Semiotic Systems, Computers, and the Mind:
How Cognition Could Be Computing

William J. Rapaport, University at Buffalo, The State University of New York, USA

ABSTRACT

In this reply to James H. Fetzer's "Minds and Machines: Limits to Simulations of Thought and Action", the author argues that computationalism should not be the view that (human) cognition is computation, but that it should be the view that cognition (simpliciter) is computable. It follows that computationalism can be true even if (human) cognition is not the result of computations in the brain. The author also argues that, if semiotic systems are systems that interpret signs, then both humans and computers are semiotic systems. Finally, the author suggests that minds can be considered as virtual machines implemented in certain semiotic systems, primarily the brain, but also AI computers. In doing so, the author takes issue with Fetzer's arguments to the contrary.

Keywords: *Cognition, Computationalism, Semantics, Semiotic Systems, Syntax*

1. INTRODUCTION

This essay is a reply to James H. Fetzer's essay in this journal, "Minds and Machines: Limits to Simulations of Thought and Action" (Fetzer, 2011). I treat three topics: computationalism, semiotic systems, and cognition (the mind), offering what I feel is the proper treatment of computationalism. From this, certain views about semiotic systems and minds follow (or, at least, are consistent): First, I argue that computationalism should not be understood as the view that (human) cognition *is* computation, but that it should be understood as the view that cognition (human or otherwise) is comput*able* (§2). On this view, it follows that computationalism can be true even if (human) cognition is not the result of computations in the brain. Second, I argue that, if semiotic systems are systems that interpret signs, then both humans and computers are semiotic systems (§5.2). Finally, I suggest that minds should be considered as virtual machines implemented in certain semiotic systems: primarily brains, but also AI computers (§6.3). In the course of presenting and arguing for these positions, I respond to Fetzer's arguments to the contrary (§§5–8).[1]

DOI: 10.4018/ijsss.2012010102

2. THE PROPER TREATMENT OF COMPUTATIONALISM

Computationalism is often characterized as the thesis that cognition is computation. Its origins can be traced back at least to Thomas Hobbes:

"For REASON, in this sense [i.e., "as among the faculties of the mind"], is nothing but reckoning—that is, adding and subtracting—of the consequences of general names agreed upon for the marking and signifying of our thoughts…" (Hobbes 1651, Part I, Ch. 5, p. 46)[2]

It is a view whose popularity, if not its origins, has been traced back to McCulloch and Pitts (1943), Putnam (1960 or 1961), and Fodor (1975) (Horst, 2009; Piccinini, 2010). This is usually interpreted to mean that the mind, or the brain—whatever it is that exhibits cognition—computes, or *is* a computer. Consider these passages, more or less (but not entirely) randomly chosen:[3]

- A Plan is any hierarchical process in the organism that can control the order in which a sequence of operations is to be performed. A Plan is, for an organism, essentially the same as a program for a computer (Miller et al., 1960, p. 16).[4]
- [H]aving a propositional attitude is being in some *computational* relation to an internal representation. …Mental states are relations between organisms and internal representations, and causally interrelated mental states succeed one another according to computational principles which apply formally *to the representations* (Fodor, 1975, p. 198).
- [C]ognition ought to be viewed as computation. [This] rests on the fact that computation is the only worked-out view of *process* that is both compatible with a materialist view of how a process is realized and that attributes the behavior of the process to the operation of rules upon representations. In other words, what makes it possible to view computation and cognition as processes of fundamentally the same type is the fact that both are physically realized and both are governed by rules and representations.… (Pylyshyn, 1980, p. 111).
- [C]ognition *is* a type of computation (Pylyshyn, 1985, p. xiii).
- 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 (Searle, 1990, p. 21).
- Computationalism is the hypothesis that cognition *is* the computation of functions. …The job for the computationalist is to determine…which specific functions explain specific cognitive phenomena (Dietrich, 1990, p. 135, emphasis added).
- [T]he Computational Theory of Mind… is…the best theory of cognition that we've got…. (Fodor, 2000, p. 1).
- Tokens of mental processes are 'computations'; that is, causal chains of (typically inferential) operations on mental representations (Fodor, 2008, pp. 5-6).
- The core idea of cognitive science is that our brains are a kind of computer…. Psychologists try to find out exactly what kinds of programs our brains use, and how our brains implement those programs (Gopnik, 2009, p. 43).
- [A] particular philosophical view that holds that the mind literally is a digital computer…, and that thought literally is a kind of computation…will be called the "Computational Theory of Mind"…. (Horst, 2009).
- Computationalism…is the view that the functional organization of the brain (or any other functionally equivalent system) is computational, or that neural states are computational states (Piccinini, 2010, p. 271).
- These remarkable 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 (Piccinini, 2010, pp. 277-278).

That cognition is computation is an interesting claim, one well worth exploring, and it may even be true. But it is too strong: It is not the kind of claim that is usually made when one says that a certain behavior can be understood computationally (Rapaport, 1998). There is a related claim that, because it is weaker, is more likely to be true and—more importantly—is equally relevant to computational theories of cognition, because it preserves the crucial insight that cognition is capable of being explained in terms of the mathematical theory of computation.

Before stating what I think is the proper version of the thesis of computationalism, let me clarify two terms:

1. I will use 'cognition'[5] as a synonym for such terms as 'thinking', 'intelligence' (as in 'AI', not as in 'IQ'), 'mentality', 'understanding', 'intentionality', etc. Cognition is whatever cognitive scientists study, including (in alphabetical order) believing (and, perhaps, knowing), consciousness, emotion, language, learning, memory, (perhaps) perception, planning, problem solving, reasoning, representation (including categories, concepts, and mental imagery), sensation, thought, etc. Knowing might *not* be part of cognition, insofar as it depends on the way the *world* is (knowing is often taken to be justified *true* belief) and thus would be independent of what goes on in the mind or brain; perception also depends on the way the world is (see §3.1).
2. An "algorithm" for an executor E to achieve a goal G is, informally, a procedure (or "method") for E to achieve G, where (a) E is the agent—human or computer—that carries out the algorithm (or executes, or implements, or "follows" it), (b) the procedure is a set (usually, a sequence) of statements (or "steps", usually "rules" or instructions), and (c) G is the solution of a (particular kind of) problem, the answer to a (particular kind of) question, or the accomplishment of some (particular kind of) task. (See the Appendix for more details.)

Various of these features can be relaxed: One can imagine a procedure that has all these features of algorithms but that has no specific goal, e.g., "Compute 2+2; then read *Moby Dick*.", or one for which there is no executor, or one that yields output that is only *approximately* correct (sometimes called a 'heuristic'; see §6.1), etc. For alternative informal formulations of "algorithm", see the Appendix.

Several different mathematical, hence precise, formulations of this still vague notion have been proposed, the most famous of which is Alan Turing's (1936) notion of (what is now called in his honor) a 'Turing machine'. Because all of these precise, mathematical formulations are logically equivalent, the claim that the informal notion of "algorithm" *is* a Turing machine is now known as "Turing's thesis" (or as "Church's thesis" or the "Church-Turing thesis", after Alonzo Church, whose "lambda calculus" was another one of the mathematical formulations).

Importantly, for present purposes, when someone says that a mathematical function (see note 42) or a certain phenomenon or behavior is "computable", they mean that there is an algorithm that outputs the values of that function when given its legal inputs[6] or that produces that phenomenon or behavior—i.e., that one could write a computer program that, when executed on a suitable computer, would enable that computer to perform (i.e., to output) the appropriate behavior.

Hence,

Computationalism, properly understood, should be the thesis that cognition is computable, i.e., that there is an algorithm (more likely, a family of algorithms) that computes cognitive functions.

I take the basic research question of computational cognitive science to ask, "*How much*

of cognition is computable?" And I take the working assumption (or expectation, or hope) of computational cognitive science to be that *all* cognition is computable. This formulation of the basic research question allows for the possibility that the hopes will be dashed—that some aspects of cognition might *not* be computable. In that event, the interesting question will be: *Which* aspects are not computable, and why?[7]

Although several philosophers have offered "non-existence proofs" that cognition is *not* computable,[8] none of these are so mathematically convincing that they have squelched all opposition. And, in any case, it is obvious that *much* of cognition *is* computable (for surveys, see, Johnson-Laird, 1988; Edelman, 2008b; Forbus, 2010). Philip N. Johnson-Laird (1988, pp. 26-27) has expressed it well:

"The goal of cognitive science is to explain how the mind works. Part of the power of the discipline resides in the theory of computability. ...Some processes in the nervous system seem to be computations…. Others…are physical processes that can be modeled in computer programs. But there may be aspects of mental life that cannot be modeled in this way…. There may even be aspects of the mind that lie outside scientific explanation."

However, I suspect that *so* much of cognition will eventually be shown to be computable that the residue, if any, will be negligible and ignorable.

This leads to the following "implementational implication": If (or to the extent that) cognition *is* computable, then *anything* that implements cognitive computations would *be* (to that extent) cognitive. Informally, such an implementation would "really think". As Newell, Shaw, and Simon (1958, p. 153) put it (explicating Turing's notion of the "universal" Turing machine, or stored-program computer), "if we put any particular program in a computer, we have in fact a machine that behaves in the way prescribed by the program". The "particular program" they were referring to was one for "human problem solving", so a computer thus programmed would indeed solve problems, i.e., exhibit a kind of cognition.

This implication is probably a more general point, not necessarily restricted to computationalism. Suppose, as some would have it, that cognition turns out to be fully understandable only in terms of differential equations (Forbus, 2010, §1, hints at this but does not endorse it) or dynamic systems (van Gelder, 1995). Arguably, anything that implements cognitive differential equations or a cognitive dynamic system would be cognitive.

The more common view, that cognition is comput*ation*, is a "strong" view that the mind or brain *is* a computer. It claims that *how* the mind or brain does what it does is by computing. My view, that cognition is comput*able*, is a weaker view that *what* the mind or brain does can be *described* in computational terms, but that *how* it does it is a matter for neuroscience to determine.[9]

Interestingly, some of the canonical statements of "strong" computationalism are ambiguous between the two versions. Consider some of Fodor's early statements in his *Language of Thought* (1975):

"[H]aving a propositional attitude is being in some computational relation to an internal representation." (p. 198, original emphasis)

This could be interpreted as the weaker claim that the relation is comput*able*. The passage continues:

"The intended claim is that the sequence of events that causally determines the mental state of an organism will be describable as a sequence of steps in a derivation…." (p. 198, emphasis added)

The use of 'causally' suggests the stronger—implementational—view, but the use of 'describable as' suggests the weaker view. There's more:

"More exactly: Mental states are relations between organisms and internal representations, and causally interrelated mental states succeed one another **according to** *computational principles which apply formally to the representations."* (p. 198, bold emphasis added)

If 'according to' means merely that they behave in accordance with those computational principles, then this is consistent with my—weaker—view, but if it means that they execute those principles, and then it sounds like the stronger view. Given Fodor's other comments and the interpretations of other scholars, and in light of later statements such as the quote from 2008, I'm sure that Fodor always had the stronger view in mind. But the potentially ambiguous readings give a hint of the delicacy of interpretation.[10]

That cognition is comput*able* is a necessary—but not sufficient—condition for it to be computa*tion*. The crucial difference between cognition as being comput*able* rather than as *being* computa*tion* is that, on the weaker view, the implementational implication holds *even if humans don't implement cognition computationally.* In other words, it allows for the possibility that human cognition is comput*able* but is not compute*d.* For instance, Gualtiero Piccinini (2005, 2007) has argued that "spike trains" (sequences of "action potential") in groups of neurons—which, presumably, implement human cognition—are not representable as strings of digits, hence not computational. But this does not imply that the *functions*[11] whose outputs they produce are not comput*able*, possibly by different mechanisms operating on different primitive elements in a different (perhaps non-biological) medium.

And Makuuchi et al. (2009, p. 8362) say:

"If the processing of PSG [phrase structure grammar] is fundamental to human language, the [sic] questions about how the brain implements this faculty arise. The left pars opercularis (LPO), a posterior part of Broca's area, was found as a neural correlate of the processing of A^nB^n sequences in human studies by an artificial grammar learning paradigm comprised of visually presented syllables…. These 2 studies therefore strongly suggest that LPO is a candidate brain area for the processor of PSG (i.e., hierarchical structures)."

This is consistent with comput*ability* without computa*tion*. However, Makuuchi et al. (2009, p. 8365) later say:

"The present study clearly demonstrates that the syntactic computations involved in the processing of syntactically complex sentences is neuroanatomically separate from the non-syntactic VWM [verbal working memory], thus favoring the view that syntactic processes are independent of general VWM."

That is, brain locations where real computa*tion* is needed in language processing are anatomically distinct from brain locations where computation is *not* needed. This suggests that the brain *could* be computational, contra Piccinini.

Similarly, David J. Lobina (2010) (Lobina & García-Albea, 2009) has argued that, although certain cognitive capabilities are recursive (another term that is sometimes used to mean "computable"), they might not be implemented in the brain in a recursive fashion. After all, algorithms that are most efficiently expressed recursively are sometimes compiled into more-efficiently executable, iterative (non-recursive) code.[12]

Often when we investigate some phenomenon (e.g., cognition, life, computation, flight), we begin by studying it as it occurs in nature, and then abstract away (or "ascend") from what might be called 'implementation details' (Rapaport, 1999, 2005b) to arrive at a more abstract or general version of the phenomenon, from which we can "descend" to (re-)implement it in a different medium. When this occurs, the term that referred to the original (concrete) phenomenon changes its referent to the abstract phenomenon and then becomes applicable—perhaps metaphorically so—to the new (concrete) phenomenon. So, for

instance, flight as it occurs in birds has been reimplemented in airplanes; 'flying' now refers to the more abstract concept that is multiply realized in birds and planes (cf. Ford & Hayes, 1998; Rapaport, 2000, §2.2; Forbus, 2010, p. 2). And computation as done by humans in the late 19th through early 20th centuries[13] was—after Turing's analysis—reimplemented in machines; 'computation' now refers to the more abstract concept. Indeed, Turing's (1936) development of (what would now be called) a *computational* theory of *human* computation seems to me to be pretty clearly the first AI program! (See the Appendix, and cf. my comments about operating systems in §6.)

The same, I suggest, may (eventually) hold true for 'cognition' (Rapaport, 2000). (And, perhaps, for artificial "life"). As Turing (1950, p. 442; boldface emphasis added) said,

"The original question, 'Can machines think?' I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the ***use of words*** *and* ***general educated opinion*** *will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."*

"General educated opinion" changes when we abstract and generalize, and "the use of words" changes when we shift reference from a word's initial application to the more abstract or general phenomenon. Similarly, Derek Jones (2010) proposed the following "metaphysical thesis": "Underlying biological mechanisms are irrelevant to the study of behavior/systems as such. The proper object of study is the abstract system considered as a multiply realized high-level object."

This issue is related to a dichotomy in cognitive science over its proper object of study: Do (or should) cognitive scientists study *human* cognition in particular, or (abstract) cognition in general? Computational psychologists lean towards the former; computational philosophers (and AI researchers) lean towards the latter (see note 9; cf. Levesque, 2012, who identifies the former with cognitive science and the latter with AI). We see this, for example, in the shift within computational linguistics from developing algorithms for understanding natural language using "human language concepts" to developing them using statistical methods: Progress was made when it was realized that "you don't have to do it like humans" (Lohr, 2010, quoting Alfred Spector on the research methodology of Frederick Jelinek; for further discussion of this point, see Forbus, 2010, §2).

3. SYNTACTIC SEMANTICS

Three principles underlie computationalism properly treated. I call them "internalism", "syntacticism", and "recursive understanding". Together, these constitute a theory of "syntactic semantics".[14] In the present essay, because of space limitations, I will primarily summarize this theory and refer the reader to earlier publications for detailed argumentation and defense (Rapaport, 1986, 1988, 1995, 1998, 1999, 2000, 2002, 2003a, 2005b, 2006, 2011).

3.1. Internalism

Cognitive agents have direct access only to internal representatives of external objects.

This thesis is related to theses called "methodological solipsism" (Fodor, 1980, arguing against Putnam, 1975)[15] and "individualism" (see, e.g., Segal, 1989, arguing against Burge, 1986).[16] It is essentially Kant's point embodied in his distinction between noumenal things-in-themselves and their phenomenal appearances as filtered through our mental concepts and categories: We only have (direct) access to "phenomena", not noumena. Or, as expressed by a contemporary computational cognitive scientist, "My phenomenal world…[is] a neural fiction perpetrated by the senses" (Edelman, 2008b, p. 426).

Internalism is the inevitable consequence of the fact that "the output [of sensory transducers] is…the only contact the cognitive system ever has with the environment" (Pylyshyn, 1985, p. 158).[17] As Ray Jackendoff (2002, §10.4)

put it, a cognitive agent understands the world by "pushing the world into the mind". Or, as David Hume (1777/1902, §12, part 1, p. 152) put it, "the existences which we consider, when we say, *this house* and *that tree*, are nothing but perceptions in the mind and fleeting copies or representations of other existences" (cf. Rapaport, 2000, §5).

This can be seen clearly in two cases:

(1) What I *see* is the result of a long process that begins outside my head with photons reflected off the physical object that I am looking at, and that ends with a qualitative mental image (what is sometimes called a "sense datum"; cf. Huemer, 2011) produced by neuron firings in my brain. These are internal representatives of the external object. Moreover, there is a time delay; what I am consciously aware of at time *t* is my mental representative of the external object as it was at some earlier time *t'*:

"Computation necessarily takes time and, because visual perception requires complex computations, ...there is an appreciable latency—on the order of 100msec—between the time of the retinal stimulus and the time of the elicited perception..." (Changizi et al., 2008, p. 460)

This is in *addition* to the time that it takes the reflected photons to reach my eye, thus *beginning* the computational process. I agree with Huemer (2011) that this implies that we do "not directly perceive *anything*…outside of" us (emphasis added).

