HOW MINDS CAN BE COMPUTATIONAL SYSTEMS

William J. Rapaport

Department of Computer Science, Department of Philosophy, and Center for Cognitive Science
State University of New York at Buffalo, Buffalo, NY 14260

`rapaport@cs.buffalo.edu`
`http://www.cs.buffalo.edu/pub/WWW/faculty/rapaport/`

October 10, 1997

Abstract

The proper treatment of computationalism, as the thesis that cognition is computable, is presented and defended. Some arguments of James H. Fetzer against computationalism are examined and found wanting, and his positive theory of minds as semiotic systems is shown to be consistent with computationalism. An objection is raised to an argument of Selmer Bringsjord against one strand of computationalism, viz., that Turing-Test–passing artifacts are persons; it is argued that, whether or not this objection holds, such artifacts will inevitably be persons. (This paper is forthcoming in *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, Vol. 10 (1998). The present document is Department of Computer Science *Technical Report* 96-10 and Center for Cognitive Science *Technical Report* 96-1.)

1 THE PROPER TREATMENT OF COMPUTATIONALISM.

Computationalism is—or ought to be—the thesis that cognition is computable. But what does this mean? What is meant by 'cognition' and by 'computable'?

I take the vague term 'cognition' to cover the phenomena that others have called, equally vaguely, 'mental states and processes', 'thinking', 'intelligence', 'mentality', or simply 'the mind'. More specifically, this includes such things as language use, reasoning, conceiving, perceiving, planning, and so on—the topics of such cognitive disciplines as linguistics, cognitive psychology, and the philosophy of mind, among others—in general, of cognitive science. Perhaps this is still vague, but it will be good enough for present purposes.

A slightly more precise story has to be told about 'computable'. Note, first, that I have said that computationalism is the thesis that cognition is comput*able*, not that it is computa*tion* (as Pylyshyn 1985: xiii characterizes it). There is a subtle but, I think, important difference, as we shall see. The kind of thing that can be computable is a function, i.e., a set of ordered pairs—"input–output" pairs, to use computer jargon—such that no two pairs have the same first element but different second elements. Roughly, a *function* is computable if and only if there is an "algorithm" that computes it, i.e., an algorithm that takes as input the first elements of the function's ordered pairs, manipulates them (in certain constrained ways), and returns the appropriate second elements.

To say that it is an *algorithm* that does this is to say that there is an explicitly given, "effective procedure" for converting the input into the output. But what does *this* mean? Following my colleague Stuart C. Shapiro, I tell my introductory computer science students that an algorithm is a procedure for solving a problem such that: (a) it is unambiguous for the computer or human who will execute it; i.e., all steps of the procedure must be clear and well-defined for the executor, and (b) it is effective; i.e., it must eventually halt and it must be correct. ((a) and (b) are the constraints alluded to above.) However, to be mathematically precise, an algorithm is—following Church's Thesis—a Turing-machine program (or a recursive function, or a lambda expression, or any one of the other, logically equivalent models of computation; for an excellent historical discussion of the interrelationships of these concepts, see Soare 1996). We'll return to this notion in more detail, below.

Thus, to say that cognition is computable is to say that there is an algorithm—more likely, a collection of interrelated algorithms—that computes it. So, what does it mean to say that something "computes cognition"? If mental (or psychological) behavior can be characterized in input–output terms as, perhaps, stimulus–response pairs, then—assuming the set of such pairs is a function (or several functions)—cognition is computable if and only if there is an algorithm (or a collection of algorithms) that computes this function (or functions). Another way to say this is to say that cognition is a recursive function (or a set of recursive functions) (cf. Shapiro 1992: 54, 1995: 517).

