Minds & Machines (2009) 19:529–541
DOI 10.1007/s11023-009-9175-1

Symbol Grounding in Computational Systems: A Paradox of Intentions

Vincent C. Müller

Published online: 2 December 2009

Abstract The paper presents a paradoxical feature of computational systems that suggests that computationalism cannot explain symbol grounding. If the mind is a digital computer, as computationalism claims, then it can be computing either over meaningful symbols or over meaningless symbols. If it is computing over meaningful symbols its functioning presupposes the existence of meaningful symbols in the system, i.e. it implies semantic nativism. If the mind is computing over meaningless symbols, no intentional cognitive processes are available prior to symbol grounding. In this case, no symbol grounding could take place since any grounding presupposes intentional cognitive processes. So, whether computing in the mind is over meaningless or over meaningful symbols, computationalism implies semantic nativism.

Keywords Artificial intelligence · Computationalism · Fodor · Putnam · Semantic nativism · Symbol grounding · Syntactic computation

Computationalism

The computational theory of the mind, or "computationalism" for short, holds that the mind is a digital computer. This view is the basis of much of the information processing model in contemporary cognitive science and artificial intelligence. As Fodor puts it: "The cognitive science that started 50 years or so ago more or less explicitly had as its defining project to examine a theory, largely owing to Turing,

V. C. Müller (✉)
Department of Philosophy & Social Sciences, Anatolia College/ACT,
P.O. Box 21021, 55510 Pylaia, Greece
e-mail: vmueller@act.edu
URL: http://www.typos.de

that cognitive mental processes are operations defined on syntactically structured mental representations that are much like sentences" (Fodor 2000, p. 3f).

To sketch the historical and systematic context, computationalism is closely connected to the view of mental states as physical states with a specific causal *functional role*, as proposed by the earlier Putnam. If the mind is described not at a physical level, but described at the level of these functional roles and if these are taken as realizations of a Turing machine, as computational states, then we have the theory commonly known as *Machine Functionalism*, which includes the stronger thesis of the necessity of computing for mentality: "Mentality, or having a mind, consists in realizing an appropriate Turing machine" (Kim 1996, p. 91; cf. Fodor 1994a, pp. 10–15). As Paul Churchland characterizes it: "What unites them [the cognitive creatures] is that (…) they are all computing the same, or some part of the same abstract ≪sensory input, prior state>, <motor output, subsequent state≫ *function*" (Churchland 2005, p. 333). The computationalist version of functionalism is particularly plausible because computers are necessarily described functionally, as in the notion of a "Turing machine". It does not make sense to describe the mind as a computer in the sense of an identity theory because the *physical description* of a particular computing machine is irrelevant, what matters is the *logical description* of its function, and there could be well be such a description of a brain (and nobody claims that our brain consists of silicon chips like the ones in our PCs).

We shall only discuss computationalism in the sense that computation is sufficient for mental states and that it is the cause of mental states in humans, not in the stronger sense that computation is necessary and sufficient (or only necessary). Computationalism is a stronger thesis, however, than the contention that some or all mental processes can be *modeled* on a digital computer. If a hurricane can be modeled on a computer, this does not mean that the hurricane *is* a computational system (it is doubtful whether such modeling is possible, strictly speaking, since a hurricane is not a discrete state phenomenon). Note, however, that minds might be special cases such that modeling a mind actually *is* producing a mind—given that it has sufficient functional properties (e.g. Chalmers 1996, p. 328).

In a first approximation, *computing* here means the manipulation of symbols following *algorithms*, i.e. explicit non-ambiguous rules that proceed step by step and that can be carried out in finite time, leading to a definite output—what is also called "effective computing". The Church-Turing thesis says that a Turing machine can compute all and only the effectively computable functions.[1]

Computationalism directly implies the possibility of strong Artificial Intelligence: "… computers can think because, in principle, they can be programmed with the same program that constitutes human thought" (Wakefield 2003, p. 286). Or, as Churchland puts it: "The central job of AI research is to create *novel physical*

[1] Copeland (e.g. 1997, 2000, 2002) and others have recently argued that this interpretation of the Church-Turing thesis is mistaken and that there are possible machines, termed "hypercomputers", that could compute functions that are not computable by some Turing machine. For the purposes of this paper, we only need a defining characteristic of "computationalism" and propose to use this standard interpretation of the Church-Turing thesis. Whether the mind is a computer in some different sense is a separate question (and I have tried to undermine the arguments for hypercomputing elsewhere).

realizations of salient parts of, and ultimately all of, the abstract function we are all (more or less) computing" (2005, p. 34).