(2) Although my two eyes *look at* a single external object, they do so from different perspectives; consequently, they *see* different things. These two perceptions are combined by the brain's visual system into a single, three-dimensional perception, which is constructed and internal (cf. Julesz's, 1971 notion of the "cyclopean eye"). Moreover, I can be aware of (i.e., perceive) the two images from my eyes simultaneously; I conclude from this that what I perceive is not what is "out there": There is only *one* thing "out there", but I perceive *two* things (cf. Hume, 1739/1888, Book I, part 4, §2, pp. 210-211; Ramachandran & Rogers-Ramachandran, 2009). Similarly the existence of saccades implies that "my subjective, phenomenal experience of a static scene" is internal ("irreal", a "simulation of reality") (Edelman, 2008b, p. 410).

There is a related point from natural-language semantics: Not all words of a language have an external referent, notably words (like 'unicorn') for non-existents. In Rapaport 1981, I argued that it is best to treat *all* words uniformly as having *only* internal "referents". Just as my meaning for 'unicorn' will be some internal mental concept, so should my meaning for 'horse'.

Consequently, both words and their meanings (including any external objects that serve as the referents of certain words) are represented internally in a *single* language of thought (LOT; see Fodor, 1975). "Methodological solipsism"—the (controversial) position that access to the external world is unnecessary (Fodor, 1980; cf. Rapaport, 2000)[18]—underlies representationalism:

"If a system—creature, network router, robot, mind—cannot "reach out and touch" some situation in which it is interested, another strategy, deucedly clever, is available: it can instead exploit meaningful or representational structures in place of the situation itself, so as to allow it to behave appropriately with respect to that distal, currently inaccessible state of affairs." (B.C. Smith, 2010, §5a)

For computers, the single, internal LOT might be an artificial neural network or some kind of knowledge-representation, reasoning, and acting system (such as SNePS) (Shapiro & Rapaport, 1987). For humans, the single, internal LOT is a biological neural network.[19]

It is this last fact that allows us to respond to most of the objections to internalism (see, e.g., Huemer, 2011 for a useful compendium of them). For example, consider the objection that an internal mental representative (call it a "sense datum", "quale", whatever) of, say, one of Wilfrid Sellars's pink ice cubes is neither pink (because it does not reflect any light) nor cubic (because it is not a three-dimensional physical object). Suppose that we really are looking at an external, pink ice cube. Light reflects off the surface of the ice cube, enters my eye through the lens, and is initially processed by the rods and cones in my retina, which transduce the information[20] contained in the photons into electrical and chemical signals that travel along a sequence of nerves, primarily the optic nerve, to my visual cortex. Eventually, I see the ice cube (or: I have a mental image of it). Exactly how that experience of seeing (that mental image) is produced is, of course a version of the "hard" problem of consciousness (Chalmers, 1996). But we do know that certain neuron firings that are the end result of the ice cube's reflection of photons into my eyes are (or are correlated with) my visual experience of pink; others are (correlated with) my visual experience of cubicness. But now *imagine* a pink ice cube; presumably, the same or similar neurons are firing and are (correlated with) my mental image of pinkness and cubicness. In both cases, it is those neuron firings (or whatever it is that might be correlated with them) that constitute my internal representative. In *neither* case is there anything internal that *is* pink or cubic; in *both* cases, there *is* something that *represents* pinkness or cubicness (cf. Shapiro, 1993; Rapaport, 2005b; Shapiro & Bona, 2010).

As I noted in §2, perception, like knowledge, might not be a strictly (i.e., internal) cognitive phenomenon, depending as it does on the external world.[21] When I see the ice cube, certain neuron firings are directly responsible for my visual experience, and I might think, "That's a pink ice cube." That thought is, presumably, also due to (or identical with) some (other) neuron firings. Finally, presumably, those two sets of neuron firings are somehow correlated or associated, either by the visual ones causing the conceptual ones or both of them being caused by the visual stimulation; in any case, they are somehow "bound" together.

My experience of the pink ice cube and my thought (or thoughts) that it is pink and cubic (or that there is a pink ice cube in front of me) occur purely in my brain. They are, if you will, purely solipsistic. (They are not merely *methodologically* solipsistic. Methodological solipsism is a research strategy: A third-person observer's theory of my cognitive processes that ignored the real ice cube and paid attention only to my neuron firings would be methodologically solipsistic.) Yet there are causal links between the neurological occurrences (my mental experiences) and an entity in the real world, namely, the ice cube.

What about a cognitively programmed computer or robot? Suppose that it has a vision system and that some sort of camera lens is facing a pink ice cube. Light reflects off the surface of the ice cube, enters the computer's vision system through the lens, and is processed by the vision system (say, in some descendent of the way that Marr 1982 described). Eventually, let's say, the computer constructs a representation of the pink ice cube in some knowledge-representation language (it may be a pictorial language). When the computer sees (or "sees") the ice cube, it might think (or "think"), "That's a pink ice cube." That thought might also be represented in the same knowledge-representation language (e.g., as is done in the knowledge-representation, reasoning, and acting system SNePS). Finally, those two representations are associated (Srihari & Rapaport, 1989, 1990).

The computer's "experience" of the pink ice cube and its thought (or "thoughts") that it is pink and cubic (or that there is a pink ice cube in front of it) occur purely in its knowledge base. They are purely solipsistic. Yet there are causal links between the computational representations and the ice cube in the real world. *There is no significant difference between the computer and the human.* Both can "ground" their "thoughts" of the pink ice cube in reality yet deal with their

representations of both the phrase 'ice cube' and the ice cube in the same, purely syntactic, language of thought. Each can have a syntactic, yet semantic, relation between its internal representation of the linguistic expression and its internal representation of the object that it "means", and each can have *external* semantic relations between those internal representations and the real ice cube. However, neither can have direct perceptual access to the real ice cube to see if it matches their representation:

"Kant was rightly impressed by the thought that if we ask whether we have a correct conception of the world, we cannot step entirely outside our actual conceptions and theories so as to compare them with a world that is not conceptualized at all, a bare "whatever there is." (Williams, 1998, p. 40)

Of course, both the computer (if equipped with effectors) and the human can grasp the real ice cube.[22]

It might be objected that internalism underestimates the role of situatedness and embodiment of cognitive agents. Quite the contrary. First, any situated or embodied cognitive agent must internalize the information it receives from the environment that it is situated in, and it must process and respond to that information in the form in which it is received. Such information may be incomplete or noisy, hence not fully representative of the actual environment, but those are the cards that that agent has been dealt and that it must play with (cf. Shapiro & Rapaport, 1991). Second, the software- and hardware-embodied cognitive agents developed by colleagues in my research group operate (i.e., are situated) in real and virtual environments, yet are constructed on the "internal" principles adumbrated here, so they constitute a demonstration that situatedness and embodiment are not inconsistent with internal processing of symbols (Shapiro, 1998, 2006; Shapiro et al., 2001; Santore & Shapiro, 2003; Shapiro & Ismail, 2003; Goldfain, 2006; Anstey et al., 2009; Shapiro & Bona, 2010; and cf. Vera & Simon, 1993).

Finally, there is the issue of whether a cognitive computer (or a cognitively programmed computational agent) must have a sensory-motor interface to the real, or a virtual, environment in which it is situated. I am willing to limit my arguments to computers that do have such an interface to the real, external world. If one is willing to allow such an interface to a *virtual* world, however, and if that virtual world is completely independent from the computational agent, then the two situations (real vs. virtual) are parallel. If on the other hand, the virtual environment is completely internal to the computational agent (i.e., the agent believes falsely that it is really situated in that environment), then we have a situation about which there might be disagreement. However, I would maintain that such a (delusional!) agent is in a situation no different from the first kind of agent, because, in both cases, the agent must internalize its environment.

3.2. Syntacticism

It follows that words, their meanings, and semantic relations between them are all syntactic.

Both 'syntax' and 'semantics' can mean different things. On one standard interpretation, 'syntax' is synonymous with 'grammar', and 'semantics' is synonymous with 'meaning' or 'reference' (to the external world). But more general and inclusive conceptions can be found in Charles Morris (1938, pp. 6-7):

"One may study the relations of signs to the objects to which the signs are applicable. ...the study of this [relation] ...will be called semantics. [The study of] ...the formal relation of signs to one another ...will be named syntactics."

On the nature of syntax, consider this early definition from the *Oxford English Dictionary*: "Orderly or systematic arrangement of parts or elements; constitution (of a body); a connected order or system of things."[23] In a study of the history of the concept, Roland Posner (1992, p. 37) says that syntax "is that branch of Semiot-

ics that studies the formal aspects of signs, the relations between signs, and the combinations of signs". On both of those senses, 'syntax' goes far beyond grammar. Throughout this essay, I use 'syntax' in that broader sense (except when quoting); when I intend the narrower meaning, I will use 'grammar' or 'grammatical syntax.'

On the nature of semantics, we might compare Alfred Tarski's (1944, p. 345) characterization: "*Semantics* is a discipline which, speaking loosely, *deals with certain relations between expressions of a language and the objects…'referred to' by those expressions*" (original italics and original scare quotes around 'referred to', suggesting that the relation need not be one of reference to *external* objects). But, surely, translation from one language to another is also an example of semantic interpretation, though of a slightly different kind: Rather than semantics considered as relations between linguistic expressions and objects in the world, in translation it is considered as relations between linguistic expressions in one language and linguistic expressions in another language (we say that the French word 'chat' *means* "cat" in English).

In fact, *all* relationships between two domains can be seen as interpretations of one of the domains in terms of the other—as a mapping from one domain to another. A mapping process is an algorithm that converts, or translates, one domain to another (possibly the same one). The input domain is the syntactic domain; the output domain is the semantic domain. (I have argued elsewhere that implementation, or realization, of an "abstraction" in some "concrete" medium is also such a mapping, hence a semantic interpretation. (see Rapaport 1999, 2005b; cf. Dresner, 2010).

Generalizing only slightly, both syntax and semantics are concerned with relationships: syntax with relations *among* members of a *single* set (e.g., a set of signs, or marks,[24] or neurons, etc.), and semantics with relations *between two* sets (e.g., a set of signs, marks, neurons, etc., on the one hand, and a set of their meanings, on the other). More generally, semantics can be viewed as the study of relations between any two sets whatsoever, including, of course, two sets of signs (as in the case of translation) or even a set and itself; in both of those cases, semantics becomes syntax. Note that a special case of this is found in an ordinary, monolingual dictionary, where we have relations between linguistic expressions in, say, English and (other) linguistic expressions *also* in English. That is, we have relations *among* linguistic expressions in English. But this is syntax!

"Pushing" meanings into the same set as symbols for them allows semantics to be done syntactically: It turns *semantic* relations *between* two sets (a set of internal marks (see §5.1) and a set of (possibly external) meanings) into *syntactic* relations *among* the marks of a *single* (internal) LOT (Rapaport, 2000, §§4, 5, 2003a, §3.1, 2006, § "Thesis 1"; 2011). For example, both truth-table semantics and formal semantics are syntactic enterprises: Truth tables relate one set of marks (strings) representing propositions to another set of marks (e.g., letters 'T' and 'F') representing truth-values. Formal semantics relates one set of marks (strings) representing propositions to another set of marks representing (e.g.) set-theoretical objects (cf. B.C. Smith, 1982). The relations between sets of neuron firings representing signs and sets of neuron firings representing external meanings are also syntactic. *Consequently, symbol-manipulating computers can do semantics by doing syntax.* As Shimon Edelman (2008a, pp. 188-189) put it, "the meaning of an internal state (which may or may not be linked to an external state of affairs) for the system itself is most naturally defined in terms of that state's relations to its other states", i.e., syntactically.

This is the notion of semantics that underlies the Semantic Web, where "meaning" is given to (syntactic) information on the World Wide Web by associating such information (the "data" explicitly appearing in Web pages, usually expressed in a natural language) with more information ("metadata" that only appears in the HTML source code for the webpage, expressed in a knowledge-representation language such as RDF). But it is not "more of the same" kind of information; rather, the additional informa-

tion takes the form of annotations of the first kind of information. Thus, relations are set up between information on, say, Web pages and annotations of that information that serve as semantic interpretations of the former. As John Hebeler has put it, "[T]he computer doesn't know anything more than it's a bunch of bits [i.e., (to rephrase his informal, spoken English), the computer only knows (i.e., doesn't know anything more than) that the Web is a bunch of bits].[25] So semantics merely adds extra information to help you with the *meaning* of the information" (as cited in Ray, 2010).

This is how data are interpreted by computer programs. Consider the following description from a standard textbook on data structures (my comments are interpolated as endnotes, not unlike the "semantic" metadata of the Semantic Web!):

"[T]he concept of information in computer science is similar to the concepts of point, line, and plane in geometry—they are all undefined terms about which statements can be made but which cannot be explained in terms of more elementary concepts.[26] ...The basic unit of information is the bit, whose value asserts one of two mutually exclusive possibilities.[27] ...[I]nformation itself has no meaning. Any meaning can be assigned to a particular bit pattern as long as it is done consistently. It is the interpretation of a bit pattern that gives it meaning.[28] ...A method of interpreting a bit pattern is often called a data type. ...It is by means of declarations[29] [in a high-level language] that the programmer specifies how the contents of the computer memory are to be interpreted by the program. ...[W]e…view…data types as a method of interpreting the memory contents of a computer (Tenenbaum & Augenstein, 1981, pp. 1, 6, 8)."[30]

3.3. Recursive Understanding

Understanding is recursive: We understand one kind of thing in terms of another that is already understood; the base case is to understand something in terms of itself, which is syntactic understanding.

There are two ways to understand something: One can understand something in terms of something else, or one can understand something directly (Rapaport, 1995). The first way, understanding one kind of thing in terms of another kind of thing, underlies metaphor, analogy, maps and grids (B. Smith, 2001), and simulation (see §8). It is also what underlies the relation between syntax and semantics. In the stereotypical case of semantics, we interpret, or give meanings to, *linguistic expressions*. Thus, we understand language in terms of the world. We can also interpret, or give meanings to, other kinds of (non-linguistic) *things*. For example, we can try to understand the nature of certain neuron firings in our brains: Is some particular pattern of neuron firings correlated with, say, thinking of a unicorn or thinking that apples are red? If so, then we have interpreted, hence understood, those neuron firings. And not only can we understand language in terms of the world, or understand parts of the world in terms of other parts of the world, but we can also understand *the world* in terms of *language about* the world: This is what we do when we learn something from reading about it.

Understanding of this first kind is recursive: We understand one thing by understanding another. It is "recursive", because we are understanding one thing in terms of another *that must already be understood*: We understand in terms of something understood.

But all recursion requires a base case in order to avoid an infinite regress. Consequently, the second way of understanding—understanding something directly—is to understand a domain in terms of itself, to get used to it. This is the fundamental kind—the base case—of understanding.

In general, we understand one domain—call it a *syntactic* domain ('SYN_1')—*indirectly* by *interpreting it* in terms of a (different) domain: a *semantic* domain ('SEM_1'). This kind of understanding is "indirect", because we understand SYN_1 by looking *elsewhere*, namely,

at SEM_1. But for this process of interpretation to result in real understanding, SEM_1 must be *antecedently understood.* How? In the same way: by considering it as a *syntactic* domain (rename it 'SYN_2') interpreted in terms of *yet another semantic* domain, which *also* must be antecedently understood. And so on. But, in order not to make this sequence of interpretive processes go on *ad infinitum*, there must be a base case: a domain that is understood directly, i.e., in terms of itself (i.e., not "antecedently"). Such direct understanding is syntactic understanding; i.e., it is understanding in terms of the relations among the marks of the system itself (Rapaport, 1986). Syntactic understanding may be related to what Piccinini (2008, p. 214) called "internal semantics"—the interpretation of an instruction in terms of "what its execution accomplishes *within* the computer." And it may be related to the kind of understanding described in Eco 1988, in which words and sentences are understood in terms of inferential (and other) relations that they have with "contextual" "encyclopedias" of other words and sentences—i.e., syntactically and holistically (cf. Rapaport, 2002).[31]

4. SYNTACTIC SEMANTICS VS. FETZER'S THESIS

Syntactic semantics implies that syntax suffices for semantic cognition, that (therefore) cognition is computable, and that (therefore) computers are capable of thinking.

In a series of papers, including Fetzer (2011) in this journal, James H. Fetzer has claimed that syntax does *not* suffice for semantic cognition, that cognition is *not* computable, and that computers are *not* capable of thinking.

More precisely, Fetzer's thesis is that computers differ from cognitive agents in three ways—statically (or symbolically), dynamically (or algorithmically), and affectively (or emotionally)—and that simulation is not "the real thing."

In the rest of this essay, I will try to show why I think that Fetzer is mistaken on all these points.

5. FETZER'S "STATIC" DIFFERENCE

In the forward to his 2001 collection, *Computers and Cognition*, as well as in his presentation at the 2010 North American Conference on Computing and Philosophy, Fetzer argued that "Computers are mark-manipulating systems, **minds are not**" on the following grounds of "static difference" (Fetzer, 2001, p. xiii, emphasis added):

Premise 1: Computers manipulate marks on the basis of their shapes, sizes, and relative locations.

Premise 2:

[a] These shapes, sizes, and relative locations exert causal influence upon computers.