But note that it is quite possible, according to this characterization of computationalism, for cognition itself—for mental states and processes, or brain states and processes—not to *be* a computation, i.e., not to *be* the execution of an algorithm. After all, a computable function need not be given as a computation: It might just be a set of input–output pairs. Arguably, such a set is a trivial computation: a table look-up. But there are other phenomena that are, or may be, computable but not computations. One standard kind of example is illustrated by the solar system, which, arguably, computes Kepler's laws. However, it could also be said that it is Kepler's laws that are computable and that describe the behavior of the solar system, yet the solar system doesn't compute them; i.e., the behavior of the solar system isn't a computation, even though its behavior is computable. In theory, a device could convert input into output in some mysterious, or causal but not computational, way, yet be such that the function that is its input–output description is computable. Perhaps the mind works this way: Mental states and processes—i.e., cognition—may be comput*able* but not comput*ed*. And similarly for the brain. That is, possibly there are minds—i.e., systems that have the input–output behavior of cognition—that accomplish it *non*-algorithmically (and maybe human minds are like this.) Personally, I do not believe that this is the case, but it is possible—and it is consistent with computationalism properly treated.

However, if cognition *is* thus comput*able*, then any (physical) device that *did* perform cognitive computations would exhibit cognition. It *would* think. And *it* would do so even if *we* (human) cognitive agents did *not* perform cognitive *computations*.

So, is computationalism true? I don't *know*, but I believe so. For one thing, I have not seen any good arguments against it—people such as Dreyfus, Penrose, and Searle notwithstanding. It is not that these people are necessarily wrong. It is just that I do not think that this is the sort of issue that is ready to be refuted by an in-principle argument. It may well be the case that some—maybe even all—aspects of cognition are not computable. But I take the goal of computational cognitive science to be the investigation of the extent to which—and the ways in which—cognition *is* computable. And I take computationalism—understood now as the thesis that *all* cognition is computable—to be its working hypothesis. In fact, much of cognition *is known to be* computable:

Most reasoning is—including first-order, default, and non-monotonic logics, belief revision, as well as searching and game playing. Much of language is. Significant aspects of visual perception, planning, and acting are. What we should do is to see how much of cognition can be computed. The history of computational cognitive science is much too short for us to give up on it yet.

Is computationalism as I have characterized it trivial? Quite the contrary, for it needs to be *demonstrated* that the cognitive functions are computable, and to do this, we need to devise appropriate algorithms. This is by no means a trivial task, as the history of AI has shown. The central question of cognitive science, as I and many other computational cognitive scientists see it, is not (merely): How is human cognition accomplished? Human cognition may, for all we know, not be accomplished computationally. Rather, the central question is the more Kantian one: How is cognition possible? Computationalism properly treated is the hypothesis that cognition can be accomplished computationally, i.e., that it is computable.

2 FETZER'S TREATMENT OF COMPUTATIONALISM.

In "Mental Algorithms" (1994), James H. Fetzer asks whether minds are computational systems, and there, as well as in "People Are Not Computers" (1996),[1] he answers in the negative. I believe that there are (or will be, in the golden age of cognitive science) some computational systems that are minds, and I believe that mentality (cognition) is comput*able*. But, as we have seen, the computationalist should also be willing to believe that it is possible that not all minds are computational systems *if* by that is meant that possibly not all minds *behave* computationally.

Thus, it is misleading for Fetzer to say that "the idea that human thought *requires* the execution of mental algorithms appears to provide a foundation for research programs in cognitive science" (Fetzer 1994: 1; my italics). Rather, what provides the foundation is the idea that thought in general (and human thought in particular) can be explained (or described, or accounted for) in terms of algorithms (call them 'mental algorithms' if you will). As I noted, this is, or can be seen as, an elaboration of behaviorism: *Behaviorism* was concerned with describing human thought in stimulus–response terms (i.e., input–output terms) and only those. *Cognitivism* posits processes that mediate the stimulus (input) with the response (output). And *computational* cognitivism posits that those processes are computable (cf. Rapaport 1993).