Computing with Meaningful Symbols: Language of Thought Computationalism

The main theoretical options within computationalism can be divided according to whether the symbols on which the computer operates (and that constitute its program) are meaningful or not. I shall call the option of operating on meaningful symbols "Language of Thought Computationalism", for reasons that shall become evident presently.

The tradition of Fodor's "Language of Thought" focuses on "cognition" or, even more narrowly, "thought", and it claims that thinking is computing over mental representations. Fodor's slogan could be said to be "no computation without representation" (1981, p. 180). Chalmers characterizes this view as follows "… the claim that the computational primitives in a computational description of cognition are also representational primitives. That is to say, the basic syntactic entities between which state-transitions are defined are themselves bearers of semantic content, and are therefore symbols" (1993). Rey sketches one consequence of this approach, namely "the view that propositional attitudes (such as believing, noticing, preferring) are to be regarded as computational relations to semantically valuable representations that are encoded in the brain or other hardware of the thinker" (Rey 2002, p. 203). Smolensky claims that what he calls the "Newell/Simon/Fodor/Pylyshyn view" says that the programs of this computational system "are composed of elements, that is, symbols, referring to essentially the same concepts as the ones used to consciously conceptualize the task domain" (1988, p. 5; cf. 1994).

So, "language of thought computationalism" could be summarized as the conjunction of two views: (1) "*Thinking is computation*" (Fodor 1998, p. 9 [his emphasis]) and: (2) Thinking computes over language-like mental representations.

As Fodor says: "The emphasis upon a syntactical character of thought suggests a view of cognitive processes in general—including, for example, perception, memory and learning—as occurring in a language like medium, a sort of 'language of thought'" (1994b, p. 9). Fodor's emphasis on the syntactical nature of the computational process should not be taken to mean that his position is anything other than language of thought computationalism. It just so happens, that thinking is a computation over symbols that are representations: "First, all mental processes are supposed to be causally sensitive to, and only to, the syntax of the mental representations that they are defined over; in particular, mental processes aren't sensitive to what mental representations mean. This is, I think, at the very heart of the Classical [Fodor's] account of cognition." (Fodor 2005, p. 26)

Given that we have explained the central term of the first conjunct (computing), it remains to specify what we mean that of the second: "language". I will just adopt the proposal by Lycan, who says: "(1) they are composed of parts and syntactically structured; (2) their simplest parts refer or denote things and properties in the world; (3) their meanings as wholes are determined by the semantical properties of their basic parts together with the grammatical rules that have generated the overall syntactic structures; (4) they have truth conditions …; (5) they bear logical relations of entailment or implication to each other" (Lycan 2003, p. 189). What is characteristic for the language of thought is not only that its parts represent, but also that it consists of sentence-like pieces, which, due to their compositionality, have systematicity and productivity, as do natural languages (we can think a virtually unlimited number of thoughts and which thoughts one can think is connected in a systematic way).

Origin of Meaning?

This brings us to the problem. How is it possible that these symbols of a computational system have meaning? Fodor himself appears to see the problem, at least sometimes: "How could a process which, like computation, merely *transforms one symbol into another* guarantee the causal relations between symbols and the world upon which … the meanings of symbols depend?" (1994a, p. 12f). There seem to be two ways in principle: meaning is built-in (innate) or meaning is acquired. What I am trying to show here is that it cannot be acquired, leaving the option of built-in (innate) meaning.

A Short Line

The situation invites a very short line indeed: If language of thought computationalism is the manipulation of meaningful symbols, then the functioning of language of thought (the "cognition" or the "thinking") *presupposes* the existence of meaningful symbols in the system. In other words, the system must have meaningful symbols *before* the language of thought can function. The acquisition of these meaningful symbols can thus not be the work of a language of thought. So, if a newborn child's cognition is within language of thought, then a child must be born with meaningful symbols: language of thought computationalism presupposes meaningful symbols. Fodor, of course, has been supporting the idea of innate meaning for some time, but I do not see much enthusiasm for this view in other computationalists. The "short line" is a simple argument against language of thought computationalism without semantic nativism—an argument we lack so far (see Fodor 2000, 2005; Pinker 2005).

