate states and processes by taking certain aspects of content into account. This insight helps to explain why we apply the computational strategy to brains but not to washing machines, and why we apply the computational strategy, rather than other strategies, in studying how the brain functions. Other accounts of computation did not suggest themselves in the course of the analysis, certainly not broader accounts: Shadmehr and Wise construe computations as processes that take place within the brain, and explicate the relations between the brain and the external environment via the notion of representation. As I just mentioned, I do not rule out the possibility that the pertinent semantic relations can be reduced, e.g., to certain causal relations involving distal stimuli and responses. And, admittedly, I cannot rule out the possibility that this reduction might give rise to a broader, non-semantic, account of computation. I doubt such an account will be forthcoming, but my reasons for this pessimism will have to wait for another occasion.

**Acknowledgements** Thanks to Arnon Levy, Gualtiero Piccinini, Haim Sompolinsky, Jonathan Yaari and two anonymous referees for many insightful comments. This research was supported by The Israel Science Foundation (Grant No. 857/03).

## References

Amit, D. J. (1989). *Modelling brain function*. Cambridge: Cambridge University Press.

Andersen, R. A., Essick, G. K., & Siegel, R. M. (1985). Encoding of spatial location by posterior parietal neurons. *Science, 230*(4724), 456–458.

Buneo, C. A., Jarvis, M. R., Batista, A. P., & Andersen, R. A. (2002). Direct visuomotor transformations for reaching. *Nature, 416*, 632–636.

Burge, T. (1986). Individualism and psychology. *Philosophical Review 95*, 3–45.

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

Chomsky, N. (1980). *Rules and representations*. New York: Columbia University Press.

Churchland, P. S., & Grush, R. (1999). Computation and the brain. In R. A. Wilson, & F. C. Keil (Eds.), *The MIT Encyclopedia of the cognitive sciences* (pp. 155–158). Cambridge, MA: MIT Press.

Churchland, P. S., Koch, C., & Sejnowski T. J. (1990). What is computational neuroscience? In E. L. Schwartz (Ed.), *Computational neuroscience* (pp. 46–55.) Cambridge, MA: MIT Press.

Churchland, P. S., & Sejnowski, T. J. (1992). *The computational brain*. Cambridge, MA: MIT Press.

Craver, C. F. (2002). Structures of scientific theories. In P. Machamer, & M. Silberstein (Eds.), *The Blackwell Guide to the Philosophy of Science*. (pp. 55–79.) Oxford: Blackwell.

[57] See Piccinini (2004). This would not mean, however, that this alleged naturalistic account can be applied to all computers, for there are other computers that operate over "non-mental" representations.

[58] See, e.g., Wilson (1994) and Piccinini (forthcoming).

Davidson, D. (1990). Turing's Test. In K. A. Mohyeldin Said, W. H. Newton-Smith, R. Viale, & K. V. Wilkes (Eds.), *Modeling the mind* (pp. 1–11.) Oxford: Oxford University Press.

Davies, M. (1991). Individualism and perceptual content. *Mind, 100*, 461–484.

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

Dretske, F. (1988). *Explaining behavior*. Cambridge, MA: MIT Press.

Egan, F. (1995). Computation and content. *Philosophical Review, 104*, 181–203.

Fodor, J. A. (1974). Special sciences, or the disunity of science as a working hypothesis. *Synthese, 28*, 97–115.

Fodor, J. A. (1980). Methodological solipsism considered as a research strategy in cognitive psychology. *Behavioral and Brain Sciences, 3*, 63–73.

Fodor, J. A. (1981). The mind-body problem. *Scientific American, 244*, 114–123.

Fodor, J. A. (1994). *The elm and the expert*. Cambridge, MA: MIT Press.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition, 28*, 3–71.

Gödel, K. (1933). The present situation in the foundations of mathematics. In S. Feferman, J. W. Dawson, W. Goldfarb, C. Parsons, & R. M. Solovay (Eds.), *Kurt Gödel collected works*, Vol. III (pp. 45–53). New York: Oxford University Press (1995).

Gödel, K. (1934). On undecidable propositions of formal mathematical systems. In S. Feferman, J. W. Dawson, S. C. Kleene, G. H. Moore, R. M. Solovay, & J. van Heijenoort (Eds.), *Kurt Gödel collected works*, Vol. I (pp. 346–371). New York: Oxford University Press (1986).

Grush, R. (2001). The semantic challenge to computational neuroscience. In P. Machamer, R. Grush, & P. McLaughlin (Eds.), *Theory and method in the neurosciences* (pp. 155–172). Pittsburgh, PA: University of Pittsburgh Press.

Haugeland, J. (1981). Semantic engines. In J. Haugeland (Ed.), *Mind design* (pp. 1–34). Cambridge, MA: MIT Press.

Hogarth, M. L. (1992). Does General Relativity allow an observer to view an eternity in a finite time? *Foundations of Physics Letters, 5*, 173–181.

Hogarth, M. (1994). Non-Turing computers and non-Turing computability. *Proceedings of the Philosophy of Science Association, 1*, 126–138.

Kitcher, P. (1988). Marr's computational theory of vision. *Philosophy of Science, 55*, 1–24.

Lehky, S. R., & Sejnowski, T. J. (1988). Network model of shape-from-shading: Neural function arises from both receptive and projective fields. *Nature, 333*, 452–454.

Marr, D. (1982). *Vision*. San Francisco: W. H. Freeman.

Morton, P. (1993). Supervenience and computational explanation in vision theory. *Philosophy of Science, 60*, 86–99.

Newell, A., & Simon, H. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the Association for Computing Machinery, 19*, 113–126.

Peacocke, C. (1994). Content, computation, and externalism. *Mind and Language, 9*, 303–335.

Peacocke, C. (1999). Computation as involving content: A response to Egan. *Mind and Language, 14*, 195–202.

Piccinini, G. (2004). Functionalism, computationalism, and mental contents. *Canadian Journal of Philosophy, 34*, 375–410.

Piccinini, G. (forthcoming). Computation without representation. *Philosophical Studies*.

Pour-El, M. B., & Richards, I. (1981). The wave equation with computable initial data such that its unique solution is not computable. *Advances in Mathematics, 39*, 215–239.

Putnam, H. (1973). Reductionism and the nature of psychology. *Cognition, 2*, 131–146.

Putnam, H. (1975). Philosophy and our mental life. In H. Putnam (Ed.), *Mind, language and reality*, philosophical papers, volume 2 (pp. 291–303). Cambridge: Cambridge University Press.

Putnam, H. (1988). *Representations and reality*. Cambridge, MA: MIT Press.

Pylyshyn, Z. W. (1984). *Computation and cognition.* Cambridge, MA: MIT Press.

Rumelhart, D. E., McLelland, J. L., & the PDP Research Group (1986). *Parallel distributed processing*, Vol. 1–2. Cambridge, MA: MIT Press.

Scheutz, M. (2001). Computational versus causal complexity. *Minds and Machines, 11*, 543–566.

Searle, J. R. (1992). *The rediscovery of the mind*. Cambridge, MA: MIT Press.

Segal, G. (1989). Seeing what is not there. *Philosophical Review, 98*, 189–214.

Segal, G. (1991). Defense of a reasonable individualism. *Mind, 100*, 485–494.

Sejnowski T. J., Koch, C., & Churchland, P. S. (1988). Computational neuroscience. *Science, 241*(4871), 1299–1306.

Shadmehr, R., & Wise, S. P. (2005). *The computational neurobiology of reaching and pointing: A foundation for motor learning*. Cambridge, MA: MIT Press.

Shagrir, O. (1992). A neural net with self-inhibiting units for the n-queens problem. *International Journal of Neural Systems, 3*, 249–252.

Shagrir, O. (1997). Two dogmas of computationalism. *Minds and Machines, 7*, 321–344.

Shagrir, O. (1998). Multiple realization, computation and the taxonomy of psychological states. *Synthese, 114*, 445–461.

Shagrir, O. (1999). What is computer science about? *Monist, 82*, 131–149.

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

Shagrir, O. (2006). Gödel on turing on computability. In A. Olszewski, J. Wolenski, & R. Janusz (Eds.), *Church's thesis after 70 years* (pp. 393–419). Frankfurt: Ontos Verlag.

Shagrir, O., and Pitowsky, I. (2003). Physical hypercomputation and the Church–Turing thesis. *Minds and Machines, 13*, 87–101.

Sher, G. Y. (1991). *The bounds of logic: A generalized viewpoint*. Cambridge, MA: MIT Press.

Sher, G. Y. (1996). Did Tarski commit "Tarski's Fallacy"? *Journal of Symbolic Logic, 61*, 653–686.

Smith, B. C. (1996). *On the origin of objects*. Cambridge, MA: MIT Press.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences, 11*, 1–23.

Sober, E. (1999). The multiple realizability argument against reductionism. *Philosophy of Science, 66*, 542–564.

Stich, S. P. (1983). *From folk psychology to cognitive science*. Cambridge, MA: MIT Press.

Wilson, R. A. (1994). Wide computationalism. *Mind, 103*, 351–372.

Zipser, D., & Andersen, R. A. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature, 331*, 679–684.

# The Mind as Neural Software?

## Revisiting Functionalism, Computationalism, and Computational Functionalism[1]

Gualtiero Piccinini
Department of Philosophy
University of Missouri – St. Louis
599 Lucas Hall (MC 73)
One University Blvd.
St. Louis, MO 63121-4499 USA
Email: piccininig@umsl.edu

12/20/2006

## Abstract

Defending or attacking either functionalism or computationalism requires clarity on what they amount to and what evidence counts for or against them. My goal here is not to evaluate their plausibility. My goal is to formulate them and their relationship clearly enough that we can determine which type of evidence is relevant to them. I aim to dispel some sources of confusion that surround functionalism and computationalism, recruit recent philosophical work on mechanisms and computation to shed light on them, and clarify how functionalism and computationalism may or may not legitimately come together.

## 1 Introduction

Functionalism is forty years old, computationalism is over sixty, and philosophers often conjoin them. Yet their relationship remains obscure. With Jerry Fodor, I am struck by "the widespread failure to distinguish the computational program in psychology from the functionalist program in metaphysics" (Fodor 2000, 104). A recent paper by Paul Churchland (2005) epitomizes such a failure. Churchland argues that functionalism is false, because the brain is not a classical (i.e., more or less Turing-machine-like) computing mechanism but a connectionist one. His argument presupposes that functionalism entails computationalism, and that the computationalism thus entailed belongs to the classical variety. But functionalism—properly understood—does not entail computationalism, either classical or non-classical.

Defending or attacking either functionalism or computationalism requires clarity on what they amount to and what evidence counts for or against them. My goal here is not to evaluate their plausibility. My goal is to formulate them and their relationship clearly enough that we can determine which type of evidence is relevant to them. I aim to dispel some sources of confusion that surround functionalism and computationalism, recruit recent philosophical work on mechanisms and computation to shed light on them, and clarify how functionalism and computationalism may or may not legitimately come together.

I will frame the discussion in terms of functionalism, because functionalism is the metaphysical doctrine most closely associated (and conflated) with computationalism.[2] But one upshot of this paper is that functionalism and computationalism need not go together. Functionalism may be combined with a non-computationalist theory of mind, and computationalism may be combined with a non-functionalist metaphysics. Once we understand how functionalism and computationalism may or may not be combined, we can generalize our picture to understand how metaphysical doctrines other than functionalism may be combined with computationalism as well as how theories other than computationalism may be combined with functionalism.

To a first approximation, functionalism is the view that the mind is the functional organization of the brain, or any other system that is functionally equivalent to the brain. Sometimes functionalism is expressed by saying that mental states are functional states.[3] Stronger or weaker versions of functionalism may be formulated depending on how much of the mind is deemed to be functional—how many mental states, or which aspects thereof, are functional. Are all mental states functional, or only some? Are all aspects of mental states functional, or only, say, their non-phenomenal aspects? How these questions are answered makes no difference here, because I'm not concerned with the plausibility of functionalism in general. I'm concerned with what functionalism amounts to. One question I will address is, what is functional organization? In due course, I will examine different notions of functional organization and search for the pertinent one.

Computationalism, for present purposes, is the view that the functional organization of the brain is computational, or that neural states are computational states. Again, stronger or weaker versions of computationalism may be formulated depending on how much of the functional organization of the brain is deemed to be computational. But again, I will not assess the plausibility of computationalism here.

Functionalism plus computationalism equals *computational functionalism*. In a well-known slogan, computational functionalism says that the mind is the software of the brain. Taken at face value, this slogan draws an analogy between the mind and the software of ordinary digital computers. But the same slogan is often understood to suggest, more modestly, that the mind is the computational organization of the brain—or equivalently, that mental states are computational states—without the implication that such a computational organization is analogous to that of a digital computer. As we shall see, the ambiguity between the strong and the weak reading is one source of confusion in this area.

Computational functionalism has been popular among functionalists who are sympathetic to computationalist research programs in artificial intelligence, psychology, and neuroscience. It has also generated a fierce opposition.[4] The present goal, however, is not to determine whether the mind is the software of the brain. It is to understand what this means and what counts as evidence for or against it.

## 2 The Analogy between Minds and Computers

Computational functionalism stems from an analogy between minds and computers. But there is more than one analogy, and different analogies pull towards different versions of the view.

Hilary Putnam, the chief founder of computational functionalism, drew an analogy between the individuation conditions of mental states and those of Turing machine states (Putnam 1960, 1967a, 1967b).[5] Putnam noticed that the states of Turing machines are individuated in terms of the way they affect and are affected by other Turing machine states, inputs, and outputs. By the same token, he thought, mental states are individuated by the way they affect and are affected by other mental states, stimuli, and behavior. At first, Putnam did not conclude from this analogy that mental states are Turing machine states, because—he said—the mind is not a causally closed system (Putnam 1960). A bit later, though, Putnam reckoned that mental states can be fully characterized functionally, like those of Turing machines, though he added that the mind might be something "quite different and more complicated" than a Turing machine (Putnam 1967a). Finally, Putnam went all the way to computational functionalism: mental states are (probabilistic) Turing machine states (Putnam 1967b).

As Putnam's trajectory illustrates, the analogy between the individuation of mental states and that of Turing machine states does not entail computational functionalism. The latter conclusion was reached by Putnam some time after drawing his analogy, on independent grounds. What grounds?

It's hard to know for sure, but the relevant papers by Putnam contain references to the plausibility and success of computational models of mental phenomena, including Warren McCulloch and Walter Pitts's theory of the brain. In 1943, McCulloch and Pitts proposed a mathematical theory of neurons and their signals to the effect that, in essence, the brain is a Turing machine (without tape).[6] A few years later, John von Neumann interpreted McCulloch and Pitts's work as *proof* that "anything that can be exhaustively and unambiguously described, anything that can be exhaustively and unambiguously put into words, is ipso facto realizable by a suitable finite neural network" of the McCulloch and Pitts type (von Neumann 1951, 23).

It is now clear that McCulloch and Pitts proved nothing of the sort. For one thing, the nervous system described by their theory is only a simplified and idealized version of the real thing. More importantly, von Neumann's statement implicitly abuses the Church-Turing thesis. The Church-Turing thesis says that anything that is computable in an informal sense, which is intuitively familiar to mathematicians, is computable by Turing machines. From this, it doesn't follow that anything that can be exhaustively and unambiguously described is computable by Turing machines. Nor does it follow, as many would soon conclude, that everything can be simulated by Turing machines or that everything is computational. Alas, neither von Neumann nor his early readers were especially careful about these matters. After von Neumann, fallacious arguments from the Church-Turing thesis—sometimes in conjunction with McCulloch and Pitts's actual or purported results—to the conclusion that the mind is computational began to proliferate.[7]

Since McCulloch and Pitts's networks can be simulated by digital computers, von Neumann's (unwarranted) statement entails that anything that can be exhaustively and unambiguously described can be simulated by a digital computer. If you add to this a dose of faith in scientists' ability to describe phenomena—"exhaustively and unambiguously"—you obtain pancomputationalism: at a suitable level of description, everything is computational. Thus, pancomputationalism made its way into the literature. As Putnam put it, "everything is a Probabilistic Automaton [i.e., a kind of Turing machine] under some Description" (Putnam 1967b, 31). Together with Putnam's analogy between minds and Turing machines and the alleged plausibility of computational psychology, pancomputationalism is the most likely ground for Putnam's endorsement of computational functionalism.