[b] But do not stand for anything for those systems.

Premise 3: Minds operate by utilizing signs that stand for other things in some respect or other for them as sign-using (or "semiotic") systems.

Conclusion 1: Computers are not semiotic (or sign-using) systems.

Conclusion 2: Computers are not the possessors of minds.

I disagree with all of the boldfaced phrases in Fetzer's static-difference argument. Before saying why, note that here—and in his arguments to follow—Fetzer consistently uses declaratives that appear to describe current-day computers: They *do* not do certain things, *are* not affected in certain ways, or *do* not have certain properties. But he really should be using modals that specify what he believes computers *cannot* do, be affected by, or have. Consider premise (2b), that the marks that computers manipulate "do not" stand for anything for those computers. Note that Fetzer's locution allows for the possibility that, although the marks *do* not stand for anything for the computer, they *could* do so. Insofar as they could, such machines *might* be capable of thinking. So Fetzer should have made the stronger claim that they "*could not*

stand for anything". But then he'd be wrong, as I shall argue.[32]

5.1. Mark Manipulation

What is a "mark"? Fetzer (2011) does not define the term; he has used it many times before (e.g., Fetzer, 1994, 1998), also without definition, but occasionally with some clarification. It seems to be a term designed to be neutral with respect to such semiotic terms as 'sign', 'symbol', etc., perhaps even 'term' itself. It seems to have the essential characteristics of being the kind of entity that syntax is concerned with (i.e., marks are the things such that syntax is the study of *their* properties and relations) and of not having an intrinsic meaning. Many logicians use the terms 'sign' or 'symbol' in this way, despite the fact that many semioticians use 'sign' and 'symbol' in such a way that all signs and symbols are meaningful. So 'mark' is intended to be used much as 'sign' or 'symbol' might be used if the sign or symbol were stripped of any attendant meaning. Elsewhere (Rapaport, 1995), I have used 'mark' to mean "(perhaps) physical inscriptions or sounds, that have only some very minimal features such as having distinguished, relatively unchanging shapes capable of being recognized when encountered again"; they have "no intrinsic meaning. But such [marks] *get* meaning the more they are used—the more roles they play in providing meaning to *other*" marks.

Fetzer's Static Premise 1 is true (i.e., I agree with it!): "computers manipulate marks on the basis of their shapes, sizes, and relative locations". But it is not the whole truth: They also manipulate marks on the basis of other, non-spatial relations of those marks to other marks, i.e., on the basis of the marks' syntax in the wide sense in which I am using that term.[33] Fetzer can safely add this to his theory.

I also agree that this manipulation (or processing) is *not* (necessarily) independent of the meanings of these marks. But my agreement follows, not from *Fetzer's* notion of meaning, but from the principle of "syntacticism" (§3.2): If *some* of the marks represent (or are) meanings of some of the *other* marks, then mark manipulation on the basis of size, shape, location, *and relations to other marks* includes manipulation on the basis of meaning!

However, this *is* independent of *external* reference. More precisely, it is independent of the actual, external referents of the marks—the objects that the marks stand for, if any (§3.1)—in this way: Any relationship between a mark and an external meaning of that mark is represented by an internal (hence syntactic) relation between that mark and another mark that is the internal representative of the external referent. The latter mark is the output of a sensory transducer; in Kantian terms, it is an internal phenomenon that represents an external noumenon. This is the upshot of internalism (§3.1).

In this way, such marks *can* stand for something *for the computer*. Computers are, indeed, "string-manipulating" systems (Fetzer 1998: 374). But they are more than "mere" string-manipulating systems, for meaning can arise from (appropriate) combinations of (appropriate) strings.

5.2. Computers and Semiotic Systems

5.2.1. Fetzer's Argument that Computers are not Semiotic Systems

Fetzer's static-difference argument claims that computers are *not* semiotic systems. In an earlier argument to the same conclusion, Fetzer (1998), following Peirce, says that a semiotic system consists of something (*S*) being a sign of something (*x*) for somebody (*z*), where:

- Thing *x* "grounds" sign *S*
- Thing *x* stands in a relation of "interpretant (with respect to a context)" to sign-user *z*.[34]
- And sign *S* stands in a "causation" relation with sign-user *z*.

This constitutes a "semiotic triangle" whose vertices are *S*, *x*, and *z* (Fetzer, 1998, p. 384, Figure 1). This cries out for clarification:

1. What is the causation relation between sign-user *z* and sign *S*? Does one cause the other? Or is it merely that they are causally—i.e., physically—related?
2. What is the grounding relation between sign *S* and the thing *x* that *S* stands for (i.e., that *S* is a sign of)? If sign *S* is grounded by what it stands for (i.e., *x*), then is the relation of *being grounded by* the same as the relation of *standing for*?
3. And, if sign *S* stands for thing *x for* sign-user *z*, then perhaps this semiotic triangle should really be a semiotic "quadrilateral", with *four* vertices: sign *S*, user *z*, thing *x*, *and an interpretant I*, where the four sides of the quadrilateral are:
   i. User *z* "causes" sign *S*,
   ii. Thing *x* "grounds" sign *S*,
   iii. Interpretant *I* is "for" user *z*,
   iv. And I stands for thing *x*.

There is also a "diagonal" in this quadrilateral: *I* "facilitates" or "mediates" *S*. (A better way to think of it, however, is that the two sides and the diagonal that all intersect at *I* represent the 4-place relation "interpretant *I* mediates sign *S* standing for object *x* for user *z*" (Rapaport, 1998, p. 410, Figure 2).

By contrast, according to Fetzer, a similar semiotic "triangle" for "input-output" systems, such as computers, *lacks* a relationship between what plays the role of sign *S* and what plays the role of thing *x* (it is only a 2-sided "triangle"; see Fetzer, 1998, p. 384, Figure 2). More precisely, for such input-output systems, Fetzer says that we have an input *i* (instead of a sign *S*), a computer *C* (instead of a sign-user *z*), and an output *o* (instead of thing *x*), where there is still a causation relation between computer *C* and input *i*, and an interpretant relation between *C* and output *o*, but—significantly—no grounding relation between input *i* and output *o*.

Again, we may raise some concerns:

1. Computers are much more than mere input-output systems, because there is always an algorithm that mediates between the computer's input and its output.
2. The marks by means of which computers operate include more than merely the external input; there are usually *stored* marks representing what might be called "background knowledge" (or "prior knowledge")—perhaps "(internal) context" would not be an inappropriate characterization: Where are these in this two-sided triangle?
3. And what about the "stands for" relation? Surely, what the input marks stand for (and surely they stand for something) is not necessarily the *output*.
4. Finally, what does it mean for a sign *S* or an input mark *i* to stand for something *x* (or *o*) yet not be "grounded" by *x*?

Fetzer's chief complaint about computers is not merely that they causally manipulate marks (premise 1) but that such causal manipulation is *all* that they can do. Hence, because such merely causal manipulation requires no mediation between input and output, computers are not semiotic systems. By contrast, semiosis does require such mediation; according to Peirce, it is a ternary relation:

"I define a Sign as anything which is so determined by something else, called its Object, and so determines an effect upon a person, which effect I call its Interpretant, that the latter is thereby mediately determined by the former. My insertion of "upon a person" is a sop to Cerberus, because I despair of making my own broader conception understood (Peirce, 1908/1977, pp. 80-81)."

The fact that "upon a person" is a "sop to Cerberus" suggests that the effect need *not* be "upon a person"; it *could*, thus, be "upon a computer". That is, the mark-user need not be human (cf. Eco, 1979, p. 15: "It is possible to interpret Peirce's definition in a non-anthropomorphic way…").[35]

5.2.2. Peirce and Computation

Given Fetzer's reliance on Peirce's version of semiotics (which I focus on rather than, say, on Saussure's version of semiotics, because

Peirce is whom Fetzer focuses on), it is worth noting that Peirce had some sympathy for—and certainly an active interest in—computation, especially its syntactic aspect:

"The secret of all reasoning machines is after all very simple. It is that whatever relation among the objects reasoned about is destined to be the hinge of a ratiocination, that same general relation must be capable of being introduced between certain parts of the machine. …When we perform a reasoning in our unaided minds we do substantially the same thing, that is to say, we construct an image in our fancy under certain general conditions, and observe the result." (Peirce, 1887, p. 168)[36]

There is even an anticipation of at least one of Turing's insights (cf. Appendix):[37]

"[T]he capacity of a machine has absolute limitations; it has been contrived to do a certain thing, and it can do nothing else. For instance, the logical machines that have thus far been devised can deal with but a limited number of different letters. The unaided mind is also limited in this as in other respects; but ***the mind working with a pencil and plenty of paper has no such limitation****. It presses on and on, and whatever limits can be assigned to its capacity to-day, may be over-stepped to-morrow."* (Peirce, 1887, p. 169, emphasis added)

Furthermore, Peirce had views on the relation of syntax to semantics that are, arguably, sympathetic to mind. As Brown (2002, p. 20) observes,[38]

"Thus Peirce, in contrast to Searle, would not…allow any separation between syntax and semantics, in the following respect. He would claim that what Searle is terming "syntactic rules" partake of what Searle would consider semantic characteristics, and, generally, that such rules must so partake. However, if those rules were simple enough so that pure deduction, i.e., thinking of the first type of thirdness, was all that was required, then a machine could indeed duplicate such 'routine operations.'" (Peirce, 1992d, p. 43). In this simple sense, for Peirce, a digital computer has 'mind' or 'understanding.'

Thus, I find Fetzer's analysis vague and unconvincing at best. We need to bring some clarity to this. First, consider what Peirce actually says:

"A sign, or ***representamen****, is something which stands to somebody for something in some respect or capacity. It addresses somebody, that is, creates in the mind of that person an equivalent sign, or perhaps a more developed sign. That sign which it creates I call the interpretant of the first sign. The sign stands for something, its object. It stands for that object, not in all respects, but in reference to a sort of idea, which I have sometimes called the ground of the representamen."* (CP 2.228)

"[B]y 'semiosis' I mean…an action, or influence, which is, or involves, a cooperation of ***three*** *subjects, such as a sign, its object, and its interpretant, this tri-relative influence not being in any way resolvable into actions between pairs."* (CP 5.484)

By 'representamen', Peirce means "sign", but I think that we may also say 'mark'. So, the Peircean analysis consists of:

1. A mark (or sign, or representamen):

   A "mark" is, roughly, an uninterpreted sign—if that's not an oxymoron: If a sign consists of a mark (signifier) plus an interpretation (signified), then it must be marks that are interpreted; after all, signs (presumably) wear their interpretations on their sleeves, so to speak. Although many semioticians think that it is anathema to try to separate a sign into its two components, that is precisely what "formality" is

about in such disciplines as "formal logic", "formal semantics", and so on: Formality is the separation of mark from meaning; it is the focus on the form or shape of marks. (cf. B.C. Smith, 2010, §4b)

2. A mark user (or interpreter):

"[A] Sign has an Object and an Interpretant, the latter being that which the Sign produces in the Quasi-mind that is the Interpreter by determining the latter to a feeling, to an exertion, or to a Sign, which determination is the Interpretant" (Peirce CP 4.536)

3. An object that the mark stands for (or is a sign of);
4. An interpretant in the mark-user's mind, which is also a mark.

The interpretant is the mark-user's idea (or concept, or mental representative)[39]—but an idea or representative of what? Of the mark? Or of the object that the mark stands for? Peirce says that the interpretant's *immediate* cause is the mark and its *mediate* cause is the object.[40] Moreover, the interpretant is also a representamen, namely, a sign of the original sign.[41]

But there is also another item: Presumably, there is a causal process that produces the interpretant in the mark-user's mind. This might be an automatic or unconscious process, as when a mental image is produced as the end product of the process of visual perception. Or it might be a conscious process, as when the mark-user reads a word (a mark) and consciously figures out what the word might mean, resulting in an interpretant (as in "deliberate" contextual vocabulary acquisition; cf. Rapaport & Kibby, 2007, 2010).

One standard term for this process (i.e., for this relation between mark and interpretant) is 'interpretation'. The word 'interpretation', however, is ambiguous. In one sense, it is a functional[42] *relation* from a mark to a thing that the mark "stands for" or "means". In another sense, it is the end product or *output* of such a functional relation. For present purposes, since we already have Peirce's term 'interpretant' for the *output*, let us restrict 'interpretation' to refer to the *process*. (What is the relationship between my *binary* relation of interpretation and Peirce's *ternary* relation of semiosis? (see the quote from Peirce CP 5.484).[43] "Interpretation" in my sense is only *part* of "semiosis" in Peirce's sense; it is a relation between only two of the three parameters of Peircean semiosis, namely, the sign and its interpretant.)

With this by way of background, I will now present three arguments that computers *are* semiotic systems.[44] Recall (§5.2.1) that Peirce's view of semiotics does not require the mark-user to be human!

5.2.3. Computers Are Semiotic Systems I: Incardona's Argument

The first argument, due to Lorenzo Incardona (personal communication), consists of three premises and a conclusion.

1. Something is a semiotic system if and only if it carries out a process that mediates between a mark and an interpretant of that mark:
   - The essential characteristic of a semiotic system is its ternary nature; it applies (i) a mediating or interpretive process to (ii) a mark, resulting in (iii) an interpretant of that mark (and indirectly of the mark's object). A semiotic system interprets marks.[45]
   - However, there must be a fourth component: After all, *whose* interpretant is it that belongs to a given mark? This is just another way of asking about "the" meaning of a word. What is "the" interpretant of 'gold'? Is it what *I* think gold is? What *experts* think it is (Putnam, 1975)? It is better to speak of "a" meaning "for" a word than "the" meaning "of" a word (Rapaport, 2005a). Surely, there are many meanings for marks but, in semiosis, the only one that matters is that of (iv) the interpreter, i.e., the executor of the interpretive process (i, above) that

mediates the mark (ii, above) and the (executor's) interpretant (iii, above).

2. Algorithms describe processes that mediate between inputs and outputs.
   - An algorithm is a *static text* (or an abstract, mathematical entity) that describes a *dynamic process*. That process can be thought of as the algorithm being executed.
   - The output of an algorithm stands (by definition) in a functional relation to its input. It describes (or can describe) a meaningful relation between the input mark and the output mark. So, the output of an algorithm is an interpretant of its input. This is a causal relationship, but it is not *merely* a causal (or stimulus-response, "behavioral") correlation between input and output: In the case of computable functions (i.e., of a computable interpretation process), there is a mediating process, namely, an algorithm. Consequently, the input-output relation is grounded—or mediated—by that algorithm, i.e., the mechanism that converts the input into the output, i.e., the interpretation.[46]
3. Clearly, computers are algorithm machines.[47]
   - That's what computers do: They execute algorithms, converting inputs into outputs, i.e., interpreting the inputs as the outputs according to an algorithm. But a computer is (usually) *not a mere* input-output system. The (typical) algorithm for converting the input to the output (i.e., for *interpreting*) the input as the output) is not a mere table-lookup. First, there is internally stored data that can be used or consulted in the conversion process; this can be thought of as a "context of interpretation". Second, there are numerous operations that must be performed on the input together with the stored data in order to produce the output, so it is a dynamic process. Moreover, the stored data can be modified by the input or the operations so that the system can "learn" and thereby change its interpretation of a given input.
4. Therefore, computers *are* semiotic systems.

5.2.4. Computers Are Semiotic Systems II: Goldfain's Argument from Mathematical Cognition

Does a calculator that computes greatest common divisors (GCDs) understand what it is doing? I think that almost everyone, including both Fetzer and I, would agree that it does not. But *could* a *computer* that computes GCDs understand what it is doing? I am certain that Fetzer would say 'no'; but I—along with Albert Goldfain (2008; Shapiro, Rapaport et al., 2007)—say 'yes': Yes, it could, as long as it had enough background, or contextual, or supporting information: A computer with a full-blown theory of mathematics at, say, the level of an algebra student learning GCDs, together with the ability to explain its reasoning when answering a question, could understand GCDs as well as the student. (Perhaps it could understand better than the student, if the student lacks the ability to fully explain his or her reasoning.)

Calculating becomes understanding if embedded in a larger framework linking the calculation to other concepts (not unlike the semantic contributions of Semantic Web annotations, which also serve as an embedding framework). So a computer *could* know, do, and understand mathematics, *if* suitably programmed with all the background information required for knowing, doing, and understanding mathematics. A computer or a human who can calculate GCDs by executing an algorithm does not (necessarily) thereby understand GCDs. But a computer or a human who has been taught (or programmed) all the relevant mathematical definitions can understand GCDs (for further discussion, see Rapaport, 1988, 1990, 2006, 2011).

Goldfain (personal communication) has offered the following argument that marks can stand for something for a computer:

1. The natural numbers that a cognitive agent refers to are denoted by a sequence of marks that are unique to the agent. These marks exemplify (or implement, Rapaport, 1999, 2005b) a finite, initial segment of the natural-number structure (i.e., 0, 1, 2, 3, …, up to some number *n*).
2. Such a finite, initial segment can be generated by a computational cognitive agent (e.g., a computer, suitably programmed) via perception and action in the world during an act of counting (e.g., using the Lisp programming language's "gensym" function) (for details see Goldfain, 2008). Thus, there would be a history of how these marks came to signify something for the agent (e.g., the computer).
3. These marks (e.g., b4532, b182, b9000, …) have no meaning for another cognitive agent (e.g., a human user of the computer) who lacks access to their ordering.
4. Such private marks (called 'numerons' by cognitive scientists studying mathematical cognition) are associable with publicly meaningful marks (called 'numerlogs').[48] For example, 'b4532' might denote the same number as the Hindu-Arabic numeral '1', 'b182' might denote the same number as '2', etc.
5. A computational cognitive agent (e.g., a computer) can do mathematics solely on the basis of its numerons (see Goldfain, 2008, for a detailed demonstration of this; cf. Shapiro, Rapaport et al., 2007).
6. Therefore, *these marks stand for something for the computer* (i.e., for the agent). Moreover, we can check the mathematics, because of premise 4.
7. Thus, such a computer would be a semiotic system.