Computing with Meaningless Symbols: Syntactic Computationalism

Given the problem described in the above paradox, it is plausible to revert to a more modest version of computationalism: computation is (or could be) purely syntactic. This does not exclude that some observer could interpret the symbols, but they have no meaning *for the system*. Their meaning plays no role in the computing. At first glance, this is the case with any conventional digital computer: One operation of a set of switches that constitute an OR-gate could be interpreted as be doing a logical operation or as computing an addition. (The logical gates for exclusive or are the same as those for binary addition plus a “carrying over” of surpluses to the next digit). The switches have no meaning for the system itself. When my pocket calculator displays the output “844$” or my washing machine displays “End”, this means something to me, but not to the computer.

Symbol System, Technically

To describe the situation more precisely, it is necessary to gain a deeper understanding of what a computer, really does. The elementary point of a computer is, as we said, that it is a universally programmable algorithmic system. A calculator can “carry out” an algorithm, so mechanical calculators were already constructed in the seventeenth Century by Schickard, Pascal and Leibniz. A computer is programmable, that is which algorithm it carries out can be changed. The universal Turing machine is a model for a computer that can run any program, essentially by giving numbers to all the other simple Turing machines that can compute only one algorithm.

Given the central role of algorithms, we can describe a computer on 3+ levels of description:

Physical Level

Some physical objects: toothed wheels, holes in cards, states of switches, states of transistors, states of neurons, etc. that are causally connected with each other—such that a state of one object can alter the state of another.

Logical Level

The physical objects are taken to be tokens of a type[2] (e.g. charge/no charge) and are manipulated according to algorithms. These algorithms are also stored and changed in the computer via some set of physical tokens (typically the same set). The manipulation follows the algorithms and only concerns these tokens, not their interpretation, it is “purely syntactical”—However, if the compute functions correctly, it recognizes each token as of a type, as a basic symbol for this system, e.g. a 0 or 1 on the basic level of a binary system. On Piccinini’s (2008) terminology, who discusses the problem of how to “individuate computational states”, I do not adhere to a semantic view (since I allow description levels below the “symbolic” level), but neither do I subscribe to his view that that a computational state must be individuated functionally, in terms of function for a whole organism. I tend to think that this would pick out *one* level of description within my “symbolic” level: Piccinini’s explanatory aim is different from mine. (Further discussion in Müller 2008.)

[2] The use of type/token is from Harnad 1990.

Symbolic Level

The physical objects that are manipulated on the logical level are taken to represent; they are (parts of) letters, numbers, words, images, vectors, … One could thus have one algorithm (on the logical level) that carries out several functions (on the symbolic level).

I propose to have "3+" levels rather than "3" because each of the symbols on the symbolic level can symbolize something else in turn. Accordingly, one might distinguish several further levels within the symbolic level when describing a computational cognitive system, for example, the distinction between nonconceptual content and conceptual content, or the distinction between symbols and concepts (for the latter, see Gärdenfors 2000, Chap. 7).

If we now describe a conventional van Neumann machine, e.g. a PC, at its logical level, rather than at its physical (realization) level, we will see basic operations on bits of main memory such as *read* (is this bit on or off?) and *write* (to this bit). These operations will be combined by building in logical (Boolean) switches where one bit takes a particular state, given the state of two other bits. With the help of such switches, one can construct algorithms of switching patterns that perform particular tasks on the symbolic level, e.g. compare, add, … The computing process is a long sequence of such basic operations resulting in a memory state (Cf. Schneider and Werner 2001, Chap. 4.). Note that it is irrelevant for the logical description of the computer whether a particular operation is carried out by the "hardware" or the "software"—one way to see this is to conceive of the computer as a Turing machine (see Davis 2000, p. 167).

Is There Computing Without Meaning?

Could there really be a computing system, however, without any meaningful symbols? One might object that the system must be able to carry out *programs*, programs that are themselves encoded in symbols, and typically stored in memory. Does this not require following rules and understanding at some level? For example, in Searle's famous computation in the "Chinese Room" (Searle 1980, cf. Preston and Bishop 2002), Searle sits in the room and manipulates Chinese symbols according to manipulation instructions given in English: a language that he understands.