For present purposes, the most important thing to notice is that the resulting version of computational functionalism is quite a weak thesis. This remains true if computational functionalism is disengaged from Putnam's appeal to Turing machine states in favor of the thesis that mental states are, more generally, computational states (Block and Fodor 1972). If mental states are computational simply because at some level, everything is computational, then computational functionalism tells us nothing specific about the mind. It is a trivial consequence of the purported general applicability of computational descriptions to the natural world. This version of computational functionalism does not tell us how the mind works or what is special about it. Such a weak thesis stands in sharp contrast with another, which derives from different analogies between minds and computers.[8]

Digital computers, unlike other artifacts, have an endless versatility in manipulating strings of digits.[9] Strings of digits may be systematically interpreted, so computers' activities are often characterized by semantic descriptions. For example, we say that computers do arithmetic calculations, which is an activity individuated in terms of operations on numbers, which are possible referents of digits. This interpretability of the digits manipulated by computers has often been seen as part of the analogy between computers and minds, because mental states are also typically seen as endowed with content. This, in turn, has contributed to making computational theories of mind attractive.[10] But matters of mental content are controversial, and without agreement on mental content, the putative semantic analogy between minds and computers gives us no firm ground on which to explicate computational functionalism. In the next section, I will argue that the semantic properties of computers make no difference for our purposes. Fortunately, we can find firmer ground on which to proceed in yet another analogy.

Computers are versatile because they can store and execute programs. To a first approximation, a program is a list of instructions for executing a task defined over strings of digits. An instruction is also a string of digits, which affects a computer in a special way. Most computers can execute many different (appropriately written) programs—typically, within certain limits of time and memory, they can execute *any* program. Because of this, computers can acquire any number of new capacities simply by acquiring programs. They can also refine their capacities by altering their programs. Just as minds can learn to execute a seemingly endless number of tasks, computers can execute any task, defined over strings of digits, for which they are given an appropriate program.

This special property of computers—their capacity to store and execute programs—gives rise to the special form of explanation that we employ to account for their behavior. How is my desktop letting me write this paper? By executing a word-processing program. How does it allow me to search the web? By executing an Internet browsing program. And so on for the myriad capacities of my computer. These are explanations by program execution:

> An *explanation by program execution* of a capacity *C* possessed by a system *X* is a description that postulates a program *P* for *C* and says that *X* possesses *C* because *X* executes *P*.

Explanation by program execution applies only to systems, such as computers, that have the meta-capacity to execute programs. Other relevant systems include certain kinds of looms and music boxes. Even though computers are not the only class of systems subject to explanation by program execution, computers have other interesting properties that they do not share with other program-executing mechanisms. The main difference is that the processes generated by computer programs depend on the precise configuration of the input data (viz., the input strings of digits) for their application: to each string of input data there corresponds a different computation. Furthermore, typical computers have an internal memory in which they can store and manipulate their own data and programs, and they can perform *any* computation for as long as they have time and memory space.

These remarkable meta-capacities of computers—to manipulate strings of digits and to store and execute programs—suggest a bold hypothesis. Perhaps brains are computers, and perhaps minds are nothing but the programs running on neural computers. If so, then we can explain the multiple capacities that minds exhibit by postulating specific programs for exhibiting those capacities. The versatility of minds would then be explained by assuming that brains have the same special power that computers have: the power to store and execute programs on strings of digits.[11] This is the true source of the computational functionalist slogan: the mind is the software of the brain.

Compare this version of computational functionalism to Putnam's weaker version. Here we have a putative explanation of human behavior, based on an analogy with what explains computers' behavior. This version tells us how the mind works and what's special about it: the brain has the capacity of storing and executing different programs, and the brain's switching between programs explains its versatility. This is quite a strong thesis: of all the things we observe, only brains and computers exhibit such seemingly endless ability to switch between tasks and acquire new skills. Presumably, there are few if any other systems whose behavior is explained in terms of (this type of) program execution.

If we take this formulation of computational functionalism seriously, we ought to find an adequate explication of explanation by program execution. We ought to make explicit what differentiates systems that compute by executing programs from other kinds of system. For if minds are to be interestingly analogous to computers, there must be something that minds and computers share and other systems lack—something that accounts for the versatility of minds and computers as well as the fact that this versatility is explained by program execution. Unfortunately, the received view of software implementation, which is behind the standard view of program execution, does not satisfy this condition of adequacy.

## 3 Troubles with Program Execution

Ever since Putnam (1967b) formulated computational functionalism, the received view of software implementation has been as follows. If there are two descriptions of a system, a physical description and a computational description, and if the computational description maps onto the physical description, then the system is a physical implementation of the computational description and the computational description is the system's software.[12]

The problem with this view is that it turns everything into a computer. As was mentioned in the previous section, everything can be given computational descriptions. For instance, some cosmologists study the evolution of galaxies using cellular automata. According to the received view of software implementation, this turns galaxies into hardware running the relevant cellular automata programs. If satisfying computational descriptions is sufficient for implementing them in the sense in which ordinary computers execute their programs, then everything is a computer executing its computational descriptions. This is not only counterintuitive—it also trivializes the notion of computer as well as the analogy at the origin of computational functionalism. If the mind is the software of the brain in the sense in which certain cellular automata are the software of galaxies, then the analogy between minds and computers becomes an analogy between minds and everything else. As a consequence, the strong version of computational functionalism collapses into something very much like the weak one.

This problem is worsened by the fact that the same system satisfies many computational descriptions. An indefinite number of cellular automata—using cells that represent regions of different sizes, or different time steps, or different state transition rules—map onto the same physical dynamics. Furthermore, an indefinite number of formalisms different from cellular automata, such as Turing machines or C++ programs, can be used to compute the same functions computed by cellular automata. Given the received view of software implementation, it follows that galaxies are running all of these programs at once.[13]

By the same token, brains implement all of their indefinitely many computational descriptions. If the mind is the software of the brain, as computational functionalism maintains, then given the standard view of software implementation, we obtain either indeterminacy as to what the mind is, or that the mind is a collection of indefinitely many computational organizations. This is not a promising metaphysics of mind, nor is it a way of explaining how minds operate by appealing to the programs they execute.[14]

The problem under discussion should not be confused with a superficially similar problem described by Putnam (1988) and Searle (1992). They argue that any physical system implements a large number of computations, or perhaps every computation, because a large number of (or perhaps all) state transitions between computational states can be freely mapped onto the state transitions between the physical states of a system. For example, I can take the state transitions my web browser is going through and map them onto the state transitions my desk is going through, and as a result, my desk implements my web browser. I can establish the same mapping relation between a large number of (or perhaps all) computations and physical systems. From this, Putnam and Searle conclude that the notion of computation is observer-relative in a way that makes it useless to the philosophy of mind. Their argument is based on the received view of software implementation, and we might avoid its conclusion by abandoning the received view.

But even under the received view of software implementation, Putnam and Searle's problem is not very serious. As many authors have noted (e.g., Chrisley 1995, Copeland 1996, Chalmers 1996a, Bontly 1998, Scheutz 2001), the computational descriptions employed by Putnam and Searle are anomalous. In the case of kosher computational descriptions—the kind normally used in scientific modeling[15]—the work of generating successive descriptions of a system's behavior is done by a computer running an appropriate program (e.g., a weather forecasting program), not by the mapping relation. In the sort of descriptions employed in Putnam and Searle's argument, instead, the descriptive work is done by the mapping relation.

In our example, my web browser does not generate successive descriptions of the state of my desk. If I want a genuine computational description of my desk, I have to identify states and state transitions of the desk, represent them by a computational description (thereby fixing the mapping relation between the computational description and the desk), and then use a computer to generate subsequent representations of the state of the desk, while the mapping relation stays fixed. So, Putnam and Searle's alleged problem is irrelevant to genuine computational descriptions. Still, the problem under discussion remains: everything can be given an indefinite number of *bona fide* computational descriptions.

To solve this problem, we must conclude that being described computationally is not sufficient for implementing software, which is to say, we must abandon the received view of software implementation. The same point is supported by independent considerations. We normally apply the word 'software' to specific mechanisms, i.e. computers, which perform activities that are different from the activities performed by other mechanisms such as drills or valves—let alone galaxies. We say that the invention of computers in the 1940s was a major intellectual breakthrough. We have specific disciplines—computer science and computer engineering—that study the peculiar activities and characteristics of computers and only computers. For all these reasons, a good account of software implementation must draw a principled distinction between computers and other systems.

Philosophers have largely ignored this problem, and the charitable reader may legitimately wonder why. A first part of the answer is that philosophers interested in computationalism have devoted most of their attention to explaining mental phenomena, leaving computation *per se* largely unanalyzed.

A second part of the answer is that computationalist philosophers typically endorse the semantic view of computation, according to which computational states are individuated, at least in part, by their content (for a recent example, see Shagrir 2001, 2006). The semantic view appears to offer protection to the received view of software implementation, because independently of the semantic view, it is plausible that only some things, such as mental states, are individuated by their content. If computational states are individuated by their content and content is present only in few things, then explanation by program execution will apply at most to things that have content, and the trivialization of the notion of software is thereby avoided. Unfortunately, the protection offered by the semantic view is illusory. Here, there is no room for the in-depth treatment the semantic view of computation deserves. I have given such a treatment elsewhere (Piccinini 2004c, 2006), and I will only reiterate its results.

First, even the conjunction of the received view of software implementation and the semantic view of computation does not capture the notion of program execution that applies to ordinary computers. Computers (and some automatic looms, for that matter) can execute programs whether or not the digits they manipulate have content, and there are mechanisms that perform computations defined over interpreted strings of digits just like those manipulated by computers but do so without executing programs (e.g. non-programmable calculators). Second, there are computationalists who maintain that content plays no explanatory or individuative role in a computational theory of mind (Stich 1983, Egan 1992, 2003). Conjoining computationalism with the semantic view of computation begs the question of whether computational states are individuated by their content. Finally, and most seriously, the semantic view of computational states is incorrect, because computability theorists and computer designers—i.e., those to whom we should defer in individuating computational states—individuate computational states without appealing to their semantic properties. For these reasons, the semantic view of computation needs to be rejected, and cannot restore to health the received view of software implementation.

To a first approximation, the distinction between computers and other systems can be drawn in terms of explanation by program execution. Computers are among the few systems whose behavior we normally explain by invoking the programs they execute.[16] When we do so, we explain each activity of a computer by appealing to the unique program being executed. A program may be described in many different ways: instructions, subroutines, whole program in machine language, assembly language, or higher level programming language. But modulo the compositional and functional relations between programs and their components at different levels of description, a computer runs one and only one program at any given time. An expert can actually retrieve the unique program run by a computer and write it down, instruction by instruction.

True, modern computers can run more than one program "at once," but this has nothing to do with applying different computational descriptions to them at the same time. It has to do with computers' capacity to devote some time to running one program, quickly switch to another program, quickly switch back, and so forth, creating the impression that they are running several programs at the same time. (Some so-called supercomputers *can* execute many programs in parallel. This is because they have many different processors, i.e., program-executing components. Each processor executes one and only one program at any given time.) A good account of software implementation must say why explanation by program execution applies only to computers and not to other systems, and hence what minds need to have in order for that explanatory strategy to apply to them. To prepare for that, it's time to clarify the relationship between functionalism and computationalism.

## 4 Mechanistic Functionalism

According to functionalism, the mind is the functional organization of the brain. According to computationalism, the functional organization of the brain is computational. These theses are prima facie logically independent—it should be possible to accept one of them while rejecting the other. But according to a popular construal, functional organizations are specified by computational descriptions connecting a system's inputs, internal states, and outputs (Putnam 1967b, Block and Fodor 1972). Under this construal, functional organizations are *ipso facto* computational, and hence functionalism entails computationalism. This consequence makes it impossible to reject computationalism without also rejecting functionalism, which may explain why attempts at refuting functionalism often address explicitly only its computational variety (e.g., Block 1978, Churchland 2005). The same consequence has led to Fodor's recent admission that he and others conflated functionalism and computationalism (2000, 104).

To avoid conflating functionalism and computationalism, we need a notion of functional organization that doesn't beg the question of computationalism. The broadest notion of functional organization is the purely causal one, according to which functional organization includes all causal relations between a system's internal states, inputs, and outputs. Given this notion, functionalism amounts to the thesis that the mind is the causal organization of the brain, or that mental states are individuated by their causal properties. Indeed, this is how functionalism is often formulated. The good news is, this version of functionalism is not obviously committed to computationalism, because prima facie, causal properties are not ipso facto computational. The bad news is, this version of functionalism is too weak to underwrite a theory of mind.

The causal notion of functional organization applies to all systems with inputs, outputs, and internal states. A liberal notion of input and output generates an especially broad causal notion of functional organization, which applies to all physical systems. For instance, every physical system may be said to take its state at time $t_0$ as input, go through a series of internal states between $t_0$ and $t_n$, and yield its state at $t_n$ as output. A more restrictive notion of input and output generates more interesting functional descriptions. For instance, opaque bodies may be said to take light of all wavelengths as input and yield light of only some wavelengths plus thermal radiation as output. Still, the purely causal notion of functional organization is too vague and broad to do useful work in the philosophy of mind (and computation, for that matter). How should the notions of input and output be applied? Which of the many causal properties of a system are relevant to explaining its capacities? Does this purely causal version of functionalism entail computationalism? To answer these questions, we need to restrict our attention to the causal properties of organisms and artifacts that are relevant to explaining their specific capacities.

To fulfill this purpose, we turn to the notion of functional analysis. Functional analysis was introduced in modern philosophy of mind by Jerry Fodor (1965, 1968a). To illustrate functional analysis, he used examples like the camshaft, whose function is to lift an engine's valve so as to let fuel into the piston. The camshaft has many causal properties, but only some of them, such as its capacity to lift valves, are functionally relevant—relevant to explaining an engine's capacity to generate motive power. Fodor argued that psychological theories are functional analyses, like our analysis of the engine's capacity in terms of the functions of its components.

When Fodor defined psychological functional analysis in general, however, he departed from his examples and assimilated psychological functional analyses to computational descriptions.[17] Several other authors developed a similar notion of functional analysis, retaining Fodor's assimilation of functional analyses to computational descriptions (Cummins 1975, 1983, 2002, Dennett 1978, Haugeland 1978, Block 1995). If functional organizations are specified by functional analyses and functional analyses are computational descriptions, then functional organizations are ipso facto computational. The mongrel of functional analysis and computational description is another source of the conflation between functionalism and computationalism.

To avoid this conflation, we need a notion of functional organization that is explanatory—like Fodor et al's—without committing us to the view that every functionally organized system is computational. The recent revival of mechanistic explanation offers us just what we need. Not only does mechanistic explanation fulfill our purposes—it is independently motivated. For an important lesson of recent philosophy of science is that (the relevant kind of) explanation in the special sciences, such as psychology and neuroscience, takes a mechanistic form[18]:

> A *mechanistic explanation* of a capacity $C$ possessed by a system $X$ is a description that postulates spatiotemporal components $A_1, \ldots, A_n$ of $X$, their functions $\boldsymbol{F}$, and $\boldsymbol{F}$'s relevant causal and spatiotemporal relations $\boldsymbol{R}$, and says that $X$ possesses $C$ because (i) $X$ contains $A_1, \ldots, A_n$, (ii) $A_1, \ldots, A_n$ have functions $\boldsymbol{F}$ organized in way $\boldsymbol{R}$**,** and (iii) $\boldsymbol{F}$, when organized in way $\boldsymbol{R}$, constitute $C$.

A mechanistic explanation in the present sense explains the capacities of a system in terms of its components' functions and the way they are organized. Biologists ascribe functions to types of biological traits (e.g., the digestive function of stomachs) and engineers ascribe them to types of artifacts and their components (e.g., the cooling function of refrigerators). The functions ascribed to traits and artifacts are distinct from their accidental effects (e.g., making noise or breaking under pressure), and hence are only a subset of their causal powers. As a consequence, tokens of organs and artifacts that do not perform their functions may be said to malfunction or be defective.