Let's consider the notion of an algorithm in a bit more detail. It is difficult to talk about what an algorithm is, since the notion is an informal one. What licenses its use is Church's Thesis, the fact that all formal explications of the informal notion have turned out to be logically equivalent, thus giving support to its being more than just an intuitively plausible idea. I offered one informal specification above, due to Shapiro. Fetzer offers two variants: (1) "algorithms ... are definite (you always get an answer), reliable (you always get a correct answer), and completable (you always get a correct answer in a finite interval of time)" (Fetzer 1994: 4)[2] and (2) "algorithms [are] completely reliable ... procedures ... that can be carried out in a finite number of steps to solve a problem" (Fetzer 1996: 9, 14). In all three cases, the informal notion of algorithm is a relative one: An algorithm is an algorithm *for* a problem or question. Thus, if an alleged algorithm for

[1] An interesting title given that, at the time that Turing wrote his original paper on what are now called 'Turing machines', a "computer" was a *human* who did computations for a living!

[2] Shapiro, in conversation, has pointed out to me that interactive algorithms, such as those used by our Cassie system for natural-language interaction with a (human) user (Shapiro & Rapaport 1987, 1995; Rapaport 1988; Shapiro 1989), are not reliable or completable, yet they are clearly algorithmic.

some problem P fails to solve P correctly, it fails to be an algorithm *for* P. However, it does not necessarily thereby fail to be an *algorithm*, for it may be an algorithm for some other problem P'.

This is where the notion of a "heuristic" enters. Consider an AI program that uses heuristic game-playing techniques to play chess. It may not always make the "best" move, but if it makes a move that's "good enough", that will suffice for the purposes of playing chess. Yet it is an *algorithm* that does this; not the algorithm you thought it was, perhaps, but an algorithm nonetheless. A *heuristic for problem* P can be defined as an *algorithm* for some problem P', where the solution to P' is "good enough" as a solution to P. (Cf. Korf 1992, Shapiro 1992: 54–55, Findler 1993, Herman 1993, Korfhage 1993.) Thus, to argue, as Fetzer (1994: 6) does, that computationalism is false to the extent that it relies on heuristics "rather" than algorithms is to set up a false dichotomy: The heuristics that AI researchers and computational cognitive scientists use *are* algorithms. In fact, in AI we don't need guaranteed solutions at all, just algorithmic processes that are cognitive. It is best to stick to the Turing-machine analysis of "algorithm" (i.e., of computation) and omit any reference to the problem for which an algorithm is designed. What is important for computationalism properly treated is whether cognitive processes are algorithmic (i.e., computable) in the Turing-machine sense.

Are there functions that are intuitively algorithmic but are not recursive or Turing-computable? Fetzer cites Carol E. Cleland (1993), who "appeals to a class of 'mundane functions', which includes recipes for cooking and directions for assembling devices, as examples of effective procedures that are not Turing computable", because they manipulate "things rather than numerals" (Fetzer 1994: 15). However, this takes the standard introduction-to-computer-science analogy for an algorithm and tries to make more of it than is there. The computationalist's point is not that cooking, e.g., is algorithmic in the sense that the recipe is an algorithm to be "followed" (and, incidentally, computers don't "follow" algorithms; they *execute* them) but that cooking is the *result* of algorithmic processes: I can write a very complex program for a robot who will plan and execute the preparation of dinner in an algorithmic fashion, even using "pinches" of salt, but the recipe is not an algorithm. A suitably-programmed Turing machine *can* cook (or so the working hypothesis of computational cognitive science would have it).

But is *all* of cognition algorithmic? Fetzer makes two claims that reveal, I think, a serious misunderstanding of the computational cognitive enterprise:

> The strongest possible version of the computational conception would therefore appear to incorporate the following claims: that all thinking is reasoning; that all reasoning is reckoning; that all reckoning is computation; and that the boundaries of computability are the boundaries of thought. Thus understood, the thesis is elegant and precise, but it also appears to suffer from at least one fatal flaw: it is (fairly obviously, I think) untrue! The boundaries of thought are vastly broader than those of reasoning, as the exercise of imagination and conjecture demonstrates. Dreams and daydreams are conspicuous examples of non-computational thought processes. (Fetzer 1994: 5; cf. Fetzer 1996: 17–18, 22.)