Haugeland (1985, p. 66) claims that in any computing system there are *primitive operations* of which the system knows how to carry them out. Indeed, he says these must involve meaning: "The *only* way that we can *make sense of* a computer *as executing a program* is by understanding its processor as responding to the program descriptions as meaningful" (2002, p. 385).[3]

[3] This understanding may be prompted by the metaphorical use of "command" and similar expressions on several levels of computer use. Not only do we say that a computer "obeys commands" of the user, we also say that a programmer writes commands, even that he/she uses algorithms. This is on a much higher technical level, however, than the one relevant here. A command in a conventional "higher" programming language, in order to be executed, must be translated ("compiled" or "interpreted") into "machine code", a code that the particular machine with a particular operating system can load into its

If this was right, a purely syntactic machine would be impossible. In any computing system we would be back at our original problem: If there are "meaningful primitives" in any computing machine—where do they get their meaning? Our short line would show that computationalism is wrong, unless one accepts semantic nativism. Or rather, semantic nativism must be true for any computer, given that we have working computers. All our computers would already have meaningful symbols built in!

I think it will be apparent form the discussion of descriptive levels above, that purely syntactic machines are possible, however. We just need to be more careful when we say that the system "follows rules", or "executes programs". Wittgenstein famously distinguished between *following* a rule and *acting according to* a rule—and only the former requires that one understands the rule (gives it an interpretation). The computer does not literally follow a rule. Being in a particular state, given a particular input, it will perform a series of steps (e.g. switches) and produce a particular output, a memory state. The same happens when it is programmed, i.e. its switches are set (this even happens in the same central memory, in the case of a 'stored program' von Neumann machine). This is a purely causal, mechanical procedure that requires no understanding of a rule. It is no different from a can vending machine taking a particular input (my coins and my pressing a button), processing, and producing a particular output (the can).

The computing machine is just constructed in such a way that it will mechanically do what *we* call "carrying out a program", on the logical or even the symbolic level. We can describe the computer as "following a rule" and some of its states as "symbols" but that is entirely irrelevant to its functioning. A computer can be described on the symbolic level, but it must not have such a level. It may also, to repeat, be described differently on the symbolic level. The resistance to call computing "purely syntactical" (e.g. Davis 2000, p. 204f; Preston 2002, p. 40f; Hauser 2002; Rey 2002) is perhaps due to the fact that this process is, of course, causal. It is not strictly speaking a formal procedure, but a mechanical one: the computer operates on meaningless symbols with programs that are meaningless to it.[4]

Can Purely Syntactic Computing Acquire Meaning?—A Challenge (the Longer Line)

So, how does syntactic computationalism, thus understood, fare with our problem of symbol grounding? The problem for a computationalist is that she has to construct a causal chain that does not involve any mental process at any stage that is other than

Footnote 3 continued
storage, where it is present in a form that the particular CPU can process. The CPU again, will have thousands of algorithms already built-in ("hard-wired"); it will not need to be programmed to the lowest level of switches each time.

[4] Accordingly, the solution to symbol grounding cannot be to give basic rules, as does for example Hofstadter in his discussion of the matter. For his MU and MIU systems (Hofstadter 1979, Chaps. I & II, p. 170, 264), you assume that rules have meaning. If you do not, then you have to postulate that "absolute meaning" comes about somehow by itself, in "strange loops" (Chap. VI and passim).

purely syntactic. Meaning-involving processes such as attention, object tracking, object-files, interest, etc. are not permitted.

Let us look at some lessons from history to understand the difficulty: I take the discussion about the so-called "causal theory of reference", developed by Putnam and Kripke in the early 1970s, to have shown two things: (a) We want to grant causal connections between tokens of some kinds of symbols and their reference a role in the determination of the meaning of the symbols—in particular, we want to do this in the case of natural kind terms, such as "gold", where the stuff they refer to, the element *gold*, plays a role in the determination of what counts as gold and what does not. This is what Putnam called the "contribution of the environment". (I say "we want to grant" because it is important to see that Putnam's and Kripke's discoveries are discoveries about our linguistic intuitions). (b) The causal relations between, for example, the tokens of "gold" and the element *gold* are immensely complex and it is extremely hard to figure out the particular causal relation that should connect the token to its referent. A given token stands in any number of causal relation and none of these by itself distinguishes itself as the right one ("gold" does not refer to jewelers shops or to chemistry textbooks or to metal or to undiscovered fake gold). What we need is a notion of "explanatory cause", the cause that is relevant for our explanatory *intentions*.[5]

The Putnam/Kripke story shows that the causal relation of a *linguistic* symbol to its referent must involve the intention of speakers to refer to a specific object or kind: otherwise it is underdetermined due to the multiplicity of causal chains.