Different variants of mechanistic explanation may be generated by employing different notions of function.[19] Drawing from William Wimsatt's helpful taxonomy, we find three especially pertinent notions (Wimsatt 1972, 4-5). *Perspectival* functions are causal powers that are relevant according to a view or perspective of what the system is doing. *Evaluative* functions are causal powers that contribute to a system's proper functioning. And *teleological* functions are causal powers that contribute to fulfilling the goal(s) of the system or its users.

These three notions are related. Fulfilling goals is one way of functioning properly, especially if proper functioning is defined as fulfilling one's goals, though something may function properly without fulfilling its goals. Functioning properly may be all that is needed to fulfill one's goals, especially if one's goal is to function properly, though something may fulfill its goals without functioning properly. So evaluative and teleological functions may or may not go together. Furthermore, goals and standards of proper functioning define perspectives that we may take towards a system. Thus, as Wimsatt points out, evaluative and teleological functions are special cases of perspectival functions. But the notion of perspectival function is broader than the others: there are perspectives towards a system that have nothing to do with proper functioning or goals.

The above notions of function can be further elaborated and explicated. There is no shortage of literature devoted to that, and I cannot hope to resolve the debate on functions here.[20] For present purposes, I will limit myself to the following caveats.

First, different authors offer slightly different explications of mechanistic explanation, and not all of them employ the word 'function'. But those differences are irrelevant here. All explications of mechanistic explanation can be subsumed under the above template by using the broad notion of perspectival function.

Second, there may be several legitimate notions of mechanistic explanation, corresponding to different legitimate notions of function and different legitimate explanatory practices. Any notion of function that aspires to be relevant here, however, must be naturalistic, in the sense of playing a legitimate role in scientific explanation. Which of the more precise notions of mechanistic explanation is most adequate to account for the explanatory practices that are relevant to the science and metaphysics of mind is a question that need not be resolved here.

Third, any further explication of the notion of function or mechanistic explanation cannot rely on the notion of computation in such a way as to turn all functionally or mechanistically analyzed systems into computing mechanisms, on pain of begging the question of computationalism again. Fortunately, computation plays no such role in current explications of these notions. As a result, mechanistic explanation does not beg the question of computationalism.

This shows that we need not fasten together functions and computations as Fodor and his followers did. When we appeal to the function of camshafts to explain the capacities of engines, our function ascription is part of a mechanistic explanation of the engine's capacities in terms of its components, their functions, and their organization. We do not appeal to programs executed by engines, nor do we employ any kind of computational language. In fact, most people would consider engines good examples of systems that do not work by executing programs (or more generally, by performing computations). The same point applies to the vast majority of mechanisms, with the notable exception of computers and other computing mechanisms (including, perhaps, minds).

Fourth and finally, we should not confuse the teleological notion of function with the etiological account of teleology. The etiological account of teleology in terms of evolutionary history is perhaps the most popular one, but it may not suit our present purposes.[21] What matters here is that teleological functions support a robust notion of functional organization and explanation without relying on the notion of computation. The question of how teleological functions ought to be explicated is surely important, but I can remain neutral about it.[22]

Mechanistic explanations are the primary way of understanding mechanisms in the biological sciences, including neuroscience and psychology, and in engineering, including computer engineering. Investigators in these disciplines analyze systems (e.g., trees) by breaking them down into component parts (e.g., roots) and discovering (or in engineering, designing) the functions of those parts (e.g., supporting the tree and absorbing water from the soil). Neuroscientists and psychologists elaborate their theories in the same way: they partition the brain or mind into components (e.g., the suprachiasmatic nuclei or episodic memory) and they ascribe them functions (respectively, regulating circadian rhythms and storing records of events). Mutatis mutandis, computer engineers do the same thing: they partition a computer into components (e.g., the memory and the processor) and ascribe them functions (respectively, storing data as well as instructions and executing instructions on the data).

Since mechanistic explanation gives us the notion of functional organization that is relevant to understanding theories in psychology, neuroscience, and computer engineering, we should adopt this notion of functional organization in our formulation of functionalism. With mechanistic explanation in place, we can give a novel and improved formulation of functionalism, which does justice to the original motivations of functionalism without begging the question of computationalism. Functionalism about a system *S* should be construed as the thesis that *S* is individuated in terms of a functional organization specified by a mechanistic explanation.

*S*'s mechanistic explanation may or may not attribute computational properties to *S*. Under this mechanistic version of functionalism, a system is individuated by its component parts, their functions, and their relevant causal and spatiotemporal relations. The functional states of the system are individuated by their role within the mechanistic explanation of the system. When a mechanistic explanation of a system is available, the states of the system are not only individuated by their relevant causal relations to other states, inputs, and outputs, but also by the component to which they belong and the function performed by that component when it is in that state. This applies to all mechanisms, including computing mechanisms. For example, ordinary Turing machine states are individuated not only as having the function of generating certain outputs and other internal states on the basis of certain inputs and states, but also as being states *of* the tape or *of* the active device, which are the components of the Turing machine.[23]

Mechanistic functionalism has a further great advantage, which is especially relevant to the concerns of this paper: it is based on a notion of mechanistic explanation that offers us the materials for explicating the notion of explanation by program execution, and more generally, computational explanation.

## 5 Mechanistic explanation, Computation, and Program Execution

A system subject to mechanistic explanation may or may not perform computations, and a system that performs computations—a computing mechanism—may or may not do so by executing programs. For example, Turing machines are made out of a tape of unbounded length, an active device that can take a number of states, letters from a finite alphabet, and relations between tape, active device, states, and letters that are specified by a machine table. Of course, Turing machines are usually thought of as abstract objects, operating on abstract inputs and outputs. But Turing machines can be physically realized, in which case they would operate on concrete counterparts of strings of letters, which I call strings of digits. Whether abstract or concrete, Turing machines are mechanisms, subject to mechanistic explanation no more and no less than other mechanisms.[24]

Some Turing machines can compute only one function. Other Turing machines, called universal, can compute any computable functions. The difference between universal and non-universal machines has a mechanistic explanation. Non-universal Turing machines manipulate the digits on their tape in accordance with their machine table, without executing any program. Universal Turing machines, by contrast, have such a special machine table of their own that they treat some of the digits on their tape as programs and others as data, so as to manipulate their data by appropriately responding to the programs. Because of this, universal Turing machines—unlike non-universal ones—may be said to execute the programs written on their tape. The behavior of all Turing machines is explained by the computations they perform on their data, but only the behavior of universal Turing machines is explained by invoking the execution of programs.

Like Turing machines, most biological systems and artifacts are mechanistically explained in terms of their components and functions (Bechtel and Richardson 1993, Craver and Darden 2001). But unlike Turing machines, the capacities of most biological systems are not explained by appealing to putative computations they perform, let alone programs that they execute (except, of course, in the case of brains and other putative computing mechanisms).

So, explaining a capacity by program execution is not the same as providing a mechanistic explanation of a system. Rather, it is to provide part of a very *specific kind* of mechanistic explanation. Computers are subject to explanation by program execution because they are a peculiar kind of mechanism, and their peculiarity is explained by a specific kind of mechanistic explanation. The goals of this section are twofold: first, to identify the subclass of mechanisms that perform computations and whose (relevant) activities are explained by the computations they perform, and second, to identify the subclass of computing mechanisms that execute programs and whose (relevant) activities are explained by the programs they execute. Once we have an account of these distinctions, we will have the resources to explicate computational functionalism.

Most mechanisms are partially individuated by their normal interactions with their environment. For instance, stomachs are things whose function is to digest food, and refrigerators are things whose function is to lower the temperature of certain regions of space. Environmental interactions, in turn, may be analyzed in terms of inputs received from the environment and outputs delivered to the environment. Stomachs take undigested food as input and yield digested food as output; refrigerators take their inside at a certain temperature as input and deliver the same region at a lower temperature as output. Inputs and outputs may be taxonomized in many ways, which are relevant to the capacities to be explained. In our examples, foods and temperatures are taxonomized, respectively, in terms of whether and how they can be processed by stomachs and refrigerators in the relevant ways. Being a specific kind of mechanism, computing mechanisms are individuated by inputs and outputs of a specific kind and by a specific way of processing those inputs and outputs.

The inputs and outputs that are relevant to computing mechanisms are what computability theorists call strings of letters, or symbols.[25] A string of digits, as I'm using the term, is a concrete counterpart of a string of letters. What does it take for a concrete entity to be a string of digits? I will now sketch an answer in terms of mechanistic explanation. A digit is a particular that belongs to one and only one of a finite number of types. The digits' types are unambiguously distinguishable (and hence individuated) by the effects they have on the mechanism that manipulates them. That is, every digit of the same type affects a mechanism in the same way relative to generating the mechanism's output, and each type of digit affects the mechanism in a different way relative to generating the mechanism's output.

In other words, ceteris paribus, if $T_2 = T_1$, then substituting a digit of type $T_1$ for a digit of type $T_2$ in a string results in the exact same computation with the same output string, whereas if $T_2 {}^{1} T_1$, then substituting a digit of type $T_2$ for a digit of type $T_1$ in a string results in a different computation, which may generate a different output string.[26] This property of digits differentiates them from many other classes of particulars, such as temperatures and bites of food, which belong to indefinitely many types. (There is no well-defined functional classification of temperatures or foods such that every temperature or bite of food belongs to one among a finite number of types).

A string is a list of permutable digits identified by the digits' types, their number, and their order within the string. Every finite string has a first and a last digit member, and each digit member (except for the last member) has a unique successor. A digit within a string can be substituted by another digit without affecting the other digits' types, number, or position within the string. In particular, when an input string is processed by a mechanism, ceteris paribus, the digits' types, their number, and their order within the string make a difference to what output string is generated.

The fact that digits are organized into strings further differentiates strings of digits from the inputs and outputs of other functionally analyzable systems. Neither temperatures nor bites of food are organized into strings in the relevant sense. The comparison is a bit unfair, because neither bites of food nor temperatures are digits to begin with. But let us suppose, for the sake of the argument, that we could find a way to unambiguously taxonomize bites of food into finitely many (functionally relevant) types. For instance, we could taxonomize bites of food into protein bites, fat bites, etc. If such a taxonomy were viable, it would turn bites of food into digits. Still, sequences of bites of food would not constitute *strings* of digits, because digestion—unlike computation—is largely indifferent to the order in which an organism bites its food. Even vending machines, which have been used to illustrate the idea of a computing mechanism (Block 1980), are not computing mechanisms in a very interesting sense, because they yield the same output regardless of the order in which the relevant inputs are inserted into them.

Among systems that manipulate strings of digits, some do so in a special way: under normal conditions, they produce output strings of digits from input strings of digits in accordance with a general rule, which applies to all relevant strings and depends on the inputs (and perhaps the internal states) for its application.[27] The rule in question specifies the *computation* performed by the system. Some systems manipulate strings without performing computations over them. For instance, a genuine random number generator yields strings of digits as outputs, but not on the basis of a general rule defined over strings. (If it did, its output would not be genuinely random.) Systems that manipulate strings of digits in accordance with the relevant kind of rule deserve to be called computing mechanisms.

The activities of computing mechanisms are explained by the computations they perform. For example, if you press the buttons marked ‘21’, ‘:’, ‘7’, and ‘=’, of a (well-functioning) calculator, after a short delay it will display ‘3’. The explanation for this behavior includes the facts that 3 is 21 divided by 7, ‘21’ represents 21, ‘:’ represents division, ‘7’ represents 7, ‘=’ represents equality, and ‘3’ represents 3. But most crucially, the explanation involves the fact that the calculator’s function under those conditions is to perform a specific calculation: to divide its first input datum by the second. The capacity to calculate is explained, in turn, by an appropriate mechanistic explanation. Calculators have input devices, processing units, and output devices. The function of the input devices is to deliver input data and commands from the environment to the processing units, the function of the processing units is to perform the relevant operations on the data, and the function of the output devices is to deliver the results of those operations to the environment. By iterating this explanatory strategy, we can explain the capacities of a calculator’s components in terms of the functions of their components and their organization.

Some computing mechanisms have special components, usually called processors. Processors are capable of performing a finite number of primitive operations on input strings (of fixed length) called data. Which operation a processor performs on its data is determined by further strings of digits, called instructions. Different instructions cause different operations to be performed by a processor. The performance of the relevant operation in response to an instruction is what constitutes the execution of that instruction. A list of instructions constitutes a program. The execution of a program’s instructions in the relevant order constitutes the execution of the program. So, by executing a program’s instructions in the relevant order, a computer processor executes the program. This is a brief mechanistic explanation of (program-controlled) computers, which appeals to kinds of components and their functions. This particular mechanistic explanation (or better, a detailed and complete version of it) explains what it means to execute a program and how computers have this capacity. The capacity of a processor to execute instructions can be further explained by a mechanistic explanation of the processor in terms of its components, their functions, and their organization.[28]

Only computing mechanisms of a specific kind, namely computers, have processors capable of executing programs (and memories for storing programs, data, and results). This is why only the capacities of computers, as opposed to the capacities of other computing mechanisms—let alone mechanisms that do not perform computations—are explained by program execution. Computational explanation by program execution says that there are strings of digits whose function is to determine a sequence of operations to be performed by a processor on its data.

In other words, explanation by program execution presupposes that (a state of) part of the computer *functions* as a program; in an explanation by program execution, 'program' is used as a function term. The way a program determines what the computer is going to do is cashed out in terms of the mechanistic explanation of the computer. So, explanation by program execution presupposes that the system executing the program is a very specific kind of computing mechanism, which has the capacity to execute programs. This is why the appeal to program execution is explanatory for computers—because it postulates programs and processors *inside* the computers.

As a consequence, when the behavior of ordinary computers is explained by program execution, the program is not just a description. The program is also a *physical component* of the computer (or a stable state of a component), whose function is to generate the relevant capacity of the computer. Programs are physically present within computers, where they have a function to perform. Somehow, this simple and straightforward point seems to have been almost entirely missed in the philosophical literature.[29]

## 6 Computational Functionalism

We now have the ingredients to explicate computational functionalism:

*Computational functionalism*: the mind is the software of the brain.

In its strong and literal form, computational functionalism says that (i) the brain contributes to the production of behavior by storing and executing programs, according to the mechanistic explanation sketched in the previous section, and (ii) the mind is constituted by the programs stored and executed by the brain. This doctrine has some interesting consequences for the study of minds and brains.

Computational functionalism licenses explanations of mental capacities by program execution. This is a kind of mechanistic explanation, which explains mental capacities by postulating a specific kind of mechanism with specific functional properties. Briefly, the postulated mechanism includes memory components, which store programs, and processing components, which execute them. Together, the interaction between memories and processors determines how the system processes its data. The capacities of the system are explained as the result of the processing of data performed by the processor(s) in response to the program(s).

Computational functionalism entails that minds are multiply realizable, in the sense in which the same computer program can run on physically different pieces of hardware. So if computational functionalism is correct, then—*pace* Bechtel and Mundale 1999, Shapiro 2000, Churchland 2005 and other foes of multiple realizability—mental programs can also be specified and studied independently of how they are implemented in the brain, in the same way in which one can investigate what programs are (or should be) run by digital computers without worrying about how they are physically implemented. Under the computational functionalist hypothesis, this is the task of psychological theorizing. Psychologists may speculate on which programs are executed by brains when exhibiting certain mental capacities. The programs thus postulated are part of a mechanistic explanation for those capacities.

The biggest surprise is that when interpreted literally, computational functionalism entails that the mind is a physical component (or a stable state of a component) of the brain, in the same sense in which computer programs are physical components (or stable states of components) of computers. As a consequence, even a brain that is not processing any data—analogously to an idle computer, or even a computer that is turned off—might still have a mind, provided that its programs are still physically present. This consequence seems to offend some people's intuitions about what it means to have a mind, but it isn't entirely implausible. It might correspond to the sense in which even people who are asleep, or have fainted, still have minds. Be that as it may, this consequence can be easily avoided by a more dynamic interpretation of computational functionalism, according to which the mind is constituted by the *processes* generated by the brain's software. This dynamic reading may well be the one intended by the original proponents of computational functionalism.

Computational functionalism offers a particular mechanistic explanation of the mind. It describes the mind as a program, which means that the function of the mind is to determine which sequences of operations the brain has to perform. This presupposes a particular mechanistic explanation of the brain as a program-controlled computer, i.e., a mechanism with certain components that have certain functions and a certain organization. Whether a mechanistic explanation applies to a system is an empirical question. In this important respect, computational functionalism turns out to embody a strong empirical hypothesis.