The first claim, that all thinking is reasoning, is a red herring. I agree with Fetzer that "the boundaries of thought are vastly broader than those of *reasoning*": When I recall a pleasant afternoon spent playing with my son, I am not *reasoning* (there are no premises or conclusions), but I *am* thinking, and—the computationalist maintains—my thinking is computable. In this case, my recollection (my reverie, if you will) is the result of a computable (i.e., algorithmic) mental process.

However, a slight amendment to Fetzer's slippery slope makes me willing to slide down it: Although *not* all thinking is *reasoning*, all reasoning *is* reckoning, all *thinking* (*including* reckoning) is comput*able*, and the boundaries of computability *are* the boundaries of thought.

This last point is also disputed by Fetzer, who says that "computability does not define the boundaries of thought. The execution of mental algorithms appears to be no more than one special kind of thinking" (Fetzer 1994: 2). On the working hypothesis that all thinking is computable, computability *does* define the boundaries of thought, and even if mental algorithms are *just* "a special kind of thinking," they are an *important* kind, because systems other than humans can execute them, and these systems can thus be said to think. But the computationalist should, as noted, be willing to recognize the possibility that actual thinkings, even though comput*able*, might not take place by computation.[3] In *that* sense, I can agree with Fetzer.[4] But this is merely to say that some acts of thinking might be comput*able* but not carried out computationally.

The notion of "boundaries", however, is a slippery one. An analogy might clarify this. The distance between locations A and B might be "drivable"—i.e., one can get from A to B by driving. But it might also be "walkable"—i.e., one can get from A to B by walking (cf. McCarthy 1968/1985: 300). If any two locations are drivable *but also walkable*, then in one sense drivability does *not* define the boundaries of getting from any A to any B, because it's *also* walkable. Yet, in another sense, drivability *does* define the boundaries: Any two locations *are* drivable.

The second mistaken claim concerns dreaming. To say that "dreams are not computational" (Fetzer 1996: 18) because they themselves do not compute any (interesting) functions, or because they are not heuristics, or because "they have no definite starting point and no definite stopping point" (Fetzer 1996: 18) is beside the point. The point is that (or whether) there are computational processes that can *result* in dreams. In fact, this point holds for all mental states and processes: The question is not whether a particular mental state or process is itself an algorithm that computes something, but whether there are algorithms that result in that mental state or process.

Dreams, in any case, are a bad example, since neuroscientists aren't really sure what they are or what purposes they serve. My understanding is that they result from possibly random neuron firings that take place when we sleep and that are interpreted *by us as if* they were due to external causes.[5] Suppose for the sake of argument that this is the case. Then, insofar as our ordinary interpretations of neuron firings in non-dreamlike situations are computable, so are dreams.

What about "a certain look, a friendly smile, a familiar scent [that] can trigger enormously varied associations of thoughts under what appear to be the same relevant conditions" (Fetzer 1994: 13) or "the associative character of ordinary thought" (as exemplified in the stream-of-

[3] Although, on Occam's-razor–like grounds, any such non-computational thinking might be so marginal as to be eliminable.

[4] I might even be able to agree with Penrose: Penrose's arguments, as I understand them, are of the form: *If* mental phenomena are quantum processes, then they are not algorithmic (Penrose 1989; cf. Fetzer 1994: 10). It is important to remember, however, that the antecedent has not been established, not even by Penrose. Even so, if mental phenomena are comput*able*, then even if they are not comput*ed*, perhaps because they are quantum phenomena, computationalism wins.

[5] Note the need to posit the brain's (or the mind's) ability to be partitioned into syntax-like neuron firings that are interpreted by semantic-like neuron firings. But it's all just neuron firings, i.e., syntax (see below). Also note the methodologically solipsistic nature of this theory.

consciousness "Cornish Game Clams" example; Fetzer 1996: 32–34)? Fetzer says "that these thought processes do not satisfy the computational conception and therefore properly count against it" (Fetzer 1996: 34). Again, however, what computationalism properly treated says is, not that such thoughts *are algorithms*, but that they *can be* the *results* of algorithms. Programs like Racter arguably behave in this associative way, yet are computational, and spreading-activation theories of associationist thinking can account for this behavior computationally (cf. Quillian