In earlier papers (Müller 2004; Raftopoulos and Müller 2006a, b), we investigated the psychological evidence that there are simple input mechanisms in vision that result in an "object file" (Kahneman et al. 1992), which could be used for reference grounding without presupposing higher cognitive mechanisms. These mechanisms are bottom up and cognitively encapsulated, so they could presumably be present (if they are computational) in a pre-cognitive computational system. However, even such simple mechanisms require that the system has *attention* directed at an object in sensation in order to differentiate it from the background and from other objects (cf. Raftopoulos 2006, p. 55ff). They are thus not purely syntactic.

To put this in terms of meaning acquisition: How can a system acquire meaningful symbols without making use of cognition or sub-cognitive intentional states? Could there be a theory of language acquisition (or machine learning) that assumes a language can be learned by a system that has no cognitive processes?

[5] What is relevant here is not so much semantic externalism (that has lead to externalism about mental states) but Putnam's critique of his own earlier causal theories of reference. This critique shows that a successful story of the causal relation between my tokens of "gold" and gold has to involve my *desire to refer* to that particular metal with that particular word. Putnam has tried to show this in his model-theoretic argument (1981, p. 34 etc.) and in the point that we need to single out what we mean by "cause", given that any event has several causes—whereas we need the one "explanatory" cause (Putnam 1982, 1985). This is supported by Wittgensteinian arguments to the effect that deixis is necessarily ambiguous (sometimes called the "disjunction problem"). When Kripke pointed at the cat (and Quine's native pointed at the rabbit), were they pointing at a cat, a feline, an animal, a flea, a colour, or a symbol? When Putnam pointed at water, how much H_2O did we need in the sample for reference to be successful?

I propose that to develop such a theory is more than just a challenge: it cannot be done.[6]

Relation to Searle's "Chinese Room Argument"

Let us illustrate the same point in the terms used in Searle's "Chinese room argument". Searle's central notion is "understanding" (of Chinese and of stories) and he claims, (1) that the symbol manipulator in the Chinese room should not be said to *understand* Chinese by virtue of his handling the symbols correctly and thus producing correct output, also that he has no chance of learning Chinese [both of this everybody agrees with], (2) that the whole system containing the Chinese room, with manipulation manuals and all, cannot be said to understand Chinese [the "systems reply"], not even if "sensory organs" (cameras, microphones, etc.) are added [the "robot reply"], since these supply "just more Chinese". He sometimes expresses this as saying that the system has syntax but no semantics for its symbols: that symbols in a system cannot acquire meaning due to mere symbol manipulation.

As several people have pointed out, (2) does not follow from (1). The upshot of the argument is, in my view, that Searle sets the task to explain *how* a system can understand Chinese given that the central symbol manipulator does not. After the Chinese Room Argument the belief that a symbol manipulating system can "understand" is in doubt and would require positive support.

Searle's claim is that he cannot learn Chinese by manipulating the symbols in his room, even if he tries hard—and then he expands this point to the whole system. But he already grants too much: Searle in the room *understands* the symbols in the instructions for manipulation, *wants* to learn Chinese, *knows* that Chinese is a language, that some of its symbols refer and which world they refer to. None of these is given in an actual purely syntactic computational system. Given that there is literally *no* understanding, desire and knowledge in the actual Chinese Room of a syntactic system (there are no intentional states), there is even less reason to believe that there is in the whole system.

The argument presented above thus goes some way towards closing the gap in Searle's argument by explaining why symbol manipulation, even under causal interaction with the environment, cannot produce intention. The system will not acquire meaningful symbols because it lacks everything necessary, specifically it has no *desire* to do so (it has no desires directed at anything). The situation is thus worse than in Searle's "Chinese Room", where Searle tries to show that an intelligent agent operating a purely syntactical system *cannot* acquire meaning. We only need to claim that a purely syntactical system itself *will not* acquire meaning—even if it could. (For the opposing view in 'epigenetic robotics', see Steels 2008 and cf. Taddeo and Floridi 2005.)

[6] Fodor's recent battle against behaviorist accounts of concept possession fires back on his Cartesian theory, when he insists on the problem that knowing how to apply "trilateral" is necessarily also knowing how to apply "triangular", even in counterfactual cases (Fodor 2004, p. 39), since whatever thing typically causes an instance of "triangular" also causes an instance of "trilateral". This is worse than Quine's undetached rabbit parts and, of course, than the rabbit fly as reference for "gavagai".