Philosophers of mind have usually recognized that computationalism is an empirical hypothesis in two respects. On the one hand, there is the empirical question of whether a computer can be programmed to exhibit all of the capacities that are peculiar to minds. This is one traditional domain of artificial intelligence. On the other hand, there is the empirical question of whether all mental capacities can be explained by program execution. This is one traditional domain of cognitive psychology. As to neuroscience, computationalists have traditionally considered it irrelevant to testing their hypothesis, on the grounds that the same software can be implemented by different kinds of hardware. This attitude is unsatisfactory in two respects.

First, as we have seen, at least two important construals of functionalism are such that they entail computationalism. But if computationalism is a logical consequence of the metaphysical doctrine of functionalism, then the empirical status of computationalism is tied to that of functionalism: if functionalism is a priori true (as some philosophers believe), then computationalism should need no empirical testing; conversely, any empirical disconfirmation of computationalism should disconfirm functionalism too. An important advantage of my proposed reformulation of functionalism is that it does not entail computationalism. This leaves computationalism free to be an empirical hypothesis about the specific functional organization of the brain, which—when conjoined with functionalism—gives rise to computational functionalism.

But second, if computationalism is an empirical hypothesis to the effect that mental capacities are explained by program execution, it isn't enough to test it by programming computers and attempting to explain mental capacities by program execution. Indeed, to assume that this is the only way of testing it begs the question of whether the brain is the sort of mechanism that could run mental programs at all—whether it is a (program-storing-and-executing) computer. Assuming that the mind is the software of the brain presupposes that the brain has components of the relevant kinds, with the relevant functional and organizational properties.

Whether the brain is the kind of mechanism whose capacities can be explained by program execution is itself an empirical question, and if the brain were not functionally organized in the right way, computational functionalism about the mind would turn out to be false. This shows computational functionalism to incorporate an empirical hypothesis that can be effectively tested only by neuroscience. Whether brains are one kind of mechanism or another can only be determined by studying brains.

This sense in which computational functionalism embodies an empirical hypothesis is more fundamental than the other two. If the brain is a computer, then both classical artificial intelligence and classical cognitive psychology are bound to succeed. But if the brain is *not* a computer, then classical artificial intelligence and cognitive psychology may or may not succeed in ways that depend on the extent to which it is possible to reproduce the capacities of systems that are not computers by executing programs. It may be possible to reproduce all or many mental capacities by computational means even though either the brain is not a computer, or the mind is something other than the programs running on the brain, or both. The extent to which this is possible is a difficult question, which there is no room to fully address here.

I have formulated and discussed computational functionalism using the notion of explanation by program execution, because the analogy between minds and program executing computers is the motivation behind the strong version of computational functionalism. There is no question that many of those who felt the pull of the analogy between minds and computers—such as Turing, von Neumann, Fodor, Newell, and Simon—did so in part because of the explanatory power that comes with explanation by program execution.

But as I noticed at the beginning of this essay, computational functionalism is ambiguous between a strong and a weak reading. It is equally obvious that many other authors, who are (or were at one point) sympathetic to the analogy between minds and computers, such as Putnam, Cummins, the Churchlands, Devitt and Sterelny (1999), and even McCulloch and Pitts (at least in 1943), would resist the conclusion that the brain stores and executes programs. Is there a way to cash out their view without falling into the trivial conclusion that the mind can be described computationally in the sense in which anything else can? Indeed there is.

The account of computational explanation I sketched in Section 5 applies not only to computation by program execution. In fact, computation by program execution is explicated in terms of the more general notion of computation tout court. Roughly, computation is the manipulation of data and (possibly) internal states according to an appropriate rule. (Computation by program execution, then, is computation performed in response to instructions that encode the relevant rule.) Most digital computers compute by executing programs, but ordinary (i.e., non-universal) Turing machines, finite state automata, and many connectionist networks perform computations without executing programs.

To cover theories that do not appeal to program execution, the present formulation of computational functionalism can be generalized by replacing program execution with other computational processes, such as connectionist computation. According to this generalized computational functionalism, the mind is the computational organization of a (computing) mechanism, regardless of whether that mechanism is a program-controlled computer, a connectionist computing mechanism, or any other kind of computing mechanism (e.g., a finite state automaton). Given the generalized formulation, psychological explanations need not invoke the execution of programs—they can invoke either program execution or some other kind of computation (connectionist or otherwise) that is presumed to generate the behavior to be explained. This kind of explanation is still a mechanistic explanation that appeals to the manipulation of strings of digits in accordance with an appropriate rule by appropriate components with appropriate functions. Hence, this generalized formulation of computational functionalism still presupposes that the brain is subject to a specific kind of mechanistic explanation, which can be studied empirically by neuroscience. Given this generalization, computational functionalism is compatible with any computational theory of mind, including connectionist computationalism.

But abandoning the strong analogy between minds and computers (based on program execution), as the generalized version of computational functionalism does, produces a loss of explanatory power. The generalized version of computational functionalism still appeals to computation in explaining mental capacities, but it can no longer appeal to the flexibility that comes with the ability to acquire, store, modify, and execute different programs. Which computing mechanisms are powerful enough to explain mental capacities? We do not have room here to enter this complex debate (Macdonald and Macdonald 1995, Aizawa 2003). But by drawing attention to all functional and organizational aspects of computing mechanisms at all relevant levels, the account here proposed promises to push this debate forward.

The present account sheds light on some other old disputes too. Two mental capacities that are especially contentious are intentionality and consciousness. Several thought experiments have been proposed to show that either intentionality or consciousness cannot be explained by program execution (e.g., Block 1978, Searle 1980, Maudlin 1989). The failure of explanation by program execution is then assumed to affect functionalism, presumably due to the assumption that functionalism entails computationalism. But now we have seen that properly construed, functionalism does not entail computationalism.

The only legitimate conclusion that may be drawn from these thought experiments is that computationalism is insufficient to explain intentionality or consciousness. This does not entail that computationalism explains no mental capacities, nor does it entail that intentionality and consciousness cannot be explained functionally by some process other than program execution. In other words, even if the intuitions behind those thought experiments are accepted, computationalism might explain many mental capacities, and functionalism might be true of the whole mind. Of course, the intuitions behind the thought experiments are themselves in dispute. They remain an unlikely basis for reaching consensus on these matters.[30]

## 7 Functionalism, Computationalism, and Computational Functionalism

I have discussed three theses:

> *Functionalism*: The mind is the functional organization of the brain.
>
> *Computationalism*: The functional organization of the brain is computational.
>
> *Computational Functionalism* (generalized): The mind is the computational organization of the brain.

Computational functionalism is the conjunction of functionalism and computationalism. I have offered a mechanistic framework within which to make sense of these doctrines and exhibit some of their mutual relations.

Functionalism does not entail computationalism, and by now it should be easy to see why. Functional organizations are specified by mechanistic explanations, and there are many mechanistic explanations that do not involve program execution or any other computational process. That the mind is subject to mechanistic explanation is consistent with any non-computational mechanistic explanation applying to the mind. Thus, it is a fallacy to attempt to refute functionalism by impugning some computationalist hypothesis or another (as done, e.g., by Churchland 2005).

Computationalism does not entail functionalism either. Computationalism is compatible with the mind being the computational organization of the brain, but also with the mind being some non-computational but still functional property of the brain, or even some non-functional property of the brain, such as its physical composition, the speed of its action, its color, or more plausibly, the intentional content or phenomenal qualities of its states. In short, one may be a computationalist while opposing or being neutral about functionalism, at least with respect to some aspects of the mind.

Computationalism is an empirical hypothesis about the particular kind of mechanistic explanation that applies to the brain. Even if the brain is a computing mechanism, the mind may or may not be the brain's computational organization—perhaps there are aspects of the mind that have to do with other properties, e.g., the phenomenal qualities of mental states. But if brains turn out *not* to be computing mechanisms, then computationalism (and hence computational functionalism) is false. So, regardless of whether one agrees with computational functionalism, one can still focus on whether the brain is a computing mechanism and investigate computationalism. This, of course, cannot be done by examining intuitions about imaginary scenarios (Block 1978, Searle 1980, Maudlin 1989)—it can only be done by studying the functional organization of the brain empirically.[31]

The standard formulations of computational functionalism in philosophy of mind have made it difficult to discuss computationalism as productively as it can be. They have convinced many philosophers that computationalism is an a priori thesis, to be discussed by philosophical arguments and thought experiments, and to be judged by the extent to which it solves philosophical problems such as the mind-body problem. This has led philosophers to ignore the fact that, in so far as it has empirical content, computationalism embodies an empirical scientific hypothesis about the functional organization of the brain, which comes in several varieties that ought to be assessed by neuroscientists on grounds that are largely empirical.

## References

Adams, F. and K. Aizawa (2001). "The Bounds of Cognition." Philosophical Psychology **14**(43-64).

Aizawa, K. (2003). The Systematicity Arguments. Boston, Kluwer.

Allen, C., M. Bekoff, et al., Eds. (1998). Nature's Purposes: Analysis of Function and Design in Biology. Cambridge, MA, MIT Press.

Armstrong, D. M. (1970). The Nature of Mind. The Mind/Brain Identity Thesis. C. V. Borst. London, Macmillan: 67-79.

Ariew, A., R. Cummins, et al., Eds. (2002). Functions: New Essays in the Philosophy of Psychology and Biology. Oxford, Oxford University Press.

Baum, E. B. (2004). What is Thought? Cambridge, MA, MIT Press.

Bechtel, W. (2001). Cognitive Neuroscience: Relating Neural Mechanisms and Cognition. Theory and Method in the Neurosciences. P. Machamer, R. Grush and P. McLaughlin. Pittsburgh, PA, University of Pittsburgh Press: 81-111.

Bechtel, W. (2006). Discovering Cell Mechanisms: The Creation of Modern Cell Biology. New York, Cambridge University Press.

Bechtel, W. and A. Abrahamsen (2005). "Explanation: A Mechanistic Alternative." Studies in History and Philosophy of Biological and Biomedical Sciences **36**(2): 421-441.

Bechtel, W. and J. Mundale (1999). "Multiple Realizability Revisited: Linking Cognitive and Neural States." Philosophy of Science **66**: 175-207.

Bechtel, W. and R. C. Richardson (1993). Discovering Complexity: Decomposition and Localization as Scientific Research Strategies. Princeton, Princeton University Press.

Bickle, J. (1998). Psychoneural Reduction: The New Wave. Cambridge, MA, MIT Press.

Block, N. (1978). Troubles with Functionalism. Perception and Cognition: Issues in the Foundations of Psychology. C. W. Savage. Minneapolis, University of Minnesota Press. **6:** 261-325.

Block, N. (1980). Introduction: What is Functionalism? Readings in Philosophy of Psychology. N. Block. London, Methuen. **1:** 171-184.

Block, N. (1995). "The Mind as the Software of the Brain." In *An Invitation to Cognitive Science*, edited by D. Osherson, L. Gleitman, S. Kosslyn, E. Smith and S. Sternberg, MIT Press.

Block, N. (2003). "Do Causal Powers Drain Away?" Philosophy and Phenomenological Research **67**(1): 133-150.

Block, N. and J. A. Fodor (1972). "What Psychological States Are Not." Philosophical Review **81**(2): 159-181.

Bogen, J. (2005). "Regularities and Causality; Generalizations and Causal Explanations." Studies in History and Philosophy of Biological and Biomedical Sciences **36**(2): 397-420.

Bontly, T. (1998). "Individualism and the Nature of Syntactic States." British Journal for the Philosophy of Science **49**: 557-574.

Boorse, C. (2002). A Rebuttal on Functions. Functions: New Essays in the Philosophy of Psychology and Biology. A. Ariew, R. Cummins and M. Perlman. Oxford, Oxford University Press**:** 63-112.

Boyd, R. N. (1980). Materialism without Reductionism: What Physicalism Does Not Entail. Readings in the Philosophy of Psychology. N. Block. London, Methuen**:** 67-106.

Buller, D. J., Ed. (1999). Function, Selection, and Design. Albany, State University of New York Press.

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

Chalmers, D. J. (1996b). The Conscious Mind: In Search of a Fundamental Theory. Oxford, Oxford University Press.

Chalmers, D. J. (unpublished). “A Computational Foundation for the Study of Cognition,” available at http://consc.net/papers/computation.html. References here are to the paper as downloaded on 12/15/2006.

Chrisley, R. L. (1995). "Why Everything Doesn't Realize Every Computation." Minds and Machines **4**: 403-430.

Christensen, W. D. and M. H. Bickhard (2002). "The Process Dynamics of Normative Function." The Monist **85**(1): 3-28.

Churchland, P. M. (2005). "Functionalism at Forty: A Critical Retrospective." The Journal of Philosophy: 33-50.

Churchland, P. M. and P. S. Churchland (1982). Functionalism, Qualia, and Intentionality. Mind, Brain, and Function: Essays in the Philosophy of Mind. J. I. B. a. R. W. Shahan. Norman, University of Oklahoma Press**:** 121-145.

Churchland, P. S. and T. J. Sejnowski (1992). The Computational Brain. Cambridge, MA, MIT Press.

Copeland, B. J. (1996). "What is Computation?" Synthese **108**: 224-359.

Copeland, B. J. (2000). "Narrow versus Wide Mechanism: Including a Re-Examination of Turing's Views on the Mind-Machine Issue." The Journal of Philosophy **XCVI**(1): 5-32.

Corcoran, J., W. Frank, and M. Maloney (1974). "String Theory." The Journal of Symbolic Logic **39**(4): 625-637.

Craver, C. (2001). "Role Functions, Mechanisms, and Hierarchy." Philosophy of Science **68**(March 2001): 53-74.

Craver, C. (2005). "Beyond Reductionism: Mechanisms, Multifield Integration and the Unity of Neuroscience." Studies in History and Philosophy of Biological and Biomedical Sciences **36**(2): 373-395.

Craver, C. F. (2006). "When Mechanistic Models Explain." Synthese.

Craver, C. F. (forthcoming). Explaining the Brain. Oxford, Oxford University Press.

Craver, C. and L. Darden (2001). Discovering Mechanisms in Neurobiology. Theory and Method in the Neurosciences. P. Machamer, R. Grush and P. McLaughlin. Pittsburgh, PA, University of Pittsburgh Press**:** 112-137.

Cummins, R. (1977). "Programs in the Explanation of Behavior." Philosophy of Science **44**: 269-287.

Cummins, R. (1983). The Nature of Psychological Explanation. Cambridge, MA, MIT Press.

Cummins. R. (2002). “Neo-teleology.” In Ariew, A., R. Cummins, et al., Eds. (2002). Functions: New Essays in the Philosophy of Psychology and Biology. Oxford, Oxford University Press.

Cummins, R. and G. Schwarz (1991). Connectionism, Computation, and Cognition. Connectionism and the Philosophy of Mind. T. Horgan and J. Tienson. Dordrecht, Kluwer**:** 60-73.

Darden, L. (2006). Reasoning in Biological Discoveries. New York, Cambridge University Press.

de Ridder, J. (2006). "Mechanistic Artefact Explanation." *Studies in History and Philosophy of Science* **37**(1): 81-96.

Dennett, D. C. (1978). Brainstorms. Cambridge, MA, MIT Press.

Devitt, M. and K. Sterelny (1999). Language and Reality: An Introduction to the Philosophy of Language. Cambridge, MA, MIT Press.

Egan, F. (1992). "Individualism, Computation, and Perceptual Content." Mind **101**(403): 443-459.

Egan, F. (2003). Naturalistic Inquiry: Where does Mental Representation Fit in? Chomsky and His Critics. L. M. Antony and N. Hornstein. Malden, MA, Blackwell**:** 89-104.

Enç, B. (1983). "In Defense of the Identity Theory." Journal of Philosophy **80**: 279-298.

Fodor, J. A. (1965). Explanations in Psychology. Philosophy in America. M. Black. London, Routledge and Kegan Paul.

Fodor, J. A. (1968a). Psychological Explanation. New York, Random House.

Fodor, J. A. (1968b). "The Appeal to Tacit Knowledge in Psychological Explanation." Journal of Philosophy **65**: 627-640.