5.2.5. Computers Are Semiotic Systems III: Argument from Embedding in the World

Besides being able to understand mathematics in this way, computers can also understand other, conventional activities. Fetzer (2008) gives the following example of something that a semiotic system can do but that, he claims, a computer cannot:

> *"A red light at an intersection…stands for* ***applying the brakes and coming to a complete halt, only proceeding when the light turns green****, for those who know 'the rules of the road.'"* (emphasis added)

As Fetzer conceives it, the crucial difference between a semiotic system and a computer is that the former, but not the latter, can use a mark as something that stands for something (else) *for itself* (cf. Fetzer, 1998). In that case, we need to ask whether such a red light can stand for "applying the brakes", etc. *for a computer*.

It could, *if* it has those rules *stored* (memorized) in a knowledge base (its mind). But merely storing, and even being able to access and reason about, this information is not enough: IBM's *Jeopardy*-winning Watson computer can do that, but no one would claim that such merely stored information is understood, i.e., stands for anything for Watson itself. At least one thing more is needed for a red light to stand for something for the computer itself: The computer must *use* those rules to drive a vehicle. But there *are* such computers (or computerized vehicles), namely, those that have successfully participated in the DARPA Grand Challenge autonomous vehicle competitions (for a relevant discussion, see Parisien & Thagard, 2008). I conclude that (such) computers can be semiotic systems.

Some might argue that such embedding in a world is not computable. There are two reasons to think that it is. First, all of the autonomous vehicles must internally store the external input in order to compute with it. Thus, the syntactic-semantic principle of internalism explains how the embedding is computable. Second, the vehicles' behaviors are the output of computable (indeed, computational) processes (I discuss this further in §6.2).[49]

6. FETZER'S "DYNAMIC" DIFFERENCE

Fetzer also argues that "computers are governed by algorithms, but **minds are not**", on the following grounds (Fetzer, 2001, p. xv, emphasis added):

Premise 1: Computers are governed by programs, *which are causal models of algorithms.*

Premise 2: Algorithms are effective decision procedures for arriving at definite solutions to problems in a finite number of steps.

Premise 3: *Most human thought processes, including dreams, daydreams, and ordinary thinking, are not procedures for arriving at solutions to problems in a finite number of steps.*

Conclusion 1: Most human thought processes are not governed by programs as causal models of algorithms.

Conclusion 2: *Minds are not computers.*

Once again, I disagree with the boldfaced claims. The italicized claims are ones that are subtly misleading; below, I will explain why. I prefer not to speak as Fetzer does.

First, let me point to a "red herring": Fetzer (2008) said that "if thinking is computing *and computing is thinking* and if computing is algorithmic, then thinking is algorithmic, but it isn't" (my emphasis). The second conjunct is false; fortunately (for Fetzer), it is also irrelevant: A computer executing a non-cognitive program (e.g., an operating system) *is* computing but is *not* thinking. (Of course, this depends on whether you hold that an operating system is a *non*-cognitive, computer program. Once upon a time, it was *humans* who operated computers. Arguably, operating systems, insofar as they do a task once reserved for humans, *are* doing a cognitive task. Alternatively, what the humans who operated computers were doing was a task requiring no "intelligence" at all.[50] Cf. my remark about Turing machines as AI programs, in §2; on performing intelligent tasks without intelligence, see Dennett, 2009.)

6.1. Algorithms

Premise 2 is consistent with the way I characterized 'algorithm' in §2, so I am willing to accept it. Yet I think that algorithms, so understood, are the wrong entity for this discussion. Instead, we need to relax some of the constraints and to embrace a more general notion of "procedure" (Shapiro, 2001).

(1) In particular, procedures (as I am using the term here) are like algorithms, but they do not necessarily halt of their own accord; they may continue executing until purposely (or accidentally) halted by an outside circumstance (cf. Knuth's, 1973 notion of a (non-finite) "computational method"; see the Appendix). For instance, an automatic teller machine or an online airline reservation system should not halt unless turned off by an administrator (cf. Wegner, 1997).

(2) Also, procedures (as I am using the term) need not yield "correct" output (goal *G* need not be completely or perfectly accomplished). Consider a computer programmed for playing chess. Even IBM's celebrated, chess champion Deep Blue will not win or draw every game, even though chess is a game of perfect (finite) information (like Tic Tac Toe) in which, mathematically, though not practically, it is knowable whether any given player, if playing perfectly, will win, lose, or draw, because the "game tree" (the tree of all possible moves and replies) can *in principle* be completely written down and examined. But because of the practical limitations (the tree would take too much space and time to actually write down; cf. Zobrist, 2000, p. 367), the computer chess program will occasionally give the "wrong" output. But it is not behaving any differently from the algorithmic way it would behave if it gave the correct answer. Sometimes, such programs are said to be based on "heuristics" instead of "algorithms", where a *heuristic for problem P* is an *algorithm* for some other problem

P′, where the solution to *P′* is "near enough" to a solution to *P* (Rapaport, 1998, §2; cf. Simon's, 1956, notion of "satisficing"). Another kind of procedure that does not necessarily yield "correct" output is a "trial and error" (or Putnam-Gold) machine that continually generates "guesses" as to the correct output and is allowed to "change its mind", with its "final answer" being its *last* output, not its *first* output (Kugel, 2002).

(3) Finally, procedures (as I am using the term) include "interactive" programs, which are best modeled, not as Turing machines, but as Turing's "oracle" machines (a generalization of the notion of Turing machine) (see Wegner, 1997, Soare, 2009).

In order for computational cognitive science's working hypothesis to be correct, the computation of cognition will have to be done by "algorithms" that are procedures in one or more of these senses.

6.2. Are Dreams Algorithms?

Fetzer (1998) argues that dreams are not algorithms and that ordinary, stream-of-consciousness thinking is not "algorithmic". I am willing to agree, up to a point, with Fetzer's Dynamic Premise 3: *Some* human thought processes may indeed *not be* algorithms (or even procedures more generally). But that is not the real issue. The real issue is this: Could there be algorithms (or procedures) that *produce* dreams, stream-of-consciousness thinking, or other mental states or procedures, including those that might not *themselves* be algorithms (or procedures)?

The difference between an entity *being* computable and being *produced by* a computable process (i.e., being the *output* of an algorithm) can be clarified by considering two ways in which images can be considered computable entities. An image could be *implemented by* an array of pixels; this is the normal way in which images are stored in—and processed by—computers. Such an image is a computable, discrete data structure reducible to arrays of 0s and 1s. Alternatively, an image could be *produced by* a computational process that drives something like a flatbed plotter (Phillips, 2011) or produces a painting (as with Harold Cohen's AARON; see McCorduck, 1990). Such an image is *not* a discrete entity—it is, in fact, continuous; it is *not* reducible to arrays of 0s and 1s. Similarly, dreams need not themselves *be* algorithms in order to be producible *by* algorithms. (The same could, perhaps, be said of pains and other qualia: They might not themselves *be* algorithmically describable states, but they might be the *outputs* of algorithmic(ally describable) processes.)

What are dreams? In fact, no one knows, though there are many rival theories. Without some scientific agreement on what dreams are, it is difficult to see how one might say that they are—or are not—algorithmic or producible algorithmically. But suppose, as at least one standard view has it, that dreams are our interpretations of random neuron firings during sleep (perhaps occurring during the transfer of memories from short- to long-term memory),[51] interpreted as if they were due to external causes. Suppose also that *non*-dream neuron firings are computable. There are many reasons to think that they are; after all, the working assumption of computational cognitive science may have been challenged, but has not yet been refuted (*pace* Piccinini, 2005, 2007–remember, I am supposing that neuron firings are comput*able*, not necessarily computa*tional*; thus, Piccinini's arguments that neuron firings are *not* computational are irrelevant—and *pace* those cited in note 8). In that case, the neuron firings that constitute dreams would also be computable.

What about stream-of-consciousness thinking? That might be computable, too, by means of spreading activation in a semantic network, apparently randomly associating one thought to another.

In fact, computational cognitive scientists have proposed computational theories of both dreaming and stream-of-consciousness thinking! (see Mueller, 1990; Edelman, 2008b; Mann, 2010). It is not a matter of whether these scientists are right and Fetzer is wrong, or the other way around. Rather, the burden of proof is on

Fetzer to say why he thinks that these proposed computational theories fail irreparably.

The important point is that whether a mental state or process is computable is at least an empirical question. Anti-computationalists must be wary of committing what many AI researchers think of as the Hubert Dreyfus fallacy: One philosopher's idea of a non-computable task may be just another computer scientist's research project. Put another way, what no one has *yet* written a computer program for is not thereby necessarily non-computable.

6.3. Are Minds Computers?

Fetzer's Dynamic Conclusion 2 is another claim that must be handled carefully: Maybe minds are computers; maybe they aren't. The more common formulation is that minds are *programs* and that *brains* are computers (cf. Searle, 1990; Piccinini, 2010). But I think that there is a better way to express the relationship than either of these slogans: A mind is a virtual machine, computationally implemented in some medium.

Roughly, a virtual machine is a computational implementation of a real machine; the virtual machine is executed as a process running on another (real or virtual) machine. For instance, there is (or was, at the time of writing) an "app" for Android smartphones that implements (i.e., simulates, perhaps emulates; see §8) a Nintendo Game Boy. Game Boy videogames can be downloaded and played on this "virtual" Game Boy "machine" (for a more complex example, of a virtual machine running on another virtual machine, see Rapaport, 2005b, §3.3).

Thus, the human mind is a virtual machine computationally implemented in the nervous system, and a robot mind would be a virtual machine computationally implemented in a computer. Such minds consist of states and processes produced by the behavior of the brain or computer that implements them (for discussion of virtual machines and this point of view, see Rapaport, 2005b, §3.2; Hofstadter, 2007; Edelman, 2008b; Pollock, 2008).

7. FETZER'S "AFFECTIVE" DIFFERENCE

Fetzer also argues that "Mental thought transitions are affected by emotions, attitudes, and histories, **but computers are not**", on the following grounds (Fetzer, 2008, 2010, emphasis added):

Premise 1: Computers are governed by programs, *which are causal models of algorithms.*

Premise 2: Algorithms are effective decisions, **which are not affected by emotions, attitudes, or histories.**

Premise 3: Mental thought transitions are affected by values of variables **that do not affect computers.**

Conclusion 1: The processes controlling mental thought transitions are fundamentally different than those that control computer procedures.

Conclusion 2: *Minds are not computers.*

Once again, I disagree with the boldfaced claims and find the italicized ones subtly misleading, some for reasons already mentioned.

Before proceeding, it will be useful to rehearse (and critique) the definitions of some of Fetzer's technical terms, especially because he uses some of them in slightly non-standard ways. On Fetzer's view:

- The *intension* of expression $E =_{\text{def}}$ the conditions that need to be satisfied for something to be an *E*. (This term is not usually limited to noun phrases in the way that Fetzer seems to limit it ("to be an *E*"), but this is a minor point.)
- The *extension* of expression $E =_{\text{def}}$ the class of all things that satisfy *E*'s intension. (A more standard definition would avoid defining 'extension' in terms of 'intension'; rather, the extension of an expression would be the class of all (existing) things to which the expression *E* is applied by (native) speakers of the language).

- The *denotation* of expression *E* for agent *A* $=_{def}$ the subset of *E*'s extension that *A* comes into contact with. (This notion may be useful, but 'denotation' is more often a mere synonym of 'extension' or 'referent'.)
- The *connotation* of expression *E* for agent *A* $=_{def}$ *A*'s attitudes and emotions in response to *A*'s interaction with *E*'s denotation for *A* (Again, a useful idea, but not the usual use of the term, which is more often a way of characterizing the other concepts that are closely related to *E* (perhaps in some agent *A*'s mind, or just the properties associated with (things called) *E*. Both 'denotation' and 'connotation' in their modern uses were introduced by Mill 1843, the former being synonymous with 'extension' and the latter referring to implied properties of the item denoted).

Fetzer then identifies the "meaning" of *E* for *A* as *E*'s denotation and connotation for *A*.

Contra Fetzer's Affective Premises 2 and 3, programs *can* be based on idiosyncratic emotions, attitudes, and histories: Karen Ehrlich and I (along with other students and colleagues) have developed and implemented a computational theory of contextual vocabulary acquisition (Ehrlich, 1995; Rapaport & Ehrlich, 2000; Rapaport, 2003b, 2005a; Rapaport & Kibby, 2007, 2010). Our system learns (or "acquires") a meaning for an unfamiliar or unknown word from the word's textual context integrated with the reader's prior beliefs. These prior beliefs (more usually called 'prior knowledge' in the reading-education literature), in turn, can—and often do—include idiosyncratic "denotations" and "connotations" (in Fetzer's senses), emotions, attitudes, and histories. In fact, contrary to what some reading educators assert (cf. Ames, 1966; Dulin, 1970), a meaning for a word can*not* be determined *solely* from its textual context. The reader's prior beliefs are essential (Rapaport, 2003b; Rapaport & Kibby, 2010). And, clearly, the meaning that one reader figures out or attributes to a word will differ from that of another reader to the extent that their prior beliefs differ.

Furthermore, several cognitive scientists have developed computational theories of affect and emotion, showing that emotions, attitudes, and histories *can* affect computers that model them (Simon, 1967; Sloman & Croucher, 1981; Wright et al., 1996; Sloman, 2004, 2009; Picard, 1997; Thagard, 2006,among others).

Once again, the burden of proof is on Fetzer.

8. SIMULATION

I close with a discussion of "the matter of simulation". Fetzer argues that "Digital machines can nevertheless simulate thought processes and other diverse forms of human behavior", on the following grounds (Fetzer, 2001, p. xvii, 2008, 2010, emphasis added):

Premise 1: Computer programmers and those who design the systems that they control can increase their performance capabilities, making them better and better simulations.

Premise 2: Their performance capabilities may be closer and closer approximations to the performance capabilities of human beings **without turning them into thinking things.**

Premise 3: Indeed, the static, dynamic, and affective differences that distinguish computer performance from human performance preclude those systems from being thinking things.

Conclusion: Although the performance capabilities of digital machines can become better and better approximations of human behavior, **they are still not thinking things.**

As before, I disagree with the boldfaced claims. But, again, we must clarify Fetzer's somewhat non-standard use of terms.

Computer scientists occasionally distinguish between a "simulation" and an "emulation", though the terminology is not fixed. In the *Encyclopedia of Computer Science*, Paul F. Roth (1983) says that *x simulates y* means that *x* is a model of some real or imagined

system *y*, and we experiment with *x* in order to understand *y*. (Compare our earlier discussion of understanding one thing in terms of another, §3.3.) Typically, *x* might be a computer program, and *y* might be some real-world situation. In an extreme case, *x* simulates *y* if and only if *x* and *y* have the same input-output behavior.

And in another article in that *Encyclopedia*, Stanley Habib (1983) says that *x emulates y* means that either: (a) computer *x* "interprets and executes" computer *y*'s "instruction set" by implementing *y*'s operation codes in *x*'s hardware—i.e., hardware *y* is implemented as a virtual machine on *x*—or (b) software feature *x* "simulates"(!) "hardware feature" *y*, doing what *y* does exactly (so to speak) as *y* does it. Roughly, *x* emulates *y* if and only if *x* and *y* not only have the same input-output behavior, but also use the same algorithms and data structures.

This suggests that there is a continuum or spectrum, with "pure" simulation at one end (input-output–equivalent behavior) and "pure" emulation at the other end (behavior that is equivalent with respect to input-output, all algorithms in full detail, and all data structures). So, perhaps there is no real distinction between simulation and emulation except for the degree of faithfulness to what is being simulated or emulated.

In contrast, Fetzer uses a much simplified version of this for his terminology (Fetzer, 1990, 2001, 2011):

- System *x simulates* system $y =_{def} x$ and *y* have the same input-output behavior.
- System *x replicates* system $y =_{def} x$ simulates *y* by the same or similar processes.
- System *x emulates* system $y =_{def} x$ replicates *y*, and *x* and *y* "are composed of the same kind of stuff".

At least the latter two are non-standard definitions and raise many questions. For instance, how many processes must be "similar", and how "similar" must they be, before we can say that one system replicates another? Or consider (1) a Turing machine (a single-purpose, or dedicated, computer) that computes GCDs and (2) a universal Turing machine (a multiple-purpose, or stored-program computer) that can be programmed to compute GCDs using exactly the same program as is encoded in the machine table of the former Turing machine. Does the latter emulate the former? Does the former emulate the latter? Do two copies of the former emulate each other? Nevertheless, Fetzer's terminology makes some useful distinctions, and, here, I will use these terms as Fetzer does.