On a cautionary note, just like Searle, we do claim to have found any bounds as to what can be done with purely syntactic computing. Clearly, advanced AI systems (and perhaps "lower" animals) have achieved impressive feats without the "meaningful symbols" we have been asking for and which humans surely possess (see Müller 2007).

This look into the Chinese room might leave a paradoxical air, one might wonder what that magical bit is which allows humans and other animals what computers cannot have. My suggestion here is that this bit has to be something that is not computational—and I think *desire* is a good candidate.

Taking Stock: Some Conclusions

What we have seen so far is that:

(1) A language of thought computational system presupposes innate meaning
(2) A purely syntactical computational system is possible
(3) A purely syntactical computational system could only acquire meaning if that process does not involve any mental states with intention (e.g. desires, beliefs, attention, …)

What we have not seen is whether there is another version of computationalism that could save the day. Perhaps there is computation without symbols or there is information processing in ways other than computing? Let us take a brief look at the options.

Vacuous Computationalism

Searle has repeatedly said that whether a system is a computer or not depends on its interpretation by some observer, a syntactic property is an observer-relative notion (cf. Preston 2002, pp. 42–44). This is why he comes to the prima facie surprising conclusion that "The brain is a computer, in the sense that it instantiates computer programs,…" because "everything is a digital computer at some level of description" (Searle 2002, p. 224).

Whether this view is true or not (I tend to think it is not; see Piccinini 2007, for a detailed discussion), as Searle knows, this makes computationalism vacuous. Clearly, computationalism cannot be the claim that, if an observer likes to see it that way, the brain is a computer, and so is a train, a tree or a bumblebee.

Non-Symbolic Computing and "Information Processing"

There are cognitive scientists that use the word "computational" in a much weaker sense than the one defined above—in fact, the plethora of definitions is depressing: I counted 9 different ones in a recent exchange between Pinker and Fodor (2005).[7]

[7] Fodor and Pinker in their 2005 exchange alone use the following, most of which are obviously either too narrow or too broad:

One prominent idea is that computing is somehow "information processing". But information processing could take many forms, some of which are not computational. There are many systems that could be used to compute but should not be called a computer. Dynamical systems in the sense of van Gelder (1995) are one example. Another are analogue systems, such as slide rules, mechanic (non-digital) adding machines, scales, tubes, etc. So, even if computing is information processing, what distinguishes it from *other* forms of information processing—some of which may even produce the same results? Surely this must be the *mechanism* by which it achieves that processing: namely computation (i.e. performing algorithms).[8]

I would therefore make the terminological suggestion to distinguish between "computationalism" and "information processing" as paradigms for cognitive science.

Footnote 7 continued

1) Literally being a Turing machine with tape and all (attributed to Fodor by Pinker 2005, p. 6). Falsely attributed and failing to mention that the relevant notion is that of the "universal" Turing machine.

2) "Cognitive architecture is Classical Turing architecture" (Pinker 2005, p. 6). If "architecture" is taken sufficiently abstractly this is different from 1). But what is that "architecture"? Perhaps being able to "compute any partial recursive function, any grammar composed of rewrite rules, and, it is commonly thought, anything that can be computed by any other physically realizable machine that works on discrete symbols and that arrives at an answer in a finite number of steps" (Pinker 2005, p. 6) on Turing). But this is a description of abilities, not of structure.

3) Having "the architecture of a Turing machine or some other serial, discrete, local processor" (Pinker 2005, p. 22—attributed to Fodor). False attribution, since in 2000, Fodor did not mention the possibility of *other* processors. Suggests that "architecture" means physical setup (tape and reader), after all—see problems in 2).

4) Being 'Turing-equivalent', in the sense of 'input–output equivalent' (Fodor 2000, pp. 30, 33, 105n3). Surely too weak. Any information processing system is input–output equivalent to more than one Turing machine.

5) Being 'defined on syntactically structured mental representations that are much like sentences' (Fodor 2000, p. 4). "Defined on" and "much like sentences"? A definition of the language of thought? Not of computation, surely.

6) Being supervenient "on some syntactic fact or other"—"minimal CTM" (Fodor 2000, p. 29). Too minimal, as Fodor himself agrees.