Fodor, J. A. (1975). The Language of Thought. Cambridge, MA, Harvard University Press.

Fodor, J. A. (1997). Special Sciences: Still Autonomous after All These Years. Ridgeview, CA.

Fodor, J. A. (2000). The Mind Doesn't Work That Way. MIT Press, Cambridge, MA.

Gillett, C. (2002). "The Dimensions of Realization: A Critique of the Standard View." Analysis **62**: 316-323.

Gillett, C. (2003). "The Metaphysics of Realization, Multiple Realizability and the Special Sciences." The Journal of Philosophy **C**(11): 591-603.

Glennan, S. S. (2002). "Rethinking Mechanistic Explanation." *Philosophy of Science* **64**: 605-206.

Glennan, S. (2005). "Modeling Mechanisms." Studies in History and Philosophy of Biological and Biomedical Sciences **36**(2): 443-464.

Harman, G. (1973). Thought. Princeton, Princeton University Press.

Harman, G. (1999). Reasoning, Meaning and Mind. Oxford, Clarendon Press.

Haugeland, J. (1978). "The Nature and Plausibility of Cognitivism." Behavioral and Brain Sciences **2**: 215-260.

Heil, J. (2003). From an Ontological Point of View. Oxford, Clarendon Press.

Heil, J. (2004). Functionalism, Realism and Levels of Being. Hilary Putnam: Pragmatism and Realism. J. Conant and U. M. Zeglen. London, Routledge**:** 128-142.

Houkes, W. (2006). "Knowledge of Artefact Functions." *Studies in History and Philosophy of Science* **37**(1): 102-113.

Houkes, W. and A. Meijers (2006). "The Ontology of Artefacts: The Hard Problem." *Studies in History and Philosophy of Science* **37**(1): 118-131.

Houkes, W. and P. Vermaas (2004). "Actions versus Functions: A Plea for an Alternative Metaphysics of Artifacts." *The Monist* **87**(1): 52-71.

Humphreys, P. (2004).

Keeley, B. (2000). "Shocking Lessons from Electric Fish: The Theory and Practice of Multiple Realizability." Philosophy of Science **67**: 444-465.

Kim, J. (1989). "The Myth of Nonreductive Materialism." Proceedings and Addresses of the American Philosophical Association **63**: 31-47.

Kim, J. (1992). "Multiple Realization and the Metaphysics of Reduction." Philosophy and Phenomenological Research **52**: 1-26.

Kim, J. (1998). Mind in a Physical World: An Essay on the Mind-Body Problem and Mental Causation. Cambridge, MA, MIT Press.

Kim, J. (2003). "Blocking Causal Drainage and Other Maintenance Chores with Mental Causation." Philosophy and Phenomenological Research **67**(1): 151-176.

Lewis, D. K. (1966). "An Argument for the Identity Theory." Journal of Philosophy **63**: 17-25.

Lewis, D. K. (1969). "Review of *Art, Mind, and Religion*." Journal of Philosophy **66**(22-27).

Lewis, D. K. (1972). "Psychophysical and Theoretical Identifications." Australasian Journal of Philosophy **50**: 249-258.

Lewis, D. K. (1980). Mad Pain and Martian Pain. Readings in Philosophy of Psychology, Volume 1. N. Block. Cambridge, MA, MIT Press**:** 216-222.

Lucas, J. R. (1996). "Minds, Machines, and Gödel: A Retrospect." *Machines and Thought: The Legacy of Alan Turing*. P. J. R. Millikan and A. Clark, Eds. Oxford, Clarendon.

Lycan, W. (1981). "Form, Function, and Feel." Journal of Philosophy **78**: 24-50.

Lycan, W. (1982). Psychological Laws. Mind, Brain, and Function: Essays in the Philosophy of Mind. J. I. Biro and R. W. Shahan. Norman, University of Oklahoma Press**:** 9-38.

Lycan, W. (1987). Consciousness. Cambridge, MA, MIT Press.

Macdonald, C. and G. Macdonald, Eds. (1995). Connectionism: Debates on Psychological Explanation, Volume Two. Oxford, Blackwell.

MacDonald and Macdonald 1995

Machamer, P. (2004). "Activities and Causation: The Metaphysics and Epistemology of Mechanisms." International Studies in the Philosophy of Science **18**(1): 27-39.

Machamer, P. K., L. Darden, and C. Craver (2000). "Thinking About Mechanisms." Philosophy of Science **67**: 1-25.

Marr, D. (1982). Vision. New York, Freeman.

Maudlin, T. (1989). "Computation and Consciousness." Journal of Philosophy **86**(8): 407-432.

Millikan, R. G. (1984). Language, Thought, and Other Biological Categories: New Foundations for Realism. Cambridge, MA, MIT Press.

Moor, J. H. (1978). "Three Myths of Computer Science." British Journal for the Philosophy of Science **29**: 213-222.

Nelson, R. J. (1987). "Church's Thesis and Cognitive Science." Notre Dame Journal of Formal Logic **28**(4): 581-614.

Newell, A. (1980). "Physical Symbol Systems." Cognitive Science **4**: 135-183.

Newell, A. (1990). Unified Theories of Cognition. Cambridge, MA, Harvard University Press.

Pereboom, D. and H. Kornblith (1991). "The Metaphysics of Irreducibility." Philosophical Studies **63**.

Perlman, M. (2004). "The Modern Philosophical Resurrection of Teleology." *The Monist* **87**(1): 3-51.

Piccinini, G. (2003a). "Alan Turing and the Mathematical Objection." Minds and Machines **13**(1): 23-48.

Piccinini, G. (2003b). "Review of John von Neumann's The Computer and the Brain." Minds and Machines **13**(2): 327-332.

Piccinini, G. (2003c). "Epistemic Divergence and the Publicity of Scientific Methods." Studies in the History and Philosophy of Science **34**(3): 597-612.

Piccinini, G. (2004a). "The First Computational Theory of Mind and Brain: A Close Look at McCulloch and Pitts's 'Logical Calculus of Ideas Immanent in Nervous Activity'." *Synthese* **141**(2): 175-215.

Piccinini, G. (2004b). "Functionalism, Computationalism, and Mental States." *Studies in the History and Philosophy of Science* **35**(4): 811-833.

Piccinini, G. (2004c). "Functionalism, Computationalism, and Mental Contents." *Canadian Journal of Philosophy* **34**(3): 375-410.

Piccinini, G. (2006). "Computation without Representation." *Philosophical Studies*.

Piccinini, G. (2007a). "Is Everything a Turing Machine, and Does It Matter to the Philosophy of Mind?" *Australasian Journal of Philosophy*.

Piccinini, G. (2007b). Computational Explanation and Mechanistic Explanation of Mind. Cartographies of the Mind: Philosophy and Psychology in Intersection. M. De Caro, F. Ferretti and M. Marraffa. Dordrecht, Springer**:** 23-36.

Piccinini, G. (forthcoming). "Computationalism, the Church-Turing Thesis, and the Church-Turing Fallacy." *Synthese*.

Preston, B. (1998). "Why is a Wing Like a Spoon? A Pluralist Theory of Function." The Journal of Philosophy XCV(5): 215-254.

Preston, B. (2003). "Of Marigold Beer: A Reply to Vermaas and Houkes." British Journal for the Philosophy of Science **54**: 601-612.

Prinz, J. (2001). Functionalism, Dualism and the Neural Correlates of Consciousness. Philosophy and the Neurosciences: A Reader. W. Bechtel, P. Mandik, J. Mundale and R. Stufflebeam. Oxford, Blackwell.

Polger, T. W. (2004a). Natural Minds. Cambridge, MA, MIT Press.

Putnam, H. (1960). Minds and Machines. Dimensions of Mind: A Symposium. S. Hook. New York, Collier: 138-164.

Putnam, H. (1967a). The Mental Life of Some Machines. Intentionality, Minds, and Perception. H. Castañeda. Detroit, Wayne State University Press: 177-200.

Putnam, H. (1967b). Psychological Predicates. Art, Philosophy, and Religion. Pittsburgh, PA, University of Pittsburgh Press.

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

Roth, M. (2005). "Program Execution in Connectionist Networks." Mind and Language **20**(4): 448-467.

Rupert, R. (2004). "Challenges to the Hypothesis of Extended Cognition." The Journal of Philosophy **CI**: 389-428.

Rupert, R. (2006). "Functionalism, Mental Causation, and the Problem of Metaphysically Necessary Effects." Noûs **40**: 256-283.

Scheele, M. (2006). "Function and Use of Artefacts: Social Conditions of Function Ascription." *Studies in History and Philosophy of Science* **37**(1): 23-36.

Scheutz, M. (2001). "Causal versus Computational Complexity." Minds and Machines **11**: 534-566.

Scheutz, M. (2004). Comments presented at the 2004 Pacific APA in Pasadena, CA.

Schlosser, G. (1998). "Self-re-Production and Functionality: A Systems-Theoretical Approach to Teleological Explanation." Synthese **116**(3): 303-354.

Searle, J. R. (1980). "Minds, Brains, and Programs." The Behavioral and Brain Sciences **3**: 417-457.

Searle, J. R. (1992). The Rediscovery of the Mind. Cambridge, MA, MIT Press.

Sellars, W. (1954). "Some Reflections on Language Games." Philosophy of Science **21**: 204-228.

Shagrir, O. (1998). "Multiple Realization, Computation and the Taxonomy of Psychological States." Synthese **114**: 445-461.

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

Shagrir, O. (2005). The Rise and Fall of Computational Functionalism. Hilary Putnam. Y. Ben-Menahem. Cambridge, Cambridge University Press.

Shagrir, O. (2006). "What is Computing in the Brain?" Synthese.

Shapiro, L. A. (1994). "Behavior, ISO Functionalism, and Psychology." Studies in the History and Philosophy of Science **25**(2): 191-209.

Shapiro, L. A. (2000). "Multiple Realizations." The Journal of Philosophy **XCVII**(12): 635-654.

Schlosser, G. (1998). "Self-re-Production and Functionality: A Systems-Theoretical Approach to Teleological Explanation." Synthese **116**(3): 303-354.

Schroeder, T. (2004). "Functions from Regulation." *The Monist* **87**(1): 115-135.

Shoemaker, S. (2001). Realization and Mental Causation. Physicalism and Its Discontents. C. Gillett and B. Loewer. Cambridge, Cambridge University Press: 74-98.

Shoemaker, S. (2003a). "Realization, Micro-Realization, and Coincidence." Philosophy and Phenomenological Research **LXVII**(1): 1-23.

Shoemaker, S. (2003b). Identity, Cause and Mind, Expanded Edition. Oxford: Clarendon Press.

Simon, H. A. (1996). The Sciences of the Artificial, Third Edition. Cambridge, MA, MIT Press.

Smith, B. C. (1996). On the Origin of Objects. Cambridge, MA, MIT Press.

Sober, E. (1990). Putting the Function Back into Functionalism. Mind and Cognition. W. Lycan. Malden, MA, Blackwell: 63-70.

Sober, E. (1999). "The Multiple Realizability Argument against Reductionism." Philosophy of Science **66**: 542-564.

Stich, S. (1983). From Folk Psychology to Cognitive Science. Cambridge, MA, MIT Press.

Tabery, J. (2004). "Synthesizing Activities and Interactions in the Concept of a Mechanism." Philosophy of Science **71**(1): 1-15.

Thagard, P. (2003). "Pathways to Biomedical Discovery." Philosophy of Science **70**(2): 235-254.

Turing, A. M. (1950). "Computing Machinery and Intelligence." Mind **59**: 433-460.

Vermaas, P. E. (2006). "The Physical Connection: Engineering Function Ascription to Technical Artefacts and their Components." *Studies in History and Philosophy of Science* **37**(1): 62-75.

Vermaas, P. E. and W. Houkes (2006). "Technical Functions: A Drawbridge between the Intentional and Structural Natures of Technical Artefacts." *Studies in History and Philosophy of Science* **37**(1): 5-18.

von Neumann, J. (1951). The General and Logical Theory of Automata. Cerebral Mechanisms in Behavior. L. A. Jeffress. New York, Wiley**:** 1-41.

von Neumann, J. (1958). The Computer and the Brain. New Haven, Yale University Press.

Webb, J. C. (1980). Mechanism, Mentalism, and Metamathematics. Dordrecht, Reidel.

Wilkes, K. V. (1982). Functionalism, Psychology, and the Philosophy of Mind. Mind, Brain, and Function: Essays in the Philosophy of Mind. J. I. Biro and R. W. Shahan. Norman, University of Oklahoma Press**:** 147-167.

Wilson, M. (1985). "What is This Thing Called "Pain"?-The Philosophy of Science Behind the Contemporary Debate." Pacific Philosophical Quarterly **66**: 227-267.

Wilson, M. (1993). Honorable Intensions. Naturalism: A Critical Appraisal. S. J. Wagner and R. Warner. Notre Dame, Indiana, University of Indiana Press**:** 53-94.

Wilson, R. A. (2004). Boundaries of the Mind: The Individual in the Fragile Sciences. Cambridge, Cambridge University Press.

Wimsatt, W. C. (1972). "Teleology and the Logical Structure of Function Statements." Studies in History and Philosophy of Science **3**(1): 1-80.

Wright, L. (1973). "Functions." Philosophical Review **82**: 139-168.

[1] A version of this paper was presented at the 2004 Pacific APA in Pasadena, CA. I thank the audience and commentators, Matthias Scheutz and Charles Wallis, for their helpful feedback. I also thank those who commented on previous versions of this paper, especially David Chalmers, Robert Cummins, Carl Craver, Chris Eliasmith, Peter Machamer, Diego Marconi, Andrea Scarantino, Oron Shagrir, and Julie Zahle. I am especially indebted to Bill Lycan for discussion and comments. In revising the paper, I benefited from attending the 2006 NEH Summer Seminar in Mind and Metaphysics and from a University of Missouri Research Grant. The views expressed here do not necessarily reflect those of these institutions.

[2] In this paper, I am mostly concerned with functionalism with respect to scientific theories of mind—what Block (1980) calls psychofunctionalism. I am not directly concerned with functionalism about folk psychological theories (Lewis 1966, 1972, 1980; Armstrong 1970), analytical or conceptual truths about the mental (Shoemaker 2003b), or the *content* of mental states (e.g., Sellars 1954; Harman 1973, 1999; Block 1986). I avoid formulating functionalism in terms of Ramsey sentences (Lewis 1966, Block 1980) because such formulations obscure the issues addressed here (cf. Gillett 2007). I am also not concerned with several other topics related to functionalism: to what extent functionalism is consistent with reductionism and the identity theory (e.g., Lewis 1969, Fodor 1975, 1997, Boyd 1980, Churchland and Churchland 1982, Lycan 1982, Enç 1983, Wilson 1984, 1993, Kim 1989, 1992, Pereboom and Kornblith 1991, Bickle 1998, Shagrir 1998, Sober 1999, Bechtel and Mundale 1999, Keeley 2000, Bechtel 2001, Prinz 2001, Pereboom 2002), whether functionalism is consistent with mental causation (Block 2003, Kim 2003, Rupert 2006), whether functionalism should be formulated in terms of roles or realizers, whether functionalism should be formulated in terms of higher level properties, higher order properties, or similarities between fundamental properties (Heil 2003, 2004), whether the mind

extends into the environment (Harman 1999, Shapiro 1994, Adams and Aizawa 2001, Wilson 2004, Rupert 2004), and the correct metaphysics of realization (Kim 1998, Shapiro 2000, Shoemaker 2001, 2003a, Gillett 2002, 2003, Polger 2004, Wilson 2004).

[3] The second formulation, though perhaps more common in the literature than the first, is not quite equivalent to it. There is more to functional organization than individual functional states and their relations. There are also aggregates of states, components bearing the states, functional properties of the components, and relations between the components. Since the first formulation is more general than the second, I prefer the first, but nothing in what follows hinges on the difference between the two.

[4] The critical literature is quite large. Representative examples include Block 1978, Putnam 1988, Searle 1992, and Lucas 1996.

[5] For a more detailed reconstruction and discussion of Putnam's functionalism and computational functionalism, see Piccinini 2004b and Shagrir 2005.

[6] As McCulloch put it, "What we thought we were doing (and I think we succeeded pretty well) was treating the brain as a Turing machine" (quoted in von Neumann 1951, 33). For a detailed study of McCulloch and Pitts's theory, see Piccinini 2004a.