The English word 'simulation' (however defined) has a sense (a "connotation"?) of "imitation" or "unreal": A simulation of a hurricane is not a real hurricane. A simulation of digestion is not real digestion. And I agree that a computer that simulates (in Fetzer's sense) some process *P* is not *necessarily* "really" doing *P*. But what, exactly, is the difference? A computer simulation (or even a replication) of the daily operations of a bank is not thereby the daily operations of a (real) bank. But I *can* do my banking online; simulations *can* be used as if they were real, as long as the (syntactic) simulations have causal impact on me (Goldfain, personal communication, 2011). And although computer simulations of hurricanes don't get real people wet (they are, after all, not emulations in Fetzer's sense), they *could* get *simulated* people *simulatedly* wet (Shapiro & Rapaport, 1991; Rapaport, 2005b):

> *"[A] simulated hurricane would feel very real, and indeed can prove fatal,* ***to a person who happens to reside in the same simulation****."* (Edelman, 2011, p. 2, note 3, emphasis added)

The "person", of course, would have to be a *simulated* person. And it well might be that the way that the simulated hurricane would "feel" to that simulated person would differ from the way that a real hurricane feels to a real person (on the meaningfulness of that comparison, see Strawson, 2010, pp. 218-219).

> *"Paul Harris [2000] found that even two-year-olds will tell you that if an imaginary teddy [bear] is drinking imaginary tea, then if he spills*

it the imaginary floor will require imaginary mopping-up." (Gopnik, 2009, p. 29)

This is a matter of the "scope" of the simulation: Are people within the scope of the hurricane simulation or not? If they are not, then the simulation won't get them wet. If they are—i.e., if they are simulated, too—then it will.

It should also be noted that sometimes things like *simulated* hurricanes *can* do something analogous to getting *real* people wet: Children "can have real emotional reactions to entirely imaginary scenarios" (Gopnik, 2009, p. 31), as, of course, can adults, as anyone who has wept at the movies can testify (cf. Schneider, 2009).

But there are cases where a simulation *is* the real thing. For example:

- A scale model of a scale model of the Statue of Liberty *is* a scale model of the Statue of Liberty.
- A Xerox copy of a document *is* that document, at least for purposes of reading it and even for some legal purposes.
- A PDF version of a document *is* that document.

More specifically, a computer that simulates an "informational process" *is* thereby actually doing that informational process, because a computer simulation of information *is* information:

- A computer simulation of a picture *is* a picture, hence the success of digital "photography".
- A computer simulation of language *is* language. Indeed, as William A. Woods (2010, p. 605, emphasis added) said:

"[L]anguage is fundamentally computational. Computational linguistics has a more intimate relationship with computers than many other disciplines that use computers as a tool. When a computational biologist simulates population dynamics, no animals die.[52] *When a meteorologist simulates the weather, nothing gets wet.*[53] ***But computers really do parse sentences. Natural language question-answering systems really do answer questions.***[54] *The actions of language are in the same space as computational activities, or alternatively, these particular computational activities are in the same space as language and communication."*

- A computer simulation of mathematics is mathematics. As Edelman (2008b, p. 81) put it (though not necessarily using the terms as Fetzer does):

"A simulation of a computation and the computation itself are equivalent: try to simulate the addition of 2 and 3, and the result will be just as good as if you "actually" carried out the addition—that is the nature of numbers."

- A computer simulation of reasoning *is* reasoning. This is the foundation of automated theorem proving.

And, in general, a computer simulation of cognition *is* cognition. To continue the just-cited quote from Edelman (2008b, p. 81):

"Therefore, if the mind is a computational entity, a simulation of the relevant computations would constitute its fully functional replica."

9. CONCLUSION

I conclude that Fetzer is mistaken on all counts: Computers *are* semiotic systems and *can* possess minds, mental processes *are* governed by algorithms (or, at least, "procedures"), and algorithms *can* be affected by emotions, attitudes, and individual histories. Moreover, computers that implement cognitive algorithms really do exhibit those cognitive behaviors—they really do think. And syntactic semantics explains how this is possible.

ACKNOWLEDGMENTS

This essay is based on my oral reply ("Salvaging Computationalism: How Cognition *Could Be* Computing") to two unpublished presentations by Fetzer: (1) "Can Computationalism Be Salvaged?", 2009 International Association for Computing and Philosophy (Chicago), and (2) "Limits to Simulations of Thought and Action", session on "The Limits of Simulations of Human Actions" at the 2010 North American Computing and Philosophy conference (Carnegie-Mellon University). I am grateful to Fetzer for allowing me to quote from his slides and from an unpublished manuscript; to our co-presenters at both meetings—Selmer Bringsjord and James H. Moor—for their commentaries; to our unique audiences at those meetings; and to Randall R. Dipert, Albert Goldfain, Lorenzo Incardona, Kenneth W. Regan, Daniel R. Schlegel, Stuart C. Shapiro, and members of the SNePS Research Group for comments or assistance on previous drafts.

REFERENCES

Ames, W. S. (1966). The development of. classification scheme of contextual aids. *Reading Research Quarterly*, *2*(1), 57–82. doi:10.2307/747039

Andersen, P. B. (1992). Computer semiotics. *Scandinavian Journal of Information Systems, 4*(1), 1-30. Retrieved May 5, 2011, from http://aisle.aisnet.org/cgi/viewcontent.cgi?article=1189&context=sjisAnstey, J., Seyed, A. P., Bay-Cheng, S., Pape, D., Shapiro, S. C., Bona, J.,. Hibit, S. (2009). The agent takes the stage. *International Journal of Arts and Technology*, *2*(4), 277–296. doi:10.1504/IJART.2009.029236

Barsalou, L. W. (1999). Perceptual symbol systems. *The Behavioral and Brain Sciences*, *22*, 577–660.

Brighton, H. (2002). Compositional syntax from cultural transmission. *Artificial Life*, *8*(1), 25–54. doi:10.1162/106454602753694756

Brown, S. R. (2002). Peirce, Searle, and the Chinese Room Argument. *Cybernetics. Human Knowing*, *9*(1), 23–38.

Burge, T. (1986). Individualism and psychology. *The Philosophical Review*, *95*(1), 3–45. doi:10.2307/2185131

Cariani, P. (2001). Symbols and dynamics in the brain. *Bio Systems*, *60*, 59–83. doi:10.1016/S0303-2647(01)00108-3

Chalmers, D. J. (1996). *The conscious mind: In search of. fundamental theory*. New York, NY: Oxford University Press.

Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics*, *58*(2), 345–363. doi:10.2307/2371045

Cleland, C. E. (1993). Is the Church-Turing thesis true? *Minds and Machines, 3*(3), 283-312.

Crane, T. (1990). The language of thought: No syntax without semantics. *Mind. Language*, *5*(3), 187–212. doi:10.1111/j.1468-0017.1990.tb00159.x

Crossley, J. N.,. Henry, A. S. (1990). Thus Spake al-Khwārizmī:. translation of the text of Cambridge University Library Ms. Ii.vi.5. *Historia Mathematica*, *17*, 103–131. doi:10.1016/0315-0860(90)90048-I

Dennett, D. (2009). Darwin's 'Strange Inversion of Reasoning.' *Proceedings of the National Academy of Sciences of the United States of America*, *106*(1), 10061–10065. doi:10.1073/pnas.0904433106

Dietrich, E. (1990). Computationalism. *Social Epistemology*, *4*(2), 135–154. doi:10.1080/02691729008578566

Dipert, R. R. (1984). Peirce, Frege, the Logic of Relations, and Church's Theorem. *History and Philosophy of Logic*, *5*, 49–66. doi:10.1080/01445348408837062

Dresner, E. (2010). Measurement-theoretic representation and computation-theoretic realization. *The Journal of Philosophy*, *107*(6), 275–292.

Dreyfus, H. L. (1965). Alchemy and artificial intelligence (Report No. P-3244). Santa Monica, CA: Rand. Retrieved November 2, 2011, from http://tinyurl.com/4hczr59

Dreyfus, H. L. (1972). *What computers can't do: The limits of artificial intelligence*. New York, NY: Harper. Row.

Dreyfus, H. L. (1979). *What computers can't do: The limits of artificial intelligence* (Rev. ed.). New York, NY: Harper. Row.

Dreyfus, H. L. (1992). *What computers still can't do:. critique of artificial reason*. Cambridge, MA: MIT Press.

Dulin, K. L. (1970). Using context clues in word recognition and comprehension. *The Reading Teacher*, *23*(5), 440–445, 469.

Eco, U. (1979).. *theory of semiotics*. Bloomington, IN: Indiana University Press.

Eco, U. (1988). On truth:. fiction. In Eco, U., Santambrogio, M.,. Violi, P. (Eds.), *Meaning and mental representations* (pp. 41–59). Bloomington, IN: Indiana University Press.

Edelman, S. (2008a). On the nature of minds, or: truth and consequences. *Journal of Experimental. Theoretical Artificial Intelligence*, *20*, 181–196. doi:10.1080/09528130802319086

Edelman, S. (2008b). *Computing the mind*. New York, NY: Oxford University Press.

Edelman, S. (2011). Regarding reality: Some consequences of two incapacities. *Frontiers in Theoretical and Philosophical Psychology*, *2*, 1–8.

Ehrlich, K. (1995). *Automatic vocabulary expansion through narrative context* (Tech. Rep. No. 95-09). Buffalo, NY: Department of Computer Science, SUNY Buffalo.

Fetzer, J. H. (1990). *Artificial intelligence: Its scope and limits*. Dordrecht, The Netherlands: Kluwer Academic.

Fetzer, J. H. (1994). Mental algorithms: Are minds computational systems? *Pragmatics. Cognition*, *2*, 1–29. doi:10.1075/pc.2.1.01fet

Fetzer, J. H. (1998). People are not computers: (Most) thought processes are not computational procedures. *Journal of Experimental. Theoretical Artificial Intelligence*, *10*, 371–391. doi:10.1080/095281398146653

Fetzer, J. H. (2001). *Computers and cognition: Why minds are not machines*. Dordrecht, The Netherlands: Kluwer Academic.

Fetzer, J. H. (2008). *Computing vs. cognition: Three dimensional differences.* Unpublished manuscript.

Fetzer, J. H. (2010, July). *Limits to simulations of thought and action*. Paper presented at the North American Conference on Computing and Philosophy, Pittsburgh, PA.

Fetzer, J. H. (2011). Minds and machines: Limits to simulations of thought and action. *International Journal of Signs and Semiotic Systems*, *1*(1), 39–48. doi:10.4018/ijsss.2011010103

Fodor, J. A. (1975). *The language of thought*. New York, NY: Crowell.

Fodor, J. A. (1980). Methodological solipsism considered as. research strategy in cognitive psychology. *The Behavioral and Brain Sciences*, *3*, 63–109. doi:10.1017/S0140525X00001771

Fodor, J. A. (2000). *The mind doesn't work that way: The scope and limits of computational psychology*. Cambridge, MA: MIT Press.

Fodor, J. A. (2008). *LOT 2: The language of thought revisited*. Oxford, UK: Clarendon.

Forbus, K. D. (2010). AI and cognitive science: The past and next 30 years. *Topics in Cognitive Science*, *2*(3), 345–356. doi:10.1111/j.1756-8765.2010.01083.x

Ford, K.,. Hayes, P. (1998). On computational wings. *Scientific American Presents*, *9*(4), 78–83.

Franzén, T. (2005). *Gödel's Theorem: An incomplete guide to its use and abuse*. Wellesley, MA: A. K. Peters. doi:10.1201/b10700

Gandy, R. (1988). The confluence of ideas in 1936. In Herken, R. (Ed.), *The universal Turing machine:. half-century survey* (2nd ed., pp. 51–102). Vienna, Austria: Springer-Verlag.

Gelman, R.,. Gallistel, C. R. (1986). *The child's understanding of number*. Cambridge, MA: Harvard University Press.

Goldfain, A. (2006). Embodied enumeration: Appealing to activities for mathematical explanation. In M. Beetz, K. Rajan, M. Thielscher,. R. Bogdan Rusu (Eds.), *Cognitive robotics: Papers from the AAAI Workshop* (pp. 69-76). Menlo Park, CA: AAAI Press.

Goldfain, A. (2008).. *computational theory of early mathematical cognition* (Unpublished doctoral dissertation). Buffalo, NY: Department of Computer Science. Engineering, SUNY Buffalo. Retrieved May 5, 2011, from http://www.cse.buffalo.edu/sneps/Bibliography/GoldfainDissFinal.pdf

Gopnik, A. (2009). *The philosophical baby: What children's minds tell us about truth, love, and the meaning of life*. New York, NY: Farrar, Straus and Giroux.

Habib, S. (1983). Emulation. In Ralston, A.,. Reilly, E. D. Jr., (Eds.), *Encyclopedia of computer science and engineering* (2nd ed., pp. 602–603). New York, NY: Van Nostrand Reinhold.

Harnad, S. (1990). The symbol grounding problem. *Physica D. Nonlinear Phenomena*, *42*, 335–346. doi:10.1016/0167-2789(90)90087-6

Harris, P. (2000). *The work of the imagination*. Malden, MA: Blackwelline

Hobbes, T. (1651). *Leviathan*. Indianapolis, IN: Bobbs-Merrill Library of Liberal Arts.

Hofstadter, D. (2007).. *am. strange loop*. New York, NY: Basic Books.

Horst, S. (2009). The computational theory of mind. In E. N. Zalta (Ed.), *Stanford encyclopedia of philosophy.* Retrieved May 6, 2011, from http://plato.stanford.edu/archives/win2009/entries/computational-mind/

Huemer, M. (2011). Sense-Data. In E. N. Zalta (Ed.), *Stanford encyclopedia of philosophy.* Retrieved May 6, 2011, from http://plato.stanford.edu/archives/spr2011/entries/sense-data/

Hume, D. (1888).. *treatise of human nature* (Selby-Bigge, L. A., Ed.). Oxford, UK: Oxford University Press. (Original work published 1739)

Hume, D. (1902). *An enquiry concerning human understanding* (Selby-Bigge, L. A., Ed.). Oxford, UK: Oxford University Press. (Original work published 1777)

Incardona, L. (in press). *Semiotica. Web Semantico: Basi teoriche. metodologiche per la semiotica computazionale* [Semiotics and Semantic Web: Theoretical and Methodological Foundations for Computational Semiotics] (Unpublished doctoral dissertation). University of Bologna, Bologna, Italy.

Jackendoff, R. (2002). *Foundations of language: Brain, meaning, grammar, evolution*. Oxford, UK: Oxford University Press.

Johnson-Laird, P. N. (1983). *Mental models: Towards. cognitive science of language, inference, and consciousness*. Cambridge, MA: Harvard University Press.

Johnson-Laird, P. N. (1988). *The computer and the mind: An introduction to cognitive science*. Cambridge, MA: Harvard University Press.

Jones, D. (2010, July). *Animat Liberation.* Paper presented at the North American Conference on Computing and Philosophy, Pittsburgh, PA.

Julesz, B. (1971). *Foundations of cyclopean perception*. Chicago, IL: University of Chicago Press.

Ketner, K. L. (1988). Peirce and Turing: Comparisons and conjectures. *Semiotica*, *68*(1-2), 33–61. doi:10.1515/semi.1988.68.1-2.33

Kirby, S. (2000). Syntax without natural selection: How compositionality emerges from vocabulary in. population of learners. In Knight, C. (Ed.), *The evolutionary emergence of language: Social function and the origins of linguistic form* (pp. 303–323). Cambridge, UK: Cambridge University Press. doi:10.1017/CBO9780511606441.019

Kleene, S. C. (1952). *Introduction to metamathematics*. Princeton, NJ: D. Van Nostrand.

Kleene, S. C. (1967). *Mathematical logic*. New York, NY: Wiley.

Knuth, D. E. (1972). Ancient Babylonian algorithms. *Communications of the ACM*, *15*(7), 671–677. doi:10.1145/361454.361514

Knuth, D. E. (1973). Basic concepts: §1.1: Algorithms. In Knuth, D. E., *The art of computer programming* (2nd ed., p. xiv-9). Reading, MA: Addison-Wesley.

Kugel, P. (2002). Computing machines can't be intelligent (…and Turing said so). *Minds and Machines*, *12*(4), 563–579. doi:10.1023/A:1021150928258

Levesque, H. (2012). *Thinking as computation:. first course*. Cambridge, MA: MIT Press.

Lobina, D. J. (2010). Recursion and the competence/performance distinction in AGL tasks. *Language and Cognitive Processes.* Retrieved November 4, 2011, from http://tinyurl.com/Lobina2010

Lobina, D. J.,. García-Albea, J. E. (2009). Recursion and cognitive science: Data structures and mechanism. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society* (pp. 1347-1352).

Lohr, S. (2010, September 24). Frederick Jelinek, who gave machines the key to human speech, dies at 77. *The New York Times*, p. B10.

Lucas, J. R. (1961). Minds, machines and Gödel. *Philosophy (London, England)*, *36*, 112–127. doi:10.1017/S0031819100057983

Makuuchi, M., Bahlmann, J., Anwander, A.,. Friederici, A. D. (2009). Segregating the core computational faculty of human language from working memory. *Proceedings of the National Academy of Sciences of the United States of America*, *106*(20), 8362–8367. doi:10.1073/pnas.0810928106

Mann, G. A. (2010).. machine that daydreams. *IFIP Advances in Information and Communication Technology*, *333*, 21–35. doi:10.1007/978-3-642-15214-6_3

Markov, A. A. (1960). Theory of algorithms (E. Hewitt, Trans.). In *American Mathematical Society Translations* (Series 2, Vol. 15, pp. 1-14). Moscow, USSR: Academy of Sciences. (Original work published 1954)

Marr, D. (1982). *Vision:. computational investigation into the human representation and processing of visual information*. New York, NY: W. H. Freeman.

McCorduck, P. (1990). *Aaron's Code: Meta-art, artificial intelligence, and the work of Harold Cohen*. New York, NY: W. H. Freeman.