7) Being "causally sensitive to, and only to, the *syntax* of the mental representations they are defined over" [not to meaning] AND being "sensitive only to the *local* syntactic properties of mental representations" (Fodor's upshot in 2005, 26)—delete "mental" above and note that none of this makes for a computational *process*.

8) "In this conception, a computational system is one in which knowledge and goals are represented as patterns in bits of matter ('representations'). The system is designed in such a way that one representation causes another to come into existence; *and* these changes mirror the laws of some normatively valid system like logic, statistics, or laws of cause and effect in the world." (Pinker 2005, p. 2). Any representational systematic process is computational, then.

9) "… human cognition is like *some kind* of computer, presumably one that engages in parallel, analog computation as well as the discrete serial variety". Pinker 2005, p. 34 on Pinker—note the "like", "some kind" and "presumably", plus the circularity of using "computer"!

[8] There are at least two notions of algorithm possible here, depending on whether the step-by-step process is one of symbol manipulation or not. (e.g. Harel 2000 introduces the notion of algorithm via a recipe for making chocolate mousse).

Outlook: Analogue and Hybrid systems

The pages above present a reason to believe that the mind is not a computational system unless it has semantics built in. However, there is still good reason to think that some parts of the human mind are computational, even if the problems explained show that it is not *only* that. Perhaps the picture that emerges is that of a hybrid and modular mind where some modules are computational but many are not. Some of the non-computational systems will be mathematically describable, perhaps as dynamic systems, and can thus be simulated on digital computers to some degree of accuracy. Perhaps some of these cognitive processes are intentional but neither syntactic computation nor LOT computation, and together with the embodiment of the whole, they can explain how meaning can be acquired (if it is not built in).

Acknowledgments My thanks to the people with whom I have discussed this paper, especially to Thanos Raftopoulos, Kostas Pagondiotis and the attendants of the "Philosophy on the Hill" colloquium. I am very grateful to two anonymous reviewers for detailed written comments.

References

Chalmers, D. J. (1993). A computational foundation for the study of cognition. Online at http://consc.net/papers/computation.html.
Chalmers, D. J. (1996). *The conscious mind: In search of a fundamental theory*. Oxford: Oxford University Press.
Churchland, P. (2005). Functionalism at forty: A critical retrospective. *Journal of Philosophy, 102*(1), 33–50.
Copeland, D. (1997). The broad conception of computation. *American Behavioral Scientist, 40*, 690–716.
Copeland, D. (2000). Narrow versus wide mechanism, including a re-examination of Turing's views on the mind–machine issue. *Journal of Philosophy, 97/1*, 5–32.
Copeland, D. (2002). Hypercomputation. *Minds and Machines, 12*, 461–502.
Davis, M. (2000). *The universal computer: The road from Leibniz to Turing*. New York: W. W. Norton.
Fodor, J. (1981). The mind-body problem. Scientific American 244. Reprinted in J. Heil (Ed.), *Philosophy of mind: A guide and anthology* (pp. 168–182). Oxford: Oxford University Press 2004.
Fodor, J. (1994a). *The elm and the expert: Mentalese and Its semantics*. Cambridge, Mass: MIT Press.
Fodor, J. (1994b). Fodor, Jerry A., In S. Guttenplan (Ed.), *A companion to the philosophy of mind* (pp. 292–300). Oxford: Blackwell.
Fodor, J. (1998). *Concepts: Where cognitive science went wrong*. Oxford: Oxford University Press.
Fodor, J. (2000). *The mind doesn't work that way: The scope and limits of computational psychology*. Cambridge, Mass: MIT Press.
Fodor, J. A. (2003). More peanuts. *The London Review of Books, 25*, 09.10.2003.
Fodor, J. (2004). Having concepts: A brief refutation of the twentieth century, with "Reply to Commentators". *Mind and Language, 19,* 29–47, 99–112.
Fodor, J. (2005). Reply to Steven Pinker 'so how does the mind work?'. *Mind & Language, 20*(1), 25–32.
Gärdenfors, P. (2000). *Conceptual spaces: The geometry of thought*. Cambridge, Mass: MIT Press.
Harel, D. (2000). *Computers Ltd.: What they really can't do*. Oxford: Oxford University Press.
Harnad, S. (1990). The symbol grounding problem. *Physica D, 42*, 335–346.
Haugeland, J. (1985). *Artificial intelligence: The very idea*. Cambridge, Mass: MIT-Press.
Haugeland, J. (2002). Syntax, semantics, physics. In Preston & Bishop (pp. 379–392).
Hauser, L. (2002). Nixin' goes to China. In Preston & Bishop (pp. 123–143).
Hofstadter, D. R. (1979). *Gödel, Escher, Bach: An eternal golden Braid*. New York: Basic Books.
Kahneman, D., Treisman, A., & Gibbs, B. J. (1992). The reviewing of the object files: Object-specific integration of information. *Cognitive Psychology, 24*, 174–219.