[7] Examples of such arguments may be found in Dennett 1978, Webb 1980, Nelson 1987, Chalmers 1996b, Simon 1996, and Baum 2004. For their refutation, see Copeland 2000 and Piccinini forthcoming.

[8] David Chalmers has pointed out to me that the weak thesis may be strengthened by arguing that while computation is insufficient for the instantiation of most properties, computation is sufficient for the instantiation of mental properties. Unlike most properties, mental properties might be such that they are instantiated "in virtue of the implementation of computations" (Chalmers, personal correspondence). This is a fair point, but it makes a difference only insofar as we have good evidence that computation is sufficient for mentation.

Chalmers defends a thesis of computational sufficiency along these lines in an insightful paper (Chalmers unpublished). Briefly, Chalmers defines a notion of abstract causal organization, which involves "the patterns of interaction among the parts of the system, abstracted away from the make-up of individual parts and from the way the causal connections are implemented," and yet includes "a level fine enough to determine the causation of behavior" (ibid.). Chalmers argues that unlike most non-mental properties, all there is to mental properties is abstract causal organization, and abstract causal organization can be fully and explanatorily captured computationally. If Chalmers is right, then (the right kind of) computation is sufficient for mentation while being insufficient for most other properties.

Lacking space for a detailed discussion of Chalmers's argument, let me make the following brief comment. I don't see that Chalmers's argument establishes computational sufficiency for mental properties in a way that makes a difference for present purposes. Chalmers faces a dilemma. If abstract causal organization is truly fine grained enough to determine the causation of a system's behavior, then—contrary to Chalmers's intent—abstract causal organization will capture (at least the causal aspects of) *any* property (including digestion, combustion, etc.). If, instead, abstract causal organization excludes enough information about a system to rule out at least certain properties (such as digestion and combustion), then—again, contrary to Chalmers's intent—there is no reason to accept that abstract causal organization will capture every aspect of mental properties. Either way, the specific connection between mentation and computation is not strengthened. Thus, Chalmers's argument does not affect our main discussion.

[9] I am using 'digit' to refer to the physical entities or states manipulated by computers, regardless of whether they represent numbers.

[10] According to Smith, "The *only* compelling reason to suppose that we (or minds or intelligence) might be computers stems from the fact that we, too, deal with representations, symbols, meanings, and the like" (1996, 11). Smith is exaggerating in calling this the *only* reason, but the semantic analogy between minds and computers does have a long and influential history. For a more detailed discussion, see Piccinini 2004c.

[11] An argument to this effect is in Fodor 1968b, which is one of the founding documents of computational functionalism and which Fodor 2000 singles out as conflating functionalism and computationalism. An argument along similar lines is in Newell 1990, 113ff. Other influential authors offered similar considerations. For the role played by program execution in Alan Turing's thinking about intelligence, see Turing 1950 and Piccinini 2003a. For the role played by program execution in von Neumann's thinking about brains, see von Neumann 1958 and Piccinini 2003b.

[12] Here is a case in point:

> [A] programming language can be thought of as establishing a mapping of the physical states of a machine onto sentences of English such that the English sentence assigned to a given state expresses the instruction the machine is said to be executing when it is in that state (Fodor 1968b, 638).

Beginning in the 1970s, some authors attempted to go beyond the mapping view by imposing further constraints on implementation. Most prominently, Bill Lycan (1987) imposed a teleological constraint. Although this was a step in the right direction (more on this later), Lycan and others used 'software/hardware' and 'role/realizer' interchangeably. They offered no account specific to *software* implementation as opposed to role realization, and the conflation between functionalism and computationalism remained unaffected. When talking specifically about computation, philosophers continued to appeal to versions of the mapping view:

> [A] physical system is a computational system just in case there is an appropriate (revealing) *mapping between the system's physical states and the elements of the function computed* (Churchland and Sejnowski 1992, p. 62; emphasis added).
>
> [C]omputational theories construe cognitive processes as formal operations defined over symbol structures… Symbols are just functionally characterized objects whose individuation conditions are specified by *a realization function $f_g$ which maps equivalence classes of physical features of a system to what we might call "symbolic" features*. Formal operations are just those physical operations that are differentially sensitive to the aspects of symbolic expressions that under the realization function $f_g$ are specified as symbolic features. The mapping $f_g$ allows a causal sequence of physical state transitions to be interpreted as a *computation* (Egan 1992, p. 446; first emphasis added).

[13] Matthias Scheutz has suggested an amendment to the standard explication of software implementation, according to which for a computational description to be considered relevant to software implementation, *all* its states and *all* its computational steps must map onto the system that is being described (Scheutz 2004). This proposal rules out many computational descriptions, such as computational models that employ C++ programs, as irrelevant to what software is being implemented by a system, and hence it improves on the standard view. But this proposal still leaves in place indefinitely many computational descriptions of any given system, and hence it doesn't solve the present problem.

[14] The thesis that everything has computational descriptions is more problematic than it may appear, in a way that adds a further difficulty for the standard view of software implementation. For ordinary computational descriptions are only approximate descriptions rather than exact ones, and hence a further undesirable consequence of the standard view is that programs can only approximate the behavior of the systems that are supposed to be implementing them. A full treatment of this further difficulty would take up more space than is available in this paper, so I will set it aside. For a detailed discussion of the relevance of approximation to the claim that everything is computational and the way this claim trivializes computationalism, see Piccinini 2007a.

[15] For a systematic treatment of computational modeling in science, see Humphreys 2004.

[16] In the relevant sense of "program execution". Another notion is that of developmental "program," which is employed in biology. That is an independent notion of program, which would require a separate account.

[17] Cf.: "the paradigmatic psychological theory is a list of instructions for producing behavior" (Fodor 1968b, 630). For a more extended discussion, see Piccinini 2004b.

[18] E.g., see Bechtel and Richardson 1993, Machamer, Darden and Craver 2000, Bechtel 2001, 2006, Craver 2001, 2005, 2006, forthcoming, Glennan 2002, 2005, Thagard 2003, Machamer 2004, Tabery 2004, Bogen 2005, Bechtel and Abrahamsen 2005, Darden 2006.

[19] Polger 2004 follows a similar strategy in distinguishing different versions of functionalism based on which notion of function they employ.

[20] The main competing accounts of function ascription in biology and engineering can be found in Allen, Bekoff, and Lauder 1998; Preston 1998; Schlosser 1998; Buller 1999; Ariew, Cummins, and Perlman 2002, Christensen and Bickhard 2002. Other contributions include Perlman 2004, Hourkes and Vermaas 2004, Cameron 2004, Johansson 2004, Schroeder 2004, Vermaas and Houkes 2006, Scheele 2006, Franssen 2006, Vermaas 2006, Houkes 2006, Houkes and Meijers 2006, Kroes 2006.

[21] Why not? First, the evolutionary account of functions does not immediately apply to artifacts, such as computers, which are not the product of evolution by natural selection. Because of this, it's unclear whether and how computational functionalism, which is based on an analogy between minds and computers, can be formulated in terms of a notion of function that relies on evolution. (This is not to say that there can't be a broader notion of selection that applies to both organisms and artifacts; cf. Wright 1973, Preston 2003.) Second, and perhaps most importantly, the evolutionary account of functions grounds any resulting theory of mind on the notions and practices of evolutionary biology rather than the empirical disciplines that are relevant to explaining the capacities of minds and computers—namely, psychology, neuroscience, and computer science.

[22] For example, non-etiological accounts of teleology are given by Schlosser 1998 and Boorse 2002.

[23] Carl Gillett (2007) has independently developed a proposal similar to mechanistic functionalism to deal with some of the more metaphysical aspects of functionalism.

When mechanistic functionalism is further specified by employing teleological functions, the resulting doctrine is a close relative of teleological functionalism (Lycan 1981, 1987, Wilkes 1982, Millikan 1984, Sober 1990, Shapiro 1994, Rupert 2006). According to teleological functionalism, the mind is the teleological organization of the brain, or mental states are individuated by their teleological function. A teleological version of mechanistic functionalism adds to traditional teleological functionalism a mechanistic framework within which to specify the functional organization of the brain. Furthermore, the following two caveats apply. First, teleological functionalism is often offered as an alternative to computational functionalism (e.g., Lycan 1981, Millikan 1984, Sober 1990). But I will soon argue that mechanistic explanation is actually the most adequate framework within which to explicate computational explanation. As a consequence, computational functionalism turns out to be a special version of mechanistic functionalism. Second, many supporters of teleological functionalism endorse an etiological account of teleology. But as I already pointed out, this may not be the most suitable account of teleology for present purposes.

[24] Unfortunately, the distinction between the abstract and the concrete is an endless source of confusion, especially in the philosophy of computation. Here I have no room for a full treatment of all the relevant issues, so the following brief point will have to suffice. It is common to see references to different levels of description of computing mechanisms, some of which are said to be more abstract than others (e.g., Newell 1980, Marr 1982). By focusing on the mechanistic explanation of computing mechanisms, I am not questioning the distinction between more abstract and more concrete levels of description and I am not focusing on the "implementation" or "physical" level at the expense of more "abstract" computational levels. Rather, I am offering a new way of understanding the computational levels, in light of the fact that insofar as levels of description are relevant to the explanation of a system's capacities, they are all describing aspects of a mechanism—they are all part of a complete mechanistic explanation of the system, regardless of how abstract they are. For a detailed account of levels within mechanistic explanation, see Craver forthcoming.

[25] For the mathematical theory of strings, see Corcoran, Frank, and Maloney 1974.

[26] It is possible for two different computations to generate the same output from the same input. This simply shows that computations are individuated more finely than input-output mappings.

[27] Which strings are relevant? All the strings from the relevant alphabet. For each computing mechanism, there is a relevant finite alphabet. Notice that the rule need not define an output for all input strings (and perhaps internal states) from the relevant alphabet. If some outputs are left undefined, then under those conditions the mechanism should produce no output strings of the relevant type.

[28] Cf. any standard textbook on computer organization and design, such as Patterson and Hennessy 1998.

[29] For an exception, see Moor 1978, 215. Robert Cummins, one of the few people to discuss this issue explicitly, maintained that "programs aren't causes but abstract objects or play-by-play accounts" (Cummins 1983, p. 34; see also Cummins 1977). This weaker notion of program execution is quite popular among philosophers, and is yet another side of the fuzziness surrounding functionalism and computationalism. This is because the weaker notion does not motivate the strong analogy between minds and computers that is behind the slogan "the mind is the software of the brain," and yet the weaker notion is often used in explicating computationalism or even functionalism. The main reason for Cummins's view of program execution seems to be the way he mixes functional analysis and computational description. Roughly, Cummins thinks that explaining a capacity by program execution is the same as giving a functional analysis of it, and therefore the program is not a part of the computer but a description of it (see section 4 above). This leads Cummins and his followers to the paradoxical conclusion that connectionist networks compute by executing algorithms or programs (Cummins and Schwarz 1991, Roth 2005). But it should be obvious that connectionist networks do not store and execute programs in the sense I explicated in the main text, which is why their behavior is not as flexible as that of digital computers. I should point out that Cummins has recently agreed that "stored programs are certainly causes" (personal correspondence).

[30] For an argument that this use of intuitions is inconclusive, see Piccinini 2003c.

[31] For some hints on the likely outcome, cf. Piccinini 2007b.

# Searle on Brains as Computers

**William J. Rapaport**

**Department of Computer Science and Engineering,**
**Department of Philosophy, Department of Linguistics**
**and Center for Cognitive Science**
**State University of New York at Buffalo, Buffalo, NY 14260-2000**
`rapaport@cse.buffalo.edu`
`http://www.cse.buffalo.edu/∼rapaport/`

January 28, 2007

**Abstract**

Is the brain a digital computer? Searle says that this is meaningless; I say that it is an empirical question. Is the mind a computer program? Searle says no; I say: properly understood, yes. Can the operations of the brain be simulated on a digital computer? Searle says: trivially yes; I say yes, but that it is not trivial.

## 1 Three Questions

In his 1990 essay, "Is the Brain a Digital Computer?", Searle factors the "slogan … 'the mind is to the brain as the program is to the hardware' " (p. 21) into three questions:

1. Is the brain a digital computer?
2. Is the mind a computer program?
3. Can the operations of the brain be simulated on a digital computer? (Searle 1990: 21.)

Let us consider each of these, beginning with the second.

## 2 Is the Mind a Computer Program?

What does it mean to say that the mind is a computer program? Surely not that there is a programming language and a program written in it that is being executed on a brain—not for humans, at least. But it could mean that by bottom-up, reverse engineering (neuroscience) together with top-down, cognitive-scientific investigation, we could write a program that would cause a computer to exhibit mental behavior. However, that's question 3, to which Searle gives a different answer.

Possibly question 2 means that the mind plays the same role with respect to the brain that a program does to a computer; call this "Good Old-Fashioned Cartesian Dualism". This may not be much progress over the "slogan" of which question 2 is supposed to be merely a part, but it is worth a small digression.

### 2.1 Good Old-Fashioned Cartesian Dualism

Computational cognitive science, including what John Haugeland (1985: 112) has termed "good old-fashioned artificial intelligence", is, I believe, good old-fashioned Cartesian dualism. The view that mental states and processes are (or are expressible as) algorithms that are *implemented in* the physical states and processes of physical devices is (a form of) Cartesian dualism: The mental states and processes and the physical states and processes can be thought of as different "substances" that "interact". How might this be?

It should be clear that an algorithm and a computer are different kinds of "substance". If one considers an algorithm as a mathematical abstraction (in the ordinary sense of the term 'abstraction'), then it is an abstract mathematical entity (like numbers, sets, etc.). Alternatively, if one considers an algorithm as a text expressed in some language, then it is, say, ink marks on paper or ASCII characters in a word-processor's file. An algorithm might even be—and indeed ultimately is—"switch settings" (or their electronic counterparts) in a computer. All of these are very different sorts of things from a very physical computer.

How do an algorithm and a computer "interact"? By the latter being a semantic interpretation—a model—of the former. More precisely, the *processes* of the brain/body/computer are semantic interpretations (or models) of the mind/algorithm in the sense of semantics as correspondence. But this is just what we call an implementation. So, an implementation is a kind of semantic interpretation. (For further discussion of this, see Rapaport 1999, 2005.)

Note, by the way, that the mind/algorithm can itself be viewed as a semantic interpretation of the brain/body/computer, since the correspondence can go both ways (Rapaport 2002). How is a mind implemented? Consider a computer program: Ultimately, the program (as text) is implemented as states of a computer (expressed in binary states of certain of its components). That is purely physical, but it is *also* purely syntactic; hence, *it* can have a semantic interpretation. An abstract data type, for instance, can be thought of as the semantic interpretation of an arrangement of bits in the computer (cf. Tenenbaum & Augenstein 1981: 1, 6, 45; see Rapaport 1995, §2.3, and Rapaport 2005). This Janus-faced aspect of the bit arrangements—thought of both as a physical model or implementation of the abstract algorithm and as a syntactic domain interpretable by the algorithm and its data structures—is Marx W. Wartofsky's "model muddle" (Wartofsky 1966, 1979: xiii–xxvi; Rapaport 1995; 1996, Ch. 2; 1999; 2000).

Now, is this really good old-fashioned Cartesian dualism? Is mind-body interaction really semantic interpretation or implementation? Or might this semantic/implementational view be more like some other theory of the mind?

It is not parallelism, since there really is a *causal* interaction: The algorithm (better: the process) *causes* the physical device to behave in certain ways.

So it's not epiphenomenalism, either. Moreover, the device—or its behavior—can produce changes in the program, as in the case of self-modifying programs, or even in the case of a system competent in natural language whose knowledge base (part of the software) changes with each interaction.

Could it be a dual-aspect theory? Perhaps: Certainly, the physical states and processes are one "level of description", and the mental states and processes are another "level of description" *of the same (physical) system*. But talk of levels of description seems to me to be less illuminating than the theory of semantics as correspondence. More to the point, neither "level" is a complete description of the system: The algorithm is not the process, nor can one infer from the algorithm what the future behavior of the process will be: The process can behave in ways not predictable by the programmer (cf. Fetzer 1988, 1991). And even a complete physical description of the system would not tell us *what* it is doing; this is one of the lessons of functionalism.

So dualism is at least plausible. Do the physical states and processes produce mental ones? Here is where the problem of qualia—i.e., subjective qualitative experiences, including pain and physical sensations—enters. (I say more about this in Rapaport 2005, §2.2.)

## 2.2 Return to Searle

Does question 2 mean that the mind is the *way* the brain behaves? That seems right, but isn't the right analogy: It *doesn't* seem right to say that a program is the way a computer behaves.

"Programs," Searle goes on to say, "are defined purely formally or syntactically" (p. 21). That, I think, is not *quite* right: They require a set of input-output conventions, which would be "links" to the world. In any case, this together with the assertion that "minds have an intrinsic content ... immediately [implies] that the program by itself cannot constitute the mind" (p. 21). What does 'content' mean?

If it means something internal to the mind (a "container" metaphor; cf. Twardowski 1894, Rapaport 1978), then that minds have intrinsic content could mean that within a mind there are links among mental concepts, some of which play the role of a language of thought and others of which play the role of mental representations of external perceptions (cf. Rapaport 1996, Ch. 3; 2000; 2002; 2006). If so, that would be purely syntactic, as Searle says programs are.

If, on the other hand, 'content' means a relation to an external entity, then why don't programs have that, too (as I noted two paragraphs back)? In any case, programs do take input from the external world: I enter '2' on the keyboard, which results (after a few transductions) in a switch being set in the computer, which the program interprets as the number 2.

So, on either interpretation, the conclusion doesn't follow, since programs can *also* have "intrinsic mental content", whatever that means.

The problem is that question 2 is not the right question. Of course "The formal syntax of the program does not by itself guarantee the presence of mental contents" (p. 26), because the program might never be executed. What Searle should have asked is whether the mind is a computer *process*. And here the answer can be 'yes', since *processes* can have contents. Searle says:

> I showed this [viz., that the formal syntax of a program doesn't guarantee the presence of mental contents] a decade ago in the Chinese Room Argument .... The argument rests on the simple logical truth that syntax is not the same as, nor is it by itself sufficient for, semantics. (Searle 1990: 21.)

This seems to follow from Charles Morris's definitions (1938) of syntax as the study of relations *among* symbols and of semantics as the study of relations *between* symbols and their meanings; thus, syntax $\neq$ semantics. Nor is it the case that semantics can be "derived", "constructed", or "produced" from syntax by Morris's definitions. But the first-person semantic enterprise *is* one of determining correspondences among symbols—between linguistic symbols and internal representations of external objects. Hence, it *is* syntactic even on Morris's definition. The *third*-person semantic enterprise is more like what Morris had in mind. But one person's third-person semantics is another's first-person semantics: If one cognitive agent, Oscar, tries to account for the semantics of another cognitive agent (Cassie) by drawing correspondences between her mental concepts and things in the world, all he can really do is draw correspondences between his representations of her concepts and his representations of things in the world. As with the turtles who hold up the Earth, it's syntax all the way down. (For more about how Cassie and Oscar understand each other, see, e.g., Rapaport 2003.)

## 3 Can the Operations of the Brain Be Simulated on a Digital Computer?

Let's turn to question 3, the answer to which Searle thinks is trivially—or, at least, uninterestingly—affirmative. "[N]aturally interpreted, the question means: Is there some description of the brain such that under that description you could do a computational simulation of the operations of the brain" (p. 21). Such a description would inevitably be partial (Smith 1985). Hence, so would be the computational simulation. But if it passed the Turing test (i.e., if its effects in the actual world were indistinguishable from those of a human), then what's not in the model is an implementation detail. What might these be? They might include sensations of pain, warm fuzzy feelings associated with categorizing something as "beautiful", etc. (cf. Rapaport 2005). As for pain, don't forget that our *sensation* of it is an internal perception, just like our sensation of an odor (cf. Crane 1998). It might be possible to be in pain and to know that one is in pain without what we normally call a pain sensation, just as it is possible to determine the presence of an object by its odor—by a chemical analysis—without sensing that odor.[1] The "triviality" or "obviousness" of the answer to question 3 stems, according to Searle, from Church's Thesis: "The operations of the brain can be simulated on a digital computer in the same sense in which weather systems, the behavior of the New York Stock market or the pattern of airline flights over Latin America can" (p. 21). And, presumably, since simulated weather isn't weather, simulated brains aren't brains. But the premise is arguable (Rapaport 2005, §3); at least, it does not follow that the behavior of simulated brains isn't *mental*. Brains and brain behavior are special cases.

## 4 Is the Brain a Digital Computer?

Searle equates question 1 to "Are brain processes computational?" (p. 22). What would it mean to say that the brain was *not* a digital computer? It might mean that the brain is *more* than a digital computer—that only some proper *part* of it *is* a digital computer. What would the rest of it be? Implementation details, perhaps. I am, however, willing to

[1] Angier 1992 reports that "Sperm cells possess the same sort of odor receptors that allow the nose to smell." This does not mean, of course, that sperm cells have the mental capacity to have smell-qualia. Blakeslee 1993 reports that "humans ... may exude ... odorless chemicals called pheromones that send meaningful signals to other humans." She calls this "a cryptic sensory system that exists without conscious awareness ...." And Fountain 2006 discusses a plant that has what might be called a sense of smell, presumably without any smell qualia.

admit that perhaps not all of the brain's processes are computational. Following Philip N. Johnson-Laird (1988: 26–27), I take the task of cognitive science to be to find out *how much* of the brain's processes *are* computational—and surely some of them are (Rapaport 1998). It is, thus, a working hypothesis that brain processes are computational, requiring an empirical answer and not subject to apriori refutation.

On the other hand, to say that the brain is not a digital computer might mean that it's a different kind of entity altogether—that no part of it is a digital computer. But that seems wrong, since it *can* execute programs (we use our brains to hand-simulate computer programs, which, incidentally, is the inverse of Turing's 1936 analysis of computation).

What are brain processes, how do they differ from mental processes, and how do both of these relate to computer processes? A computer process is a program being executed; therefore, it is a physical thing that implements an abstract program. A brain process is also a physical thing, so it might correspond to a computer process. A mental process could be either (i) something abstract yet dynamic or (ii) a brain process. The former (i) makes no sense if programs and minds are viewed as static entities. The latter (ii) would mean that *some* brain processes are mental (others, like raising one's arm, are not). So to ask if brain processes are computational is like asking if a computer process is computational. That question means: Is the current behavior of the computer describable by a recursive function (or is it just a fuse blowing)? So Searle's question 1 is: Is the current (mental) behavior of the brain describable by a recursive function? This is the fundamental question of artificial intelligence as computational philosophy. It is a major research program, not a logical puzzle capable of apriori resolution.

Searle's categorization of the possible positions into "strong AI" ("all there is to having a mind is having a program"), "weak AI" ("brain processes (and mental processes) can be simulated computationally"), and "Cognitivism" ("the brain is a digital computer") is too coarse (p. 22). What about the claim that a computer running the "final" AI program (the one that passes the Turing test, let's say) has mentality? As I argued above, that's not necessarily "just" having a program. But on the *process* interpretation of question 2, Strong AI could be the view that all there is to having a mind is having a *process*, and that's more than having a program. What about the claim that the "final" AI program need not be the one that humans use—i.e., the claim that computational *philosophy* might "succeed", not computational *psychology*? (Computational philosophy seeks to learn which aspects of cognition in general are computable; computational psychology studies *human* cognition using computational techniques. Cf. Shapiro 1992, Rapaport 2003.) This is a distinction that Searle does not seem to make. Finally, Pylyshyn's version of "cognitivism" (1985) does not claim that the brain *is* a digital computer, but that mental processes are computational processes. That seems to me to be compatible with the brain being "more" than a digital computer.

Searle complains that multiple realizability is "disastrous" (p. 26; cf. Rapaport 2005, §4.1). The first reason is that *anything* can be described in terms of 0s and 1s (p. 26), and there might be *lots* of 0-1 encodings of the brain. But the real question, it seems to me, is this: Does the brain *compute* (effectively) some function? What is the input-output description of that function? The answer to the latter question is whatever psychology tells us is intelligence, cognition, etc. For special cases, it's easier to be a bit more specific: For natural-language understanding, the input is some utterance of natural language, and the output is an "appropriate" response (where the measure of "appropriateness" is defined, let's say, sociolinguistically). For vision, the input is some physical object, and the output is, again, some "appropriate" response (say, an utterance identifying the object or some scene, or some behavior to pick up or avoid the object, etc.). Moreover, these two modules (natural-language understanding and vision) must be able to "communicate" with each other. (They might or might not be modular in Fodor's sense (1983), or cognitively impenetrable in Pylyshyn's sense (1985). In any case, solving one of these problems will require a solution to the other; they are "AI-complete" (Shapiro 1992).)

The second allegedly disastrous consequence of multiple realizability is that "syntax is not intrinsic to physics. The ascription of syntactical properties is always relative to an agent or observer who treats certain physical phenomena as syntactical" (p. 26). The observer assigns 0s and 1s to the physical phenomena. But Morris's definition of syntax as relations among symbols (uninterpreted marks) can be extended to relations among components of any system. Surely, physical objects stand in those relationships "intrinsically". And if 0s and 1s *can* be ascribed to a physical object (by an observer), that *fact* exists independently of the agent who *discovers* it.

Searle's claim "that syntax is essentially an observer relative notion" (p. 27) is very odd. One would have expected him to say that about *semantics*, not syntax. Insofar as one can look at a complex system and describe (or discover) relations among its parts (independently of any claims about what it does at any higher level), one is doing *non*–observer-relative syntax. Searle says that "this move is no help. A physical state of a system is a computational state only relative to the assignment to that state of some computational role, function, or interpretation" (p. 27), where, presumably, the assignment is made by an observer. But an assignment is an assignment of meaning; it's an interpretation. So is Searle saying that computation is fundamentally a *semantic* notion? But, for Church, Turing, et al., computation is purely *syntactic*. It's only the input-output coding that *might* constitute an assignment. But such coding is only needed in order to be able to link the syntax with the standard theory of computation in terms of functions from natural numbers to natural numbers. If we're willing to express the theory of computation in terms of functions from physical states to physical states (and why shouldn't we?), then it's not relative.

Searle rejects question 1: "There is no way you could discover that something is intrinsically a digital computer because the characterization of it as a digital computer is always relative to an observer who assigns a *syntactical* interpretation to the purely physical features of the system" (p. 28, my italics). I, too, reject question 1, but for a very different reason: I think the question is really whether *mental processes* are computational. In any event, suppose we *do* find computer programs that exhibit intelligent input-output behavior, i.e., that pass the Turing Test. Computational *philosophy* makes no claim about whether that tells us that the human *brain* is a digital computer. It only tells us that intelligence is a computable function. So at best Searle's arguments are against computational *psychology*. But even that need not imply that the brain *is* a digital computer, only that it behaves as if it were. To discover that something $X$ is intrinsically a digital computer, or a $Y$, is to have an abstraction $Y$, and to find correspondences between $X$ and $Y$.

Perhaps what Searle is saying is that being computational is not a *natural* kind, but an artifactual kind (cf. Churchland & Sejnowski 1992):

> I am not saying there are *a priori* limits on the patterns we could discover in nature. We could no doubt discover a pattern of events in my brain that was isomorphic to the implementation of the vi program on this computer. (Searle 1990: 28.)

This is to admit what I observed two paragraphs back. Searle continues:

> But to say that something is *functioning as* a computational process is to say something more than that a pattern of physical events is occurring. It requires the assignment of a computational interpretation by some agent. (Searle 1990: 28.)

But why? Possibly because to find correspondences between two things (say, a brain and the Abstraction ComputationalProcess—better, the Abstraction Computer) is observer-relative? But if we have *already* established that a certain brain process is an implementation of `vi`, what *extra* "assignment of a computational interpretation by some agent" is needed?

Searle persists:

> Analogously, we might discover in nature objects which had the same sort of shape as chairs and which could therefore be used as chairs; but we could not discover objects in nature which were functioning as chairs, except relative to some agent who regarded them or used them as chairs. (Searle 1990: 2.)

The analogy is clearly with *artifacts*. But the notion of a computational process does not seem to me to be artifactual; it is *mathematical*. So the proper analogy would be something like this: Can we discover in nature objects that were, say, sets, or numbers, or Abelian groups? Here, the answer is, I think, (a qualified) 'yes'. (It is qualified, because sets and numbers are abstract and infinite, while the world is concrete and finite. Groups may be a clearer case.) In any event, is Searle claiming that the implementation of `vi` in my brain isn't `vi` until someone *uses it as* `vi`? If there is an implementation of `vi` on my Macintosh that no one ever uses, it's still `vi`.

Searle accuses computational cognitive scientists of "commit[ing] the homunculus fallacy ... treat[ing] the brain as if there were some agent inside it using it to compute with" (p. 28). But consider Patrick Hayes's (1990, 1997) objection to the Chinese-Room Argument: Computation is a series of switch-settings; it isn't rule-following. (On this view, by the way, the solar system *does* compute certain mathematical functions.) Turing machines do *not* follow rules; they simply change state. There are, however, descriptions—programs—of the state changes, and anything that follows (executes) that program computes the same function computed by the Turing machine. A *universal* Turing machine can also follow that program. But the original, special-purpose Turing machine's program is "hardwired" (an analogy, of course, since everything is abstract here). A universal Turing machine has *its* program similarly hardwired. It is only when the universal Turing machine is fed a program that it follows the rules of that program. But that's what *we* do when *we* consciously follow (hand-simulate) the rules of a program. So it's *Searle* who commits the homuncular fallacy in the Chinese-Room Argument by putting a person in the room. It is not the person in the room who either does or does not understand Chinese; it is the entire system (Rapaport 2000, 2006). Similarly, it is not some *part* of my brain that understands language; it is *I* who understands.

In his discussion of "discharging" the homunculus, Searle says that "All of the higher levels reduce to this bottom level. Only the bottom level really exists; the top levels are all just *as-if*" (p. 29). But *all* levels exist, and *all* levels "do the same thing", albeit in different ways (Rapaport 1990, 2005).

I noted above that systems that don't follow rules can still be said to be computing. My example was the solar system. Searle offers "nails [that] compute the distance they are to travel in the board from the impact of the hammer and the density of the wood" (p. 29) and the human visual system; "neither," according to him, "compute anything" (p. 29). But in fact they both do. (The nail example might not be ideal, but it's a nice example of an *analog* computation.)

> But you do not *understand* hammering by supposing that nails are somehow intrinsically implementing hammering algorithms and you do not *understand* vision by supposing the system is implementing, e.g., the shape from shading algorithm. (Searle 1990: 29; my italics.)

Why not? It gives us a theory about how the system might be performing the task. We can falsify (or test) the theory. What more could *any* (scientific) theory give us? What further kind of understanding could there be? Well, there could be first-person understanding, but I doubt that we could ever know what it is like to be a nail or a solar system. We *do* understand what it is like to be a cognitive agent!

The problem, I think, is that Searle and I are interested in different (but complementary) things:

> ... you cannot explain a physical system such as a typewriter or a brain by identifying a pattern which it shares with its computational simulation, because the existence of the pattern does not explain how the system actually works *as a physical system*. (Searle 1990: 32.)

Of course not. That would be to confuse the implementation with the Abstraction. Searle is interested in the former; he wants to know *how the (human) brain works*. I, however, want to know *what the brain does* and how *anything* could do it. For that, I need an account at the functional/computational level, not a biological (or neuroscientific) theory.

> The mistake is to suppose that in the sense in which computers are used to process information, brains also process information. [Cf. Johnson 1990.] To see that that is a mistake, contrast what goes on in the computer with what goes on in the brain. In the case of the computer, an outside agent encodes some information in a form that can be processed by the circuitry of the computer. That is, he or she provides a syntactical realization of the information that the computer can implement in, for example, different voltage levels. The computer then goes through a series of electrical stages that the outside agent can interpret both syntactically and semantically even though, of course, the hardware has no intrinsic syntax or semantics: It is all in the eye of the beholder. And the physics does not matter provided only that you can get it to implement the algorithm. Finally, an output is produced in the form of physical phenomena which an observer can interpret as symbols with a syntax and a semantics.
>
> But now contrast this with the brain. ... none of the relevant neurobiological processes are observer relative ... and the specificity of the neurophysiology matters desperately. (Searle 1990: 34.)

There is much to disagree with here. First, "an outside agent" need *not* "encode ... information in a form that can be processed by the circuitry of the computer". A computer could be (and typically is) designed to take input directly from the real world and to perform the encoding (better: the transduction) itself, as, e.g., in document-image understanding (cf. Srihari & Rapaport 1989, 1990; Srihari 1991, 1993, 1994). Conversely, abstract concepts are "encoded" in natural language so as to be processable by *human* "circuitry".

Second, although I find the phrase 'syntactical realization' quite congenial (cf. Rapaport 1995), I'm not sure how to parse the rest of the sentence in which it appears. What does the computer "implement in voltage levels": the information? the syntactical realization? I'd say the former, and that the syntactical realization *is* the voltage levels. So there's an issue here of whether the voltage levels are *interpreted as* information, or vice versa.

Third, the output need not be physical phenomena interpreted by an observer as symbols. The output *could* be an action, or more internal data (e.g., as in a vision system),[2] or even natural language to be interpreted by another

[2] Searle seems to think (p. 34) that vision systems yield sentences as output! (See below.)

*computer*. Indeed, the latter suggests an interesting research project: Set up Cassie and Oscar, two computational cognitive agents implemented in a knowledge-representation, reasoning, and acting system such as SNePS.[3] Let Cassie have a story pre-stored or as the result of "reading" or "conversing". Then let her tell the story to Oscar and ask him questions about it. No *humans* need be involved.

Fourth, neurobiological processes aren't observer-relative, only because we don't care to, or need to, describe them that way. The computer works as it does independently of us, too. Of course, for *us* to understand what the brain is doing—from a third-person point of view—we need a psychological level of description (cf. Chomsky 1968, Fodor 1968, Enç 1982).

Finally, why should "the specificity of the neurophysiology matter desperately"? Does this mean that if the neurophysiology were different, it wouldn't be a human brain? I suppose so, but that's relevant only for the implementation side of the issue, not the Abstraction side, with which I am concerned.

Here is another example of how Searle does not seem to understand what computational cognitive science is about:

> A standard computational model of vision will take in information about the visual array on my retina and eventually print out the sentence, "There is a car coming toward me". But that is not what happens in the actual biology. In the biology a concrete and specific series of electro-chemical reactions are set up by the assault of the photons on the photo receptor cells of my retina, and this entire process eventually results in a concrete visual experience. The biological reality is not that of a bunch of words or symbols being produced by the visual system, rather it is a matter of a concrete specific conscious visual event; this very visual experience. (Searle 1990: 34–35.)

The first sentence is astounding. First, why does he assume that the input to the computational vision system is *information on the retina*, rather than *things in the world*? The former is close to an *internal* symbol representing external information! Second, it is hardly "standard" to have a vision system yield a *sentence* as an output. It might, of course ("Oh, what a pretty red flower."), but, in the case of a car coming at the system, an aversive maneuver would seem to be called for, not a matter-of-fact description. Nonetheless, precisely that input-output interaction *could*, *pace* Searle, be "what happens in the actual biology": I could say that sentence upon appropriate retinal stimulation.

Of course, as the rest of the quotation makes clear, Searle is more concerned with the intervening qualitative experience, which, he seems to think, humans have but computers don't (or can't). Well, could they? Surely, there ought to be an intervening stage in which the retinal image is processed (perhaps stored) before the information thus processed or stored is passed to the natural-language module and interpreted and generated. Does that process have a qualitative feel? Who knows? *How* would you know? Indeed, how do I know (or believe) that *you* have such a qualitative feel? The question is the same for both human and computer. Stuart C. Shapiro has suggested how a pain-feeling computer could be built (Rapaport 2005, §2.3.1); similarly, it's possible that a physical theory of sensation could be constructed. Would it be computational? Perhaps not—but so what? Perhaps some "mental" phenomena are not *really* mental (or computational) after all (Rapaport 2005, §2.3). Or perhaps a computational theory will always be such that there is a role to play for some sensation or other, even though the actual sensation in the event is not computational. That is, every computational theory of pain or vision or what have you will be such that it will refer to a sensation without specifying what the sensation is. (Cf. Gracia's (1990) example of a non-written universal for a written text, discussed in Rapaport 2005, §2.2. See also McDermott 2001.)

Of course, despite my comments about the linguistic output of a vision system, the sentence that Searle talks about could be a "sentence" of one's language of thought. That, however, would fall under the category of being a "concrete specific conscious visual event" and "not ... a bunch of words or symbols" (Cf. Pylyshyn 1981; Srihari op. cit.; Srihari & Rapaport op. cit.)

Searle's final point about question 1 is this:

> The point is not that the claim "The brain is a digital computer" is false. Rather it does not get up to the level of falsehood. It does not have a clear sense. (Searle 1990: 35.)

This is because "you could not *discover* that the brain *or anything else* was intrinsically a digital computer" (p. 35, my italics). "Or anything else"? Even an IBM PC? Surely not. Possibly he means something like this: Suppose we find an alien physical object and theorize that it is a digital computer. Have we *discovered* that it is? No—we've got an *interpretation* of it *as* a digital computer (cf. "you could assign a computational interpretation to it as you could to

[3] Shapiro 1979, 2000; Shapiro & Rapaport 1987, 1992, 1995; Shapiro et al. 2006. Further information is available online at: [http://www.cse.buffalo.edu/sneps] and at: [http://www.cse.buffalo.edu/∼rapaport/snepskrra.html].

anything else" (p. 35)). But how else *could* we "discover" anything about it? Surely, we could discover that it's made of silicon and has $10^k$ parts. But that's consistent with his views about *artifacts*. Could we *discover* the topological arrangement of its parts? I'd say 'yes'. Can we *discover* the sequential arrangement of its behaviors? Again, I'd say 'yes'. Now consider this: How do we determine that it's made of silicon? By subjecting it to certain physical or chemical tests and having a theory that says that any substance that behaves thus and so is (made of) silicon. But if anything that *behaves* such and thus is a computer, then so is this machine! So we *can* discover that (or whether) it is a computer. (Better: We can discover whether its processing is computational.)

# References

Angier, Natalie (1992), "Odor Receptors Discovered in Sperm Cells", *The New York Times* (30 January 1992): A19.

Blakeslee, Sandra (1993), "Human Nose May Hold An Additional Organ For a Real Sixth Sense", *The New York Times* (7 September 1993): C3.

Chomsky, Noam (1968), *Language and Mind* (New York: Harcourt, Brace & World).

Churchland, Patricia S., & Sejnowski, Terrence J. (1992), *The Computational Brain* (Cambridge, MA: MIT Press).

Crane, Tim (1998), "Intentionality as the Mark of the Mental", in Anthony O'Hear (ed.), *Current Issues in the Philosophy of Mind* (Cambridge, UK: Cambridge University Press): 229–251.

Enç, Berent (1982), "Intentional States of Mechanical Devices", *Mind* 91(362; April): 161–182.

Fetzer, James H. (1988), "Program Verification: The Very Idea", *Communications of the ACM* 31: 1048–1063.

Fetzer, James H. (1991), "Philosophical Aspects of Program Verification", *Minds and Machines* 1: 197–216.

Fodor, Jerry A. (1968), *Psychological Explanation: An Introduction to the Philosophy of Psychology* (New York: Random House).

Fodor, Jerry A. (1983), *The Modularity of Mind: An Essay in Faculty Psychology* (Cambridge, MA: MIT Press).

Fountain, Henry (2006), "This Plant Has the Sense of Smell (Loves Tomatoes, Hates Wheat)", *The New York Times* (3 October 2006): F1.

Gracia, Jorge J.E. (1990), "Texts and Their Interpretation", *Review of Metaphysics* 43: 495–542.

Haugeland, John (1985), *Artificial Intelligence: The Very Idea* (Cambridge, MA: MIT Press).

Hayes, Patrick J. (1990), "Searle's Chinese Room", abstract of presentation to the Society for Philosophy and Psychology Symposium on Searle's Chinese Room and Workshop on Symbol Grounding, University of Maryland (electronic mail posting from Stevan Harnad, 6 June 1990).

Hayes, Patrick J. (1997), "What Is a Computer? An Electronic Discussion", *The Monist* 80(3).

Johnson-Laird, Philip N. (1988), *The Computer and the Mind: An Introduction to Cognitive Science* (Cambridge, MA: Harvard University Press).

McDermott, Drew (2001), *Mind and Mechanism* (Cambridge, MA: MIT Press).

Morris, Charles (1938), *Foundations of the Theory of Signs* (Chicago: University of Chicago Press).

Pylyshyn, Zenon (1981), "The Imagery Debate: Analog Media versus Tacit Knowledge", in Ned Block (ed.), *Imagery* (Cambridge, MA: MIT Press): 151–206.

Pylyshyn, Zenon (1985), *Computation and Cognition: Toward a Foundation for Cognitive Science, 2nd edition* (Cambridge, MA: MIT Press).

Rapaport, William J. (1978), "Meinongian Theories and a Russellian Paradox", *Noûs* 12: 153–180; errata, *Noûs* 13 (1979) 125.

Rapaport, William J. (1990), "Computer Processes and Virtual Persons: Comments on Cole's 'Artificial Intelligence and Personal Identity'," *Technical Report 90-13* (Buffalo: SUNY Buffalo Department of Computer Science, May 1990) [http://www.cse.buffalo.edu/∼rapaport/Papers/cole.tr.17my90.pdf].

Rapaport, William J. (1995), "Understanding Understanding: Syntactic Semantics and Computational Cognition", in James E. Tomberlin (ed.), *Philosophical Perspectives, Vol. 9: AI, Connectionism, and Philosophical Psychology* (Atascadero, CA: Ridgeview): 49–88; reprinted in Toribio, Josefa, & Clark, Andy (eds.) (1998), *Language and Meaning in Cognitive Science: Cognitive Issues and Semantic Theory*, *Artificial Intelligence and Cognitive Science: Conceptual Issues*, Vol. 4 (New York: Garland): 73–88.

Rapaport, William J. (1996), "Understanding Understanding: Semantics, Computation, and Cognition", *Technical Report 96-26* (Buffalo: SUNY Buffalo Department of Computer Science).

Rapaport, William J. (1998), "How Minds Can Be Computational Systems", *Journal of Experimental and Theoretical Artificial Intelligence* 10: 403–419.

Rapaport, William J. (1999), "Implementation is Semantic Interpretation", *The Monist* 82: 109–130.

Rapaport, William J. (2000), "How to Pass a Turing Test: Syntactic Semantics, Natural-Language Understanding, and First-Person Cognition", *Journal of Logic, Language, and Information* 9(4): 467–490; reprinted in James H. Moor (ed.), *The Turing Test: The Elusive Standard of Artificial Intelligence* (Dordrecht: Kluwer, 2003): 161–184.

Rapaport, William J. (2002), "Holism, Conceptual-Role Semantics, and Syntactic Semantics", *Minds and Machines* 12(1): 3–59.

Rapaport, William J. (2003), "What Did You Mean by That? Misunderstanding, Negotiation, and Syntactic Semantics", *Minds and*

*Machines* 13(3): 397–427.

Rapaport, William J. (2005), "Implementation Is Semantic Interpretation: Further Thoughts", *Journal of Experimental and Theoretical Artificial Intelligence* 17(4; December): 385–417.

Rapaport, William J. (2006), "How Helen Keller Used Syntactic Semantics to Escape from a Chinese Room" [http://www.cse.buffalo.edu/∼rapaport/Papers/helenkeller.pdf].

Searle, John R. (1990), "Is the Brain a Digital Computer?", *Proceedings and Addresses of the American Philosophical Association*, 64(3): 21–37.

Shapiro, Stuart C. (1979), "The SNePS Semantic Network Processing System", in Nicholas Findler (ed.), *Associative Networks: Representation and Use of Knowledge by Computers* (New York: Academic Press): 179–203.

Shapiro, Stuart C. (1992), "Artificial Intelligence", in Stuart C. Shapiro (ed.), *Encyclopedia of Artificial Intelligence, second edition* (New York: John Wiley & Sons): 54–57.

Shapiro, Stuart C. (2000), "SNePS: A Logic for Natural Language Understanding and Commonsense Reasoning", in Łucja M. Iwańska & Stuart C. Shapiro (eds.), *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language* (Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press): 175–195.

Shapiro, Stuart C., & Rapaport, William J. (1987), "SNePS Considered as a Fully Intensional Propositional Semantic Network", in Nick Cercone & Gordon McCalla (eds.), *The Knowledge Frontier: Essays in the Representation of Knowledge* (New York: Springer-Verlag): 262–315; shorter version appeared in *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86, Philadelphia)* (Los Altos, CA: Morgan Kaufmann): 278–283; a revised shorter version appears as "A Fully Intensional Propositional Semantic Network", in Leslie Burkholder (ed.), *Philosophy and the Computer* (Boulder, CO: Westview Press, 1992): 75–91.

tem Shapiro, Stuart C., & Rapaport, William J. (1992), "The SNePS Family", *Computers and Mathematics with Applications* 23: 243–275; reprinted in F. Lehmann (ed.), *Semantic Networks in Artificial Intelligence* (Oxford: Pergamon Press, 1992): 243–275.

Shapiro, Stuart C., & Rapaport, William J. (1995), "An Introduction to a Computational Reader of Narratives", in Judith F. Duchan, Gail A. Bruder, & Lynne E. Hewitt (eds.), *Deixis in Narrative: A Cognitive Science Perspective* (Hillsdale, NJ: Lawrence Erlbaum Associates): 79–105.

Shapiro, Stuart C., & the SNePS Research Group (2006), "SNePS" [http://en.wikipedia.org/wiki/SNePS].

Smith, Brian Cantwell (1985), "Limits of Correctness in Computers", reprinted in Charles Dunlop & Rob Kling (eds.), *Computerization and Controversy* (San Diego: Academic Press, 1991): 632–646.

Srihari, Rohini K. (1991), "PICTION: A System that Uses Captions to Label Human Faces in Newspaper Photographs", *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91, Anaheim)* (Menlo Park, CA: AAAI Press/MIT Press): 80–85.

Srihari, Rohini K. (1993), "Intelligent Document Understanding: Understanding Photos with Captions", *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR-93, Tsukuba City, Japan).*

Srihari, Rohini K. (1994), "Use of Collateral Text in Understanding Photos in Documents", *Proceedings of the Conference on Applied Imagery and Pattern Recognition (AIPR/SPIE, Washington, DC).*

Srihari, Rohini K., & Rapaport, William J. (1989), "Extracting Visual Information From Text: Using Captions to Label Human Faces in Newspaper Photographs", *Proceedings of the 11th Annual Conference of the Cognitive Science Society (Ann Arbor, MI)* (Hillsdale, NJ: Lawrence Erlbaum Associates): 364–371.

Srihari, Rohini K., & Rapaport, William J. (1990), "Combining Linguistic and Pictorial Information: Using Captions to Interpret Newspaper Photographs", in Deepak Kumar (ed.), *Current Trends in SNePS—Semantic Network Processing System*, (Berlin: Springer-Verlag Lecture Notes in Artificial Intelligence 437): 85–96.

Tenenbaum, Aaron M., & Augenstein, Moshe J. (1981), *Data Structures using Pascal* (Englewood Cliffs, NJ: Prentice-Hall).

Turing, Alan M. (1936), "On Computable Numbers, with an Application to the *Entscheidungsproblem*", *Proceedings of the London Mathematical Society*, Ser. 2, Vol. 42: 230–265; reprinted, with corrections, in Martin Davis (ed.), *The Undecidable: Basic Papers On Undecidable Propositions, Unsolvable Problems And Computable Functions* (New York: Raven Press, 1965): 116–154.

Twardowski, Kasimir (1894), *On the Content and Object of Presentations*, Reinhardt Grossmann (trans.) (The Hague: Nijhoff, 1977).

Wartofsky, Marx W. (1966), "The Model Muddle: Proposals for an Immodest Realism", in *Models: Representation and the Scientific Understanding* (Dordrecht, Holland: D. Reidel, 1979): 1–11.

Wartofsky, Marx W. (1979), "Introduction", in *Models: Representation and the Scientific Understanding* (Dordrecht, Holland: D. Reidel, 1979): xiii–xxvi.