McCulloch, W. S.,. Pitts, W. H. (1943).. logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*, 115–133. doi:10.1007/BF02478259

McGee, K. (2011). Review of K. Tanaka-Ishii's *Semiotics of Programming. Artificial Intelligence*, *175*, 930–931. doi:10.1016/j.artint.2010.11.023

Mill, J. S. (1843).. *system of logic*. Retrieved November 4, 2011, from http://tinyurl.com/Mill-SystemLogic

Miller, G. A., Galanter, E.,. Pribram, K. H. (1960). *Plans and the structure of behavior*. New York, NY: Henry Holt. doi:10.1037/10039-000

Morris, C. (1938). *Foundations of the theory of signs*. Chicago, IL: University of Chicago Press.

Mueller, E. T. (1990). *Daydreaming in humans and machines:. computer model of the stream of thought*. Norwood, NJ: Ablex.

Nadin, M. (2007). Semiotic machine. *Public Journal of Semiotics*, *1*(1), 85–114.

Nadin, M. (2010). Remembering Peter Bøgh Andersen: Is the computer. semiotic machine?. discussion never finished. *Semiotix: New Series, 1*. Retrieved May 5, 2011, from http://www.semioticon.com/semiotix/2010/03

Newell, A., Shaw, J. C.,. Simon, H. A. (1958). Elements of. theory of human problem solving. *Psychological Review*, *65*(3), 151–166. doi:10.1037/h0048495

Newell, A.,. Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, *19*(3), 113–126. doi:10.1145/360018.360022

Nöth, W. (2003). Semiotic machines. *S.E.E.D. Journal (Semiotics, Evolution, Energy, and Development), 3*(3), 81-99.

Parisien, C.,. Thagard, P. (2008). Robosemantics: How Stanley represents the world. *Minds and Machines*, *18*(2), 169–178. doi:10.1007/s11023-008-9098-2

Pattee, H. H. (1969). How does. molecule become. message? *Developmental Biology*, *3*(Supplement), 1–16.

Pattee, H. H. (1982). Cell psychology: An evolutionary approach to the symbol-matter problem. *Cognition and Brain Theory*, *5*(4), 325–341.

Peirce, C. S. (1887). Logical machines. *The American Journal of Psychology*, *1*, 165–170.

Peirce, C. S. (1977). *Semiotic and Significs: The correspondence between Charles S. Peirce and Victoria Lady Welby* (Hardwick, C. S.,. Cook, J., Eds.). Bloomington, IN: Indiana University Press. (Original work published 1908)

Peirce, C. S. (1992). Some consequences of four incapacities. In Houser, N.,. Kloesel, C. (Eds.), *The essential Peirce: Selected philosophical writings* (Vol. 1, pp. 186–199). Bloomington, IN: Indiana University Press.

Peirce, C. S. (1998). *The essential Peirce: Selected philosophical writings* (Peirce Edition Project, Ed.) (Vol. 2). Bloomington, IN: Indiana University Press.

Peirce, C. S. (1931-1958). In Hartshorne, C., Weiss, P.,. Burks, A. W. (Eds.), *Collected Papers of Charles Sanders Peirce* (Vol. 1-8). Cambridge, MA: Harvard University Press.

Penrose, R. (1989). *The emperor's new mind: Concerning computers, minds and the laws of physics*. New York, NY: Oxford University Press.

Perlovsky, L. I. (2007). Symbols: Integrated cognition and language. In Gudwin, R.,. Queiroz, J. (Eds.), *Semiotics and intelligent systems development* (pp. 121–151). Hershey, PA: Idea Group.

Phillips, A. L. (2011). The algorists. *American Scientist*, *99*(2), 126. doi:10.1511/2011.89.126

Picard, R. W. (1997). *Affective computing*. Cambridge, MA: MIT Press.

Piccinini, G. (2005). Symbols, strings, and spikes: The empirical refutation of computationalism. Retrieved November 4, 2011, from http://tinyurl.com/Piccinini2005

Piccinini, G. (2007). Computational explanation and mechanistic explanation of mind. In Marraffa, M., De Caro, M.,. Ferretti, F. (Eds.), *Cartographies of the mind: Philosophy and psychology in intersection* (pp. 23–36). Dordrecht, The Netherlands: Springer-Verlag. doi:10.1007/1-4020-5444-0_2

Piccinini, G. (2008). Computation without representation. *Philosophical Studies*, *137*(2), 205–241. doi:10.1007/s11098-005-5385-4

Piccinini, G. (2010). The mind as neural software? Understanding functionalism, computationalism, and computational functionalism. *Philosophy and Phenomenological Research*, *81*(2), 269–311. doi:10.1111/j.1933-1592.2010.00356.x

Piccinini, G.,. Scarantino, A. (2011). Information processing, computation, and cognition. *Journal of Biological Physics*, *37*, 1–38. doi:10.1007/s10867-010-9195-3

Pollock, J. L. (2008). What am I? Virtual machines and the mind/body problem. *Philosophy and Phenomenological Research*, *76*(2), 237–309. doi:10.1111/j.1933-1592.2007.00133.x

Posner, R. (1992). Origins and development of contemporary syntactics. *Languages of Design*, *1*, 37–50.

Putnam, H. (1960). Minds and machines. In Hook, S. (Ed.), *Dimensions of mind:. symposium* (pp. 148–179). New York, NY: New York University Press.

Putnam, H. (1961). *Brains and behavior.* Paper presented at the American Association for the Advancement of Science, Section. (History and Philosophy of Science).

Putnam, H. (1975). The meaning of 'meaning. In Gunderson, K. (Ed.), *Language, mind, and knowledge (Minnesota Studies in the philosophy of science)* (Vol. 7, pp. 131–193). Minneapolis, MN: University of Minnesota Press. (Reprinted in *Mind, language, and reality*, pp. 215-271, by H. Putnam, 1979, Cambridge, UK: Cambridge University Press)

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

Pylyshyn, Z. W. (1980). Computation and cognition: Issues in the foundations of cognitive science. *The Behavioral and Brain Sciences*, *3*, 111–169. doi:10.1017/S0140525X00002053

Pylyshyn, Z. W. (1985). *Computation and cognition: Toward. foundation for cognitive science* (2nd ed.). Cambridge, MA: MIT Press.

Ramachandran, V.,. Rogers-Ramachandran, D. (2009). Two eyes, two views. *Scientific American Mind*, *20*(5), 22–24. doi:10.1038/scientificamericanmind0909-22

Rapaport, W. J. (1978). Meinongian Theories and. Russellian Paradox. *Noûs, 12*, 153-180. [Errata, Nous, 13, 125]. doi:10.2307/2214805

Rapaport, W. J. (1979). An Adverbial Meinongian Theory. *Analysis*, *39*, 75–81. doi:10.2307/3327784

Rapaport, W. J. (1981). How to make the world fit our language: An essay in Meinongian Semantics. *Grazer Philosophische Studien*, *14*, 1–21.

Rapaport, W. J. (1985). Non-existent objects and epistemological ontology. [Reprinted in *Non-existence and predication*, R. Haller, Ed., 1986, Amsterdam, The Netherlands: Rodopi]. *Grazer Philosophische Studien*, *25-26*, 61–95.

Rapaport, W. J. (1986). Searle's experiments with thought. *Philosophy of Science*, *53*, 271–279. doi:10.1086/289312

Rapaport, W. J. (1988). Syntactic semantics: Foundations of computational natural-language understanding. In J. H. Fetzer (Ed.), *Aspects of artificial intelligence* (pp. 81-131). Dordrecht, The Netherlands: Kluwer Academic. Eratta retrieved November 4, 2011, from http://tinyurl.com/SynSemErrata1 (Reprinted in *Thinking computers and virtual persons: Essays on the intentionality of machines*, pp. 225-273, E. Dietrich, Ed., 1994, San Diego, CA: Academic Press)

Rapaport, W. J. (1990). *Computer processes and virtual persons: Comments on Cole's 'Artificial Intelligence and Personal Identity'* (Tech. Rep. No. 90-13). Buffalo, NY: Department of Computer Science, SUNY Buffalo. Retrieved November 4, 2011, from http://tinyurl.com/Rapaport1990

Rapaport, W. J. (1995). Understanding understanding: Syntactic semantics and computational cognition. In Tomberlin, J. E. (Ed.), *AI, connectionism, and philosophical psychology (Philosophical perspectives,* Vol. 9, pp. 49–88). Atascadero, CA: Ridgeview. (Reprinted in *Language and meaning in cognitive science: Cognitive issues and semantic theory (Artificial intelligence and cognitive science: Conceptual issues)*, Vol. 4, pp. 73-88, J. Toribio. A. Clark, Ed., 1998, New York, NY: Garland) doi:10.2307/2214212

Rapaport, W. J. (1998). How minds can be computational systems. *Journal of Experimental. Theoretical Artificial Intelligence*, *10*, 403–419. doi:10.1080/095281398146671

Rapaport, W. J. (1999). Implementation is semantic interpretation. *The Monist*, *82*, 109–130.

Rapaport, W. J. (2000). How to pass. Turing Test: Syntactic semantics, natural-language understanding, and first-person cognition. [Reprinted in *The Turing Test: The elusive standard of artificial intelligence*, pp. 161-184, J. H. Moor, Ed., 2003, Dordrecht, The Netherlands: Kluwer Academic]. *Journal of Logic Language and Information*, *9*(4), 467–490. doi:10.1023/A:1008319409770

Rapaport, W. J. (2002). Holism, conceptual-role semantics, and syntactic semantics. *Minds and Machines*, *12*(1), 3–59. doi:10.1023/A:1013765011735

Rapaport, W. J. (2003a). What did you mean by that? Misunderstanding, negotiation, and syntactic semantics. *Minds and Machines*, *13*(3), 397–427. doi:10.1023/A:1024145126190

Rapaport, W. J. (2003b). What is the 'context' for contextual vocabulary acquisition? In *Proceedings of the 4th International Conference on Cognitive Science and the 7th Australasian Society for Cognitive Science Conference*, Sydney, Australia (Vol. 2, pp. 547-552). Sydney, Australia: University of New South Wales.

Rapaport, W. J. (2005a). In defense of contextual vocabulary acquisition: How to do things with words in context. In A. Dey, B. Kokinov, D. Leake,. R. Turner (Eds.), *Proceedings of the 5th International and Interdisciplinary Conference on Modeling and Using Context*, Paris, France. Berling: Springer (LNCS 3554, pp. 396-409).

Rapaport, W. J. (2005b). Implementation is semantic interpretation: Further thoughts. *Journal of Experimental. Theoretical Artificial Intelligence*, *17*(4), 385–417. doi:10.1080/09528130500283998

Rapaport, W. J. (2006). How Helen Keller used syntactic semantics to escape from. Chinese Room. *Minds and Machines*, *16*(4), 381–436. doi:10.1007/s11023-007-9054-6

Rapaport, W. J. (2011). Yes, she was! Reply to Ford's 'Helen Keller was never in. Chinese Room'. *Minds and Machines*, *21*(1), 3–17. doi:10.1007/s11023-010-9213-z

Rapaport, W. J.,. Ehrlich, K. (2000).. computational theory of vocabulary acquisition. In L. M. Iwańska. S. C. Shapiro (Eds.), *Natural language processing and knowledge representation: Language for knowledge and knowledge for language* (pp. 347-375). Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press. Errata retrieved November 4, 2011, from http://tinyurl.com/RapaportEhrlichErrata

Rapaport, W. J.,. Kibby, M. W. (2007). Contextual vocabulary acquisition as computational philosophy and as philosophical computation. *Journal of Experimental. Theoretical Artificial Intelligence*, *19*(1), 1–17. doi:10.1080/09528130601116162

Rapaport, W. J.,. Kibby, M. W. (2010). *Contextual vocabulary acquisition: From algorithm to curriculum*. Retrieved November 4, 2011, from http://tinyurl.com/RapaportKibby2010

Ray, K. (2010). *Web 3.0.* Retrieved March 1, 2011, from http://kateray.net/film

Rosen, R. (1987). On the scope of syntactics in mathematics and science: The machine metaphor. In Casti, J. L.,. Karlqvist, A. (Eds.), *Real brains, artificial minds* (pp. 1–23). New York, NY: Elsevier Science.

Rosser, B. (1939). An informal exposition of proofs of Gödel's Theorem. *Journal of Symbolic Logic*, *4*(2), 53–60. doi:10.2307/2269059

Roth, P. F. (1983). Simulation. In Ralston, A.,. Reilly, E. D. Jr., (Eds.), *Encyclopedia of computer science and engineering* (2nd ed., pp. 1327–1341). New York, NY: Van Nostrand Reinhold.

Santore, J. F.,. Shapiro, S. C. (2003). Crystal Cassie: Use of. 3-D gaming environment for. cognitive agent. In *Proceedings of the IJCAI Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions*, Acapulco, Mexico (pp. 84-91).

Schagrin, M. L., Rapaport, W. J.,. Dipert, R. R. (1985). *Logic:. computer approach*. New York, NY: McGraw-Hill.

Schneider, S. (2009). The paradox of fiction. In *Internet encyclopedia of philosophy.* Retrieved May 6, 2011, from http://www.iep.utm.edu/fict-par/

Searle, J. R. (1980). Minds, brains, and programs. *The Behavioral and Brain Sciences*, *3*, 417–457. doi:10.1017/S0140525X00005756

Searle, J. R. (1990). Is the brain. digital computer? *Proceedings and Addresses of the American Philosophical Association*, *64*(3), 21–37. doi:10.2307/3130074

Segal, G. (1989). Seeing what is not there. *The Philosophical Review*, *98*(2), 189–214. doi:10.2307/2185282

Shapiro, S. C. (1992). Artificial intelligence. In Shapiro, S. C. (Ed.), *Encyclopedia of artificial intelligence* (2nd ed., pp. 54–57). New York, NY: John Wiley. Sons.

Shapiro, S. C. (1993). Belief spaces as sets of propositions. *Journal of Experimental. Theoretical Artificial Intelligence*, *5*(2-3), 225–235. doi:10.1080/09528139308953771

Shapiro, S. C. (1998). Embodied Cassie. In *Proceedings of the AAAI Fall Symposium on Cognitive Robotics* (Tech. Rep. No. FS-98-02) (pp. 136-143). Menlo Park, CA: AAAI Press.

Shapiro, S. C. (2001). Computer science: The study of procedures. Retrieved May 5, 2011, from http://www.cse.buffalo.edu/~shapiro/Papers/whatiscs.pdf

Shapiro, S. C. (2006). Natural-language-competent robots. *IEEE Intelligent Systems*, *21*(4), 76–77.

Shapiro, S. C., Amir, E., Grosskreutz, H., Randell, D.,. Soutchanski, M. (2001, May 20-22). Commonsense and embodied agents:. panel discussion. In *Proceedings of the 5th International Common Sense Symposium on Logical Formalizations of Commonsense Reasoning, Courant Institute of Mathematical Sciences*, New York, NY. Retrieved October 20, 2011, from http://www.cse.buffalo.edu/~shapiro/Papers/commonsense-panel.pdf

Shapiro, S. C.,. Bona, J. P. (2010). The GLAIR cognitive architecture. *International Journal of Machine Consciousness*, *2*, 307–332. doi:10.1142/S1793843010000515

Shapiro, S. C.,. Ismail, H. O. (2003). Anchoring in. grounded layered architecture with integrated reasoning. *Robotics and Autonomous Systems*, *43*(2-3), 97–108. doi:10.1016/S0921-8890(02)00352-4

Shapiro, S. C.,. Rapaport, W. J. (1987). SNePS considered as. fully intensional propositional semantic network. In Cercone, N.,. McCalla, G. (Eds.), *The knowledge frontier: Essays in the representation of knowledge* (pp. 262–315). New York, NY: Springer. doi:10.1007/978-1-4612-4792-0_11

Shapiro, S. C.,. Rapaport, W. J. (1991). Models and minds: Knowledge representation for natural-language competence. In Cummins, R.,. Pollock, J. (Eds.), *Philosophy and AI: Essays at the interface* (pp. 215–259). Cambridge, MA: MIT Press.

Shapiro, S. C., Rapaport, W. J., Kandefer, M., Johnson, F. L.,. Goldfain, A. (2007). Metacognition in SNePS. *AI Magazine*, *28*(1), 17–31.

Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological Review*, *63*(2), 129–138. doi:10.1037/h0042769

Simon, H. A. (1967). Motivational and emotional controls of cognition. *Psychological Review*, *74*, 29–39. doi:10.1037/h0024127

Sloman, A. (2004). What are emotion theories about? In *Proceedings of the AAAI Spring Symposium Workshop on Architectures for Modeling Emotion*, Stanford, CA.

Sloman, A. (2009). The cognition and affect project. Retrieved May 6, 2011, from http://www.cs.bham.ac.uk/research/projects/cogaff/cogaff.html

Sloman, A.,. Croucher, M. (1981). Why robots will have emotions. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Vancouver, BC, Canada. Retrieved May 6, 2011, from http://www.cs.bham.ac.uk/research/projects/cogaff/81-95.html#36

Smith, B. (2001). True grid. In D. Montello (Ed.), *Proceedings of the COSIT Conference on Spatial Information Theory: Foundations of Geographic Information Science*, Morro Bay, CA, Berlin: Springer (LNCS 2205, pp. 14-27).

Smith, B. C. (1982). Linguistic and computational semantics. In *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics*, Toronto, ON, Canada (pp. 9-15). Morristown, NJ: Association for Computational Linguistics.

Smith, B. C. (1985). Limits of correctness in computers. [Reprinted in *Program verification*, pp. 275-293, T. R. Colburn, J. H. Fetzer,. T. Rankin, Eds., 1993, Dordrecht, The Netherlands: Kluwer Academic]. *ACM SIGCAS Computers and Society*, *14-15*(1-4), 18–26. doi:10.1145/379486.379512

Smith, B. C. (2010). Introduction. In B. C. Smith (Ed.), *Age of significance.* Cambridge, MA: MIT Press. Retrieved March 24, 2011, from http://www.ageofsignificance.org/aos/en/aos-v1c0.html

Soare, R. I. (2009). Turing oracle machines, online computing, and three displacements in computability theory. *Annals of Pure and Applied Logic*, *160*, 368–399. doi:10.1016/j.apal.2009.01.008

Srihari, R. K.,. Rapaport, W. J. (1989). Extracting visual information from text: Using captions to label human faces in newspaper photographs. In *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, Ann Arbor, MI (pp. 364-371). Mahwah, NJ: Lawrence Erlbaum.

Srihari, R. K.,. Rapaport, W. J. (1990). Combining linguistic and pictorial information: Using captions to interpret newspaper photographs. In D. Kumar (Ed.), *Current Trends in SNePS—Semantic Network Processing System,* Berlin: Springer (LNAI 437, pp. 85-96).

Strawson, G. (2010). *Mental reality* (2nd ed.). Cambridge, MA: MIT Press.

Tarski, A. (1944). The semantic conception of truth and the foundations of semantics. *Philosophy and Phenomenological Research*, *4*(3), 341–376. doi:10.2307/2102968

Tenenbaum, A. M.,. Augenstein, M. J. (1981). *Data structures using Pascal*. Upper Saddle River, NJ: Prentice Hall.

Thagard, P. (2006). *Hot thought: Mechanisms and applications of emotional cognition*. Cambridge, MA: MIT Press.

Toussaint, G. (1993).. new look at Euclid's Second Proposition. *Mathematical Intelligencer*, *15*(3), 12–23. doi:10.1007/BF03024252

Turing, A. M. (1936). On computable numbers, with an application to the *Entscheidungsproblem. Proceedings of the London Mathematical Society*, *42*(2), 230–265.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*, 433–460. doi:10.1093/mind/LIX.236.433

Van Gelder, T. (1995). What might cognition be, if not computation? *The Journal of Philosophy*, *92*(7), 345–381. doi:10.2307/2941061

Vera, A. H.,. Simon, H. A. (1993). Situated action:. symbolic interpretation. *Cognitive Science*, *17*(1), 7–48. doi:10.1207/s15516709cog1701_2

Wegner, P. (1997). Why interaction is more powerful than algorithms. *Communications of the ACM*, *40*(5), 80–91. doi:10.1145/253769.253801

Weizenbaum, J. (1976). *Computer power and human reason: From judgment to calculation*. New York, NY: W.H. Freeman.

Widdowson, H. G. (2004). *Text, context, pretext*. Malden, MA: Blackwell. doi:10.1002/9780470758427

Williams, B. (1998). The end of explanation? *The New York Review of Books*, *45*(18), 40–44.

Winograd, T.,. Flores, F. (1987). *Understanding computers and cognition:. new foundation for design*. Reading, MA: Addison-Wesley.

Woods, W. A. (2010). The right tools: Reflections on computation and language. *Computational Linguistics*, *36*(4), 601–630. doi:10.1162/coli_a_00018

Wright, I., Sloman, A.,. Beaudoin, L. (1996). Towards. design-based analysis of emotional episodes. *Philosophy, Psychiatry,. Psychology*, *3*(2), 101–126. doi:10.1353/ppp.1996.0022

Wu, H.-H. (2011). *Understanding numbers in elementary school mathematics*. Providence, RI: American Mathematical Society.

Zobrist, A. L. (2000). Computer games: Traditional. In Ralston, A., Reilly, E. D.,. Hemmendinger, D. (Eds.), *Encyclopedia of computer science* (4th ed., pp. 364–368). New York, NY: Grove's Dictionaries.

ENDNOTES

1 Although I take full responsibility for this essay, I am also using it as an opportunity to publicize two arguments for the claim that computers are semiotic systems. These arguments were originally formulated by my former students Albert Goldfain and Lorenzo Incardona. I am grateful to both of them for many discussions on the topics discussed here.

2 Throughout this essay, all italics in quotes are in the original, unless otherwise noted or supplied by the publisher.

3 The authors quoted here include most of the principal philosophers who have written on computationalism. There are, of course, many researchers in both symbolic and connectionist AI who, without necessarily taking explicit philosophical positions on computationalism, are computationalists of one stripe or another. These include, according to one anonymous referee for this journal, Barsalou, Grossberg, Kosslyn, Kozma, Perlovsky, and Rocha, all of whom do computational modeling of (human) cognition. One must be cautious, however; Perlovsky, in particular, misunderstands at least one important logical claim: Contrary to what he consistently says in several of his writings (e.g., Perlovsky, 2007, p. 123), Gödel *never* "proved inconsistency of logic". Rather, Gödel proved, roughly, that *if* a formal system consisting of first-order logic together with Peano's axioms for the natural numbers was consistent, then it was incomplete. Thus, such a system of "arithmetic" (which contains, but is not the same as, "logic") would only be inconsistent if it were complete. The usual interpretation of Gödel is that arithmetic *is* (in all likelihood) consistent; hence, it is incomplete. See note 49 for further discussion. See Franzén (2005) to be disabused of misinterpretations of Gödel.

4 See the rest of their Ch. 1, "Images and Plans" for some caveats, e.g., their endnote 12 (p. 16) says: "comparing the sequence of operations executed by an organism and by a properly programmed computer is quite different from comparing computers with brains, or electrical relays with synapses."

5 Except when quoting or in bibliographic references, I use single quotes to form names of expressions, and I use double quotes as "scare quotes" or to mark quoted passages.

6 By 'legal', I mean values in the domain of the function. So a partial function (i.e., one that is undefined on some set of values) would be such that those values on which it is un-

defined are "illegal". The notion of "legal" inputs makes sense in a mathematical context, perhaps less so in a biological one (Goldfain, personal communication, 2011). Biologically, presumably, all values are "legal", except that some of them are filtered out by our senses or produce unpredictable behavior.

[7] Or, as Randall R. Dipert pointed out to me (personal communication, 2011), we might be able to understand only those aspects of cognition that *are* computable.

[8] Notably, J. R. Lucas (1961), the ever-pessimistic Hubert Dreyfus (1965, 1972, 1979, 1992), John Searle (1980), Roger Penrose (1989), Terry Winograd and Fernando Flores (1987), to some extent Joseph Weizenbaum (1976, esp. Chs. 7, 8), and even Putnam (1988) and Fodor (2000) themselves!

[9] The "strong" and "weak" views are, perhaps, close to, though a bit different from, what Stuart C. Shapiro and I have called "computational psychology" and "computational philosophy", respectively (Shapiro, 1992; Rapaport, 2003a).

[10] Cf. also these passages from Newell, Shaw, and Simon (1958): "The theory [of human problem solving] postulates…[a] number of *primitive information processes*, which operate on the information in the memories. Each primitive process is a perfectly definite operation for which known physical mechanisms exist. (The mechanisms are not necessarily known to exist in the human brain, however—we are only concerned that the processes be described without ambiguity)" (p. 151, original emphasis). The parenthetical phrase can be given the "computable" reading. But I hear the stronger, "computational" reading in the next passage: "[O]ur theory of problem solving…shows specifically and in detail how the processes that occur in human problem solving can be compounded out of elementary information processes, and hence how they can be carried out by mechanism" (p. 152). I admit that the 'can be' weakens the "computational" reading to the "computable" reading.

[11] Here, 'function' is to be taken in the mathematical sense explicated in note 42, below, rather than in the sense of an activity or purpose. On the computational theory, the brain performs certain "purposeful functions" by computing certain "mathematical functions". Anti-computationalists say that the brain does not perform its "purposeful functions" by computing mathematical ones, but in some other way. Yet, I claim, those purposeful functions might be accomplished in a computational way. A computer programmed to compute those mathematical functions would thereby perform those purposeful functions.

[12] Here, however, there might be a conflation of two senses of 'recursive': (a) as synonymous with 'computable' and (b) as synonymous with "inductive" (Soare, 2009).

[13] See an ad for a (human) "computer," in *The New York Times* (May 2, 1892). Retrieved October 31, 2011, from http://tinyurl.com/NYT-ComputerWanted

[14] An anonymous referee commented, "These views are popular, but it does not make them true. They are wrong, (1) there is evidence that syntax can evolve as a secondary feature in language (Brighton, 2002; Kirby, 2000), (2) reclusiveness [*sic*] never was demonstrated to be useful, it follows from hierarchy, which is much more useful, (3) semantics refers to objects and events in the world, there can be no semantics inside a semiotic system, even if many people misunderstood semiotics this way." As will become clearer in §3.2, by 'syntacticism' here I do not mean *grammatical* syntax, as the referee seems to think I do; hence, point (1), though of independent interest, is irrelevant. I am at a loss as to what to say about point (2), other than that I am not referring to *recursiveness* in language, but to a recursive feature of *understanding*, as will be seen in §3.3. Finally, point (3) is simply false; semantics can "refer" to *mental* concepts as well as *worldly* objects and events. And at least one semiotician has remarked to me (personal communication) that many semioticians "misunderstand" semiotics in exactly that way: that talk about "objects and events in the world" is simplistic, if not blasphemous!

[15] "[S]o long as we are thinking of mental processes as purely computational, the bearing of environmental information upon such processes is exhausted by the formal character of whatever the oracles ["analogs to the senses"] write on the tape [of a Turing machine]. In particular, it doesn't matter to such processes whether what the oracles write is *true*….[T]he formality condition…is tantamount to a sort of methodological solipsism" (Fodor, 1980, p. 65).

[16] "According to individualism about the mind, the mental natures of all a person's or animal's mental states (and events) are such that there is no necessary or deep individuative relation between the individual's being in states of those kinds and the nature of the individual's physical or social environments" (Burge, 1986, pp. 3-4).

[17] An anonymous referee commented about this quote, "again, wrong, despite many famous people could be misinterpreted this way [*sic*]. Interaction between an organism and the world is a *process*. In this process there are (1) intuition[s] about the world, (2) predictions and (3) confirmations or disconfirmations of these predictions. The same process is the process of science." I agree that organism-world interaction is a process. Pylyshyn's point, however, is that the *output* of that process is the *only* input to our cognitive system. Hence, intuitions, predictions, and (dis-)confirmations are all internal.

[18] Strict adherence to methodological solipsism would seem to require that LOT have syntax but no semantics. Fodor (2008, p. 16) suggests that it needs a purely referential semantics. I have proposed instead a Meinongian semantics for LOT, on the grounds that "non-existent" objects are best construed as internal mental entities (Rapaport, 1978, 1979, 1981, 1985/1986).

[19] The terms (or "marks"; see §5.1) of this LOT (e.g., nodes of a semantic network; terms and predicates of a language; or their biological analogues, etc.) need not all be alike, either in "shape" or function, e.g., natural languages use a wide variety of letters, numerals, etc.; neurons include afferent, internal, and efferent ones (and the former do much of the internalization or "pushing"). (Albert Goldfain, personal communication.) Moreover, some of the internal marks might well include Barsalou's (1999) "perceptual symbols."

[20] When I use 'information' (except when quoting), I am using it in a sense that is neutral among several different meanings it can have; in particular, I do not necessarily intend the Shannon theory of information. For further discussion, see Piccinini and Scarantino (2011).

[21] Perception is input; so, if one wants to rule it out as an example of a cognitive phenomenon, then perhaps *outputs* of cognitive processes (e.g., motor activity, or actions more generally) should also be ruled out. This would be one way to respond to Cleland (1993; see n. 46).

[22] One anonymous referee objected that "Despite the limitations on [human] sensation, there is an analog aspect and nonarbitrariness about the [human] cognitive representations that is not shared with a computer." But a computer's representations of visual input are surely just as non-arbitrary (or, conversely, both are equally arbitrary; it's just that we're more familiar with our own) and just as "analog" as ours are; this is one of the main points of Marr's (1982) computational vision theory.

[23] http://www.oed.com/view/Entry/196559, accessed May 5, 2011.

[24] My use of 'mark' derives from Fetzer 2011 and will be clarified in §5.1.

[25] For this reason, Barry Smith and Werner Ceusters (personal communication) prefer to call it the '*Syntactic* Web'!

[26] i.e., they are pure syntax—WJR.

[27] Although this is suggestive of semantic interpretation, it is really just syntax: A bit is not something (a sign) that is then interpreted as something else (e.g., 0 or 1). Rather, a bit *is* 0 or else it *is* 1 (or it *is* high voltage or else it *is* low voltage; or it *is* magnetized or else it *is* not magnetized; and so on).—WJR

[28] This certainly *sounds* like ordinary semantics, but the very next passage clearly indicates a syntactic approach.—WJR

[29] "Declarations" are *syntactic* parts of a computer program.

[30] For more quotes along these lines, see my Web page "Tenenbaum & Augenstein on Data, Information, & Semantics." Retrieved November 1, 2011, from http://www.cse.buffalo.edu/~rapaport/563S05/data.html

[31] Shapiro (personal communication, 2011) suggests that, when we stop the recursion, we may only *think* that we understand, as in the case of the sentence, "During the Renaissance, Bernini cast a bronze of a mastiff eating truffles" (Johnson-Laird, personal communication, 2003; cf. Johnson-Laird, 1983, p. 225; Widdowson, 2004). The claim is that many people can understand this sentence without being able to precisely define any of the principal words, as long as they have even a vague idea that, e.g., the Renaissance was some period in history, 'Bernini' is someone's name, "casting a bronze" has something to do with sculpture, bronze is some kind of (perhaps yellowish) metal, a mastiff is some kind of animal (maybe a dog), and truffles are something edible (maybe a kid of mushroom, maybe a kind of chocolate candy). Still, such understanding is syntactic—it is understanding of one's internal concepts.

[32] A referee commented, concerning §§5 and 5.1, "it seems that the author assumes that language and cognition is the same thing.—this should be clarified. Syntax is a set of relations among signs. But how signs are related to external objects and situations—not a word so far." However, in these sections, I do not make this assumption. Moreover, a large part of the entire essay concerns how semantics in the sense of

the relation of signs to external objects and situations can be handled by syntax.

[33] Crane 1990 also points out that shape alone is not sufficient for syntax (as he interprets Fodor 1980 as holding). But Crane uses 'syntax' in the narrow sense of "grammar"; he is correct that a sentence printed in all capital letters has a different shape from—but the same (grammatical) syntax as—the same sentence printed normally. But, as I use the term, although shape does not *suffice* for syntax, it is surely part of it.

[34] The notion of "interpretant" will be clarified below.

[35] I owe the observations in this paragraph to Incardona, personal communication, 2010.

[36] I am indebted to Incardona for directing me to both this article by Peirce and an interpretation by Kenneth Laine Ketner (1988, see esp. pp. 34, 46, 49). I have two problems with Ketner's interpretation, however: (1) Ketner (p. 49) cites Peirce's distinction between "corollarial" reasoning (reading off a conclusion of an argument from a diagram of the premises) and "theorematic" reasoning (in which the reasoner must creatively *add* something to the premises). But neither Peirce nor Ketner offers any arguments that theorematic reasoning cannot be reduced to corollarial reasoning. If corollarial reasoning can be interpreted as referring to ordinary arguments with all premises explicitly stated, and theorematic reasoning can be interpreted as referring to arguments (viz., enthymemes) where a missing premise has to be supplied by the reasoner as "background knowledge", then I would argue that the latter can be reduced to the former (see Rapaport & Kibby, 2010). However, there are other interpretations (see Dipert, 1984, §4). (2) Ketner seems to misunderstand the Church-Turing Thesis (pp. 51-52). He presents it as stating that computers can only do what they have been programmed to do (or "calculated to do", to use Peirce's phrase). But what it really asserts is that the *informal* notion of "algorithm" can be *identified* with the *formal* notions of Church's lambda calculus or Turing's machines. Ketner also seems to misunderstand the import of Turing's proof of the existence of non-computable functions: He thinks that this shows "that mathematical method is not universally deterministic" (p. 57). But what it really shows is that Hilbert's decision problem (viz., for any mathematical proposition P, is there an algorithm that decides whether P is a theorem?) must be answered in the negative. Unfortunately, in both cases, Ketner cites as an authority a logic text (Schagrin et al., 1985, pp. 304-305) co-authored by me! Granted, my co-authors and I use Church's Thesis to justify a claim that "Computers can perform only what algorithms describe"—which *sounds like* "computers can only do what they have been programmed to do"—but our intent was to point out that, *if* a computer can perform a task, *then* that task can be described by an algorithm. Ketner's sound-alike claim is usually taken to mean that computers can't do anything other than what they were explicitly programmed to do (e.g., they can't show creativity or initiative). But computers that can learn can certainly do things that they weren't explicitly programmed to do; the Church-Turing thesis claims that anything that they *can* do—including those things that they learned how to do—must be computable in the technical sense of computable in the lambda calculus or by a Turing machine.

[37] Dipert (1984, p. 59) suggests that Peirce might also have anticipated the Church-Turing Thesis.

[38] Again, thanks to Incardona, whose forthcoming dissertation explores these themes (Incardona, forthcoming).

[39] One anonymous referee commented, "[T]he mark's user is always the same as the interpretant…so the user is the same as the interpretant". This is a misreading, either of me or of Peirce. I am not using 'interpretant' to be synonymous with 'mark user'. Rather, I am following Peirce's use, roughly to the effect that an interpretant is a sign (or sign-like entity) *in* the user's mind.

[40] "That determination of which the immediate cause, or determinant, is the Sign, and of which the mediate cause is the Object may be termed the Interpretant…." (CP 6.347).

[41] "I call a representamen which is determined by another representamen, an interpretant of the latter" (CP 5.138).

[42] Mathematically a binary *relation* defined on the Cartesian product of two sets A and B (i.e., defined on $A \times B$) is a set of ordered pairs ("input-output" pairs), where the first member comes from A and the second from B. A *function* is a binary relation f on $A \times B$ ("from A to B") such that any given member a of A is related by f to (at most) a unique b in B, i.e., if $(a, b_1) \in f$ and $(a, b_2) \in f$ (where $a \in A$ and $b_1, b_2 \in B$), then $b_1 = b_2$. Conversely, if $b_1 \neq b_2$, then f cannot map (or interpret) a as *both* b_1 *and* b_2 simultaneously, i.e., no *two* outputs have the same input. If the outputs are thought of as the meanings or "interpreta-

tions" (i.e., *interpretants*) of the inputs, then an interpretation *function* cannot allow for ambiguity.

[43] This question was raised by an anonymous referee.

[44] For further discussion of the relation of computers to semiotic systems, see Andersen (1992), who argues that "computer systems" are (merely) "sign-vehicles whose main function is to be perceived and interpreted by some group of users"; Nöth (2003), who argues that "none of the criteria of semiosis is completely absent from the world of machines"; and Nadin (2007, 2010), who seems to agree that "computers are semiotic machines". Other sources are cited in McGee (2011).

[45] In Rapaport (1999, 2005b), I argue that implementation as semantic interpretation is also a ternary relation: Something is (1) a (concrete or abstract) implementation (i.e., a semantic "interpretant") of (2) an abstraction in (3) a (concrete or abstract) medium.

[46] One anonymous referee said: "I disagree here: The output of a process (algorithm) need not be caused by a certain input. It could have been caused by something else." This is technically true: The output of a constant function is not causally or algorithmically related to any input. But in all other cases an algorithm's output *is* a function of its input. The referee continues: "For instance, me going to eat (output) is not necessarily an interpretant of my being hungry. It could be because I have an appointment with a friend to go out eating. Likewise, a given input can yield different outputs. When hungry, I may decide to eat, but also may decide to continue working." I agree about hunger and eating, but the relation between being hungry and going out to eat is not algorithmic; this is not sign interpretation. The referee continues: "what I mean to say is that both input and output—and past experiences—are so tightly linked that the link/process should be part of the interpretant." If 'interpretant' means 'mark user', then I agree that the "process should be part of" the user (or the computer), but this makes no sense on the Peircean characterization of 'interpretant'. Finally, the referee says, "the context (input) and process are crucial to meaning formation." I quite agree that all *three* of these are crucial, but it is important to note that context and input are *two* distinct things. A context is not input to an algorithm; rather, it is the environment in which the algorithm is executed. Elaboration of this would take us too far afield; the interested reader should consult Smith (1985), Cleland (1993), and Soare (2009) for further discussion of the relationship between context and input.

[47] Related to the notion of a semiotic system is that of a *symbol* system, as described by Newell and Simon (1976, p. 116): "A physical symbol system consists of a set of entities, called symbols, which are physical patterns that can occur as components of another type of entity called an expression (or symbol structure). Thus, a symbol structure is composed of a number of instances (or tokens) of symbols related in some physical way (such as one token being next to another). At any instant of time the system will contain a collection of these symbol structures. Besides these structures, the system also contains a collection of processes that operate on expressions to produce other expressions: processes of creation, modification, reproduction and destruction. A physical symbol system is a machine that produces through time an evolving collection of symbol structures. Such a system exists in a world of objects wider than just these symbolic expressions themselves. …An expression designates an object if, given the expression, the system can either affect the object itself or behave in ways dependent on the object. … The system can interpret an expression if the expression designates a process and if, given the expression, the system can carry out the process." In short, a physical symbol system is a computer.

[48] "…in order to be able to refer separately to the general category of possible count tags and the subset of such tags which constitute the traditional count words[, w]e call the former *numerons*; the latter *numerlogs*. Numerons are any distinct and arbitrary tags that a mind (human or nonhuman) uses in enumerating a set of objects. Numerlogs are the count words of a language" (Gelman & Gallistel, 1986, pp. 76-77).

[49] A referee suggested that the views of Pattee (1969, 1982), Rosen (1987), and Cariani (2001) may also be relevant, though I have my doubts. In any case, detailed discussion of their views is beyond the scope of the present essay, so a few words will have to suffice. As I emphasized in §2, I am not seeking to understand the (human) *brain* (as they are); I am seeking to understand *mind*, and not only the *human* mind, but mind more generally and abstractly, such that both cognitive computers and humans can be said to have minds. This requires an emphasis on symbols rather than matter, but must be accompanied by a study of implementation (or

realization)—the ways in which symbols can be "grounded"—and its converse (as Pattee notes). For my thoughts on Harnad's (1990) theory of symbol grounding, in the context of syntactic semantics, see Rapaport (1995). For my thoughts on the relation between implementation and semantics, see Rapaport (1999, 2005b). Briefly, as I read Pattee (1969), he supports what I call syntacticism, and his 1982 notion of semantic closure seems to describe a symbolic (i.e., syntactic) system whose semantic interpretation is the system itself—but that would make it fully syntactic. Rosen offers two reasons why syntax is not sufficient for semantics: The first is a variation on the Lucas-Penrose argument from Gödel's Incompleteness Theorem. This argument is highly debatable, but one simple response in the present context is to note that it might well be irrelevant: Gödel's Incompleteness Theorem does not say that there are true but unprovable statements of arithmetic (where truth is a matter of semantics and provability is a matter of syntax; see note 3 for what is meant by 'arithmetic'). What it says, rather, is that there is an arithmetic statement S such that neither S nor $\neg S$ can be *proved*. Semantics and truth do not have to enter the picture. Of course, because S can be interpreted as *saying* that S is unprovable, and because Gödel's Incompleteness Theorem proves that, in fact, S *is* unprovable, it follows by a trivial conception of "truth" that S is true. Rosen's second reason is that biological systems are "complex" and hence their "behavior…cannot be fully captured by any syntactic means" (Rosen, 1987, p. 14). But this depends on unprovable, metaphorical analogies between syntax and physical reductionism and between causation and implication. Moreover, Rosen (1987, p. 2) misunderstands Kleene when he disapprovingly quotes Kleene as saying that, in formal, axiomatic systems, meanings are "left out of account". What Kleene means is, not to *eliminate* meaning altogether, but that the goal of formal axiomatics is to *capture* all meaning syntactically, i.e., to remove it from the *symbols* (i.e., to turn the symbols into "marks", as I used the term in §5.1) and to move it (by explicitly re-encoding it) to the (syntactic) axioms. Finally, although Cariani (2001, p. 71) says, "One cannot replace semantics with syntactics", in fact one *can*, *if* the "world states" (Cariani, 2001, p. 70) are *also* symbolized; this is my point about internalism (§3.1). Oddly, Cariani (2001) also says, "Contra Fodor and Putnam, meaning can and does lie in the head". I quite agree; that's part of syntacticism! Fodor (1980) also agrees about the location of meaning, so I fail to understand why Cariani places Fodor and Putnam in the same camp. As Incardona (personal communication, 2011) observed, the real issue might be that I need to explain more clearly than I try in §3.1 "how syntacticism justifies the relationship between cognitive agents and the world. [But] internalism does not mean that cognitive agents cannot transform the 'outside world'." Readers who want such an explanation from me should read my papers cited in §3.

[50] These ideas were first suggested to me by Shoshana Hardt Loeb (personal communication, ca. 1983).

[51] See discussion and citations in Edelman (2011). Edelman says, "The *phenomenal experience* that arises from th[e] dynamics [of the anatomy and physiology of dreaming] is that of the dream self, situated in a dream world" (p. 3, emphasis added). So Fetzer could be charitably interpreted as meaning, not that dreams (hence minds) are not computable, but that *dream phenomenology* is not computable, i.e., that the "hard problem" of consciousness is, indeed, hard. But this raises a host of other issues that go beyond the scope of this paper. For some relevant remarks, see Rapaport (2005b, §2.3).

[52] But *simulated* animals *simulatedly* die!—WJR

[53] But see my comments about simulated hurricanes!—WJR

[54] Cf. IBM's *Jeopardy*-winning Watson.—WJR

[55] See my Web page "What Is Hypercomputation?" for a partial bibliography. Retrieved November 2, 2011, from http://www.cse.buffalo.edu/~rapaport/584/hypercompn.html

William J. Rapaport (PhD, Philosophy, Indiana University, 1976; MS, Computer Science, SUNY Buffalo, 1984) is an Associate Professor in the Department of Computer Science and Engineering, an Affiliated Faculty member in Philosophy and in Linguistics, and a member of the Center for Cognitive Science, all at State University of New York at Buffalo. His research interests are in semantics, cognitive science, and philosophy of computer science.

APPENDIX

What is an Algorithm?

Before anyone attempted to define 'algorithm', many algorithms were in use both by mathematicians as well as by ordinary people (e.g., Euclid's procedures for construction of geometric objects by compass and straightedge—see Toussaint, 1993; Euclid's algorithm for computing the GCD of two integers; the algorithms for simple arithmetic with Hindu-Arabic numerals–see Crossley & Henry, 1990, Wu, 2011; cf. Knuth, 1972). When David Hilbert investigated the foundations of mathematics, his followers began to try to make the notion of algorithm precise, beginning with discussions of "effectively calculable", a phrase first used by Jacques Herbrand in 1931 (Gandy, 1988, p. 68) and later taken up by Church (1936) and Stephen Kleene (1952), but left largely undefined, at least in print.

J. Barkley Rosser (1939, p. 55) made an effort to clarify the contribution of the modifier "effective" (emphasis added):

"'Effective method' is here used in the rather special sense of a method each step of which is [1] ***precisely predetermined*** *and which is [2]* ***certain to produce the answer*** *[3] in a* ***finite number of steps****."*

But what, exactly, does 'precisely predetermined' mean? And does 'finite number of steps' mean that the written statement of the algorithm has a finite number of instructions, or that, when executing them, only a finite number of tasks must be performed (i.e., what gets counted: written steps or executed instructions? One written step—"for i:= 1 to 100 do x:= x + 1"—can result in 100 executed instructions.)?

Much later, *after* Turing's, Church's, Gödel's, and Post's precise formulations and *during* the age of computers and computer programming, Markov, Kleene, and Knuth also gave slightly less vague, though still informal, characterizations.

According to Markov (1954/1960, p. 1), an algorithm is a "computational process" satisfying three (informal) properties: (1) being "determined" ("carried out according to a precise prescription…leaving no possibility of arbitrary choice, and in the known sense generally understood"), (2) having "applicability" ("The possibility of starting from original given objects which can vary within known limits"), and (3) having "effectiveness" ("The tendency of the algorithm to obtain a certain result, finally obtained for appropriate original given objects"). These are a bit obscure: Being "determined" may be akin to Rosser's "precisely predetermined". But what about being "applicable"? Perhaps this simply means that an algorithm must not be limited to converting one, specific input to an output, but must be more general. And Markov's notion of "effectiveness" seems restricted to only the second part of Rosser's notion, namely, that of "producing the answer". There is no mention of finiteness, unless that is implied by being computational.

In his undergraduate-level, logic textbook, Kleene (1967) elaborated on the notions of "effective" and "algorithm" that he had left unspecified in his earlier, classic, graduate-level treatise on metamathematics. He continues to identify "effective procedure" with "algorithm" (Kleene, 1967, p. 231), but now he offers a characterization of an algorithm as (1) a "procedure" (i.e., a "finite" "set of rules or instructions") that (2) "in a finite number of steps" answers a question, where (3) each instruction can be "followed" "mechanically, like robots; no insight or ingenuity or invention is required", (4) each instruction "tell[s] us what to do next", and (5) the algorithm "enable[s] us to recognize when the steps come to an end" (Kleene, 1967, p. 223).

Knuth (1973, pp. 1-9) goes into considerably more detail, albeit still informally. He says that an algorithm is "a finite set of rules which gives a sequence of operations for solving a specific type of problem", with "five important features" (Knuth, 1973, p. 4):

1. "*Finiteness.* An algorithm must always terminate after a finite number of steps" (Knuth, 1973, p. 4). Note the double finiteness: A finite number of rules in the text of the algorithm *and* a finite number of steps to be carried out. Moreover, algorithms must halt. (Halting is not guaranteed by finiteness; see point 5, below.) Interestingly, Knuth also says that an algorithm is a finite "computational method", where a "computational method" is a "procedure" that only has the next four features (Knuth, 1973, p. 4).
2. "*Definiteness.* Each step…must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified…" (Knuth, 1973, p. 5). This seems to be Knuth's analogue of the "precision" that Rosser and Markov mention.
3. "*Input.* An algorithm has zero or more inputs" (Knuth, 1973, p. 5). Curiously, only Knuth and Markov seem to mention this explicitly, with Markov's "applicability" property suggesting that there must be at least *one* input. Why does Knuth say *zero* or more? Presumably, he wants to allow for the possibility of a program that simply outputs some information. On the other hand, if algorithms are procedures for computing functions, and if functions are "regular" sets of input-output pairs (regular in the sense that the same input is always associated with the same output), then algorithms would always have to have input. Perhaps Knuth has in mind the possibility of the input being internally stored in the computer rather than having to be obtained from the external environment.
4. "*Output.* An algorithm has one or more outputs" (Knuth, 1973, p. 5). That there must be at least one output echoes Rosser's property (2) ("certain to produce the answer") and Markov's notion (3) of "effectiveness" ("a certain result"). But Knuth characterizes outputs as "quantities which have a specified relation to the inputs" (Knuth, 1973, p. 5); the "relation" would no doubt be the functional relation between inputs and outputs, but if there is no input, what kind of a relation would the output be in? Very curious!
5. "*Effectiveness.* This means that all of the operations to be performed in the algorithm must be sufficiently basic that they can in principle be done exactly and in a finite length of time by a man [*sic*] using pencil and paper" (Knuth, 1973, p. 6). Note, first, how the term 'effective' has many different meanings among all these characterizations of "algorithm", ranging from it being an unexplained term, through being synonymous with 'algorithm', to naming very particular—and very different—properties of algorithms. Second, it is not clear how Knuth's notion of effectiveness differs from his notion of definiteness; both seem to have to do with the preciseness of the operations. Third, Knuth brings in another notion of finiteness: finiteness in time. Note that an instruction to carry out an infinite sequence of steps in a finite time could be accomplished in principle by doing each step twice as fast as the previous step; or each step might only take a finite amount of time, but the number of steps required might take longer than the expected life of the universe (as in computing a perfect, non-losing strategy in chess; see §6.1). These may have interesting theoretical implications (see the vast literature on hypercomputation),[55] but do not seem very practical. Knuth (1973, p. 7) observes that "we want *good* algorithms in some loosely-defined aesthetic sense. One criterion of goodness is the length of time taken to perform the algorithm…." Finally, the "gold standard" of "a [hu]man using pencil and paper" seems clearly to be an allusion to Turing's (1936) analysis (see §5.2.2).

We can summarize these informal observations as follows: An algorithm (for executor *E* to accomplish goal *G*) is:

- A procedure (or "method")—i.e., a finite set (or sequence) of statements (or rules, or instructions)—such that each statement is:
  - Composed of a finite number of symbols (or marks) from a finite alphabet
  - And unambiguous for *E*—i.e.,
    - *E* knows how to do it
    - *E* can do it
    - it can be done in a finite amount of time
    - and, after doing it, *E* knows what to do next—
- And the procedure takes a finite amount of time, i.e., halts,
- And it ends with *G* accomplished.

But the important thing to note is that the more one tries to make precise these *informal* requirements for something to be an algorithm, the more one recapitulates Turing's motivation for the formulation of a Turing machine! Turing (1936, esp. §9) describes in excruciating detail what the minimal requirements are for a human to compute:

"[T]he computation is carried out on one-dimensional paper, i.e., on a tape divided into squares. …The behaviour of the computer [i.e., the human who computes!—WJR] at any moment is determined by the symbols which he [sic] is observing, and his "state of mind" at that moment. … [T]here is a bound B to the number of symbols or squares which the computer can observe at one moment. …[A]lso…the number of states of mind which need be taken into account is finite. … [T]he operations performed by the computer…[are] split up into "simple operations" which are so elementary that it is not easy to imagine them further divided. Every such operation consists of some change of the physical system consisting of the computer and his tape. We know the state of the system if we know the sequence of symbols on the tape, which of these are observed by the computer…, and the state of mind of the computer. …[I]n a simple operation not more than one symbol is altered. …[T]he squares whose symbols are changed are always "observed" squares. …[T]he simple operations must include changes of distribution of observed squares. …[E]ach of the new observed squares is within L squares of an immediately previously observed square. …The most general single operation must therefore be taken to be one of the following: (A) A possible change…of symbol together with a possible change of state of mind. (B) A possible change…of observed squares, together with a possible change of mind. The operation actually performed is determined… by the state of mind of the computer and the observed symbols. In particular, they determine the state of mind of the computer after the operation is carried out. We may now construct a machine to do the work of this computer." (Turing, 1936, pp. 249-251)

The "machine", of course, is what is now known as a "Turing machine"; it "does the work of this" *human* computer. Hence, my claim (§2), that the Turing machine was the first AI program.