Kim, J. (1996). *Philosophy of mind*. Boulder, Col: Westview Press.

Lycan, W. G. (2003). Philosophy of mind. In N. Bunnin & E. P. Tsui-James (Eds.), *The Blackwell companion to philosophy* (2nd ed., pp. 173–202). Oxford: Blackwell.

Müller, V. C. (2004). There must be encapsulated nonconceptual content in vision. In A. Raftopoulos (Ed.), *Cognitive penetrability of perception: Attention, action, attention and bottom-up constraints* (pp. 181–194). Huntington, NY: Nova Science.

Müller, V. C. (2007). Is there a future for AI without representation? *Minds and Machines, 17*, 101–115.

Müller, V. C. (2008). Representation in digital systems. In A. Briggle, K. Waelbers & P. Brey (Eds.), *Current issues in computing and philosophy* (pp. 116–121). Amsterdam: IOS Press.

Piccinini, G. (2007). Computational modeling vs. computational explanation: Is everything a Turing machine and does it matter to the philosophy of mind? *The Australasian Journal of Philosophy, 85*, 93–116.

Piccinini, G. (2008). Computation without representation. *Philosophical Studies, 134*, 205–241.

Pinker, S. (2005). So how does the mind work? and A reply to Jerry Fodor on how the mind works. *Mind & Language, 20*(1), 1–24, 33–38.

Preston, J. (2002). Introduction. In Preston & Bishop (pp. 1–50).

Preston, J., & Bishop, M. (Eds.). (2002). *Views into the Chinese room: New essays on searle and artificial intelligence*. Oxford: Oxford University Press.

Putnam, H. (1981). *Reason, truth and history*. Cambridge: Cambridge University Press.

Putnam, H. (1982). Why there isn't a ready-made world. In *Realism and reason: Philosophical papers* (Vol. 3, pp. 205–228). Cambridge: Cambridge University Press.

Putnam, H. (1985). Reflexive reflections. In *Words and life* (pp. 416–427). Cambridge, Mass: Harvard University Press 1994.

Raftopoulos, A. (2006). Defending realism on the proper ground. *Philosophical Psychology, 19*(1), 47–77.

Raftopoulos, A., & Müller, V. C. (2006a). The phenomenal content of experience. *Mind and Language, 21*(2), 187–219.

Raftopoulos, A., & Müller, V. C. (2006b). Nonconceptual demonstrative reference. *Philosophy and Phenomenological Research, 72*, 251–285.

Rey, G. (2002). Searle's misunderstandings of functionalism and strong AI. In Preston & Bishop (pp. 201–225).

Schneider, U., & Werner, D. (2001). *Taschenbuch der Informatik* (4th ed ed.). Leipzig: Fachbuchverlag Leipzig.

Searle, J. (1980). Minds, brains and programs. *Behavioral and Brain Sciences, 3*, 417–457.

Searle, J. (2002). *Consciousness and language*. Cambridge: Cambridge University Press.

Smolensky, P. (1988). Computational models of mind. In S. Guttenplan (Ed.), *A companion to the philosophy of mind* (pp. 176–185). Oxford: Blackwell.

Smolensky, P. (1999). On the proper treatment of connectionism. *Behavioral and Brain Sciences, 11*, 1–23.

Steels, L. (2008). The symbol grounding problem has been solved, so what's next? In M. de Vega, A. Glenberg & A. Graesser (Eds.), *Symbols and embodiment: Debates on meaning and cognition* (pp. 223–244). Oxford: Oxford University Press.

Taddeo, M., & Floridi, L. (2005). Solving the symbol grounding problem: A critical review of fifteen years of research. *Journal of Experimental and Theoretical Artificial Intelligence, 17*, 419–445.

Van Gelder, T. (1995). What might cognition be if not computation? *The Journal of Philosophy, 91*(7), 345–381.

Wakefield, J. C. (2003). The Chinese room argument reconsidered: Essentialism, indeterminacy, and strong AI. *Minds and Machines, 13*, 285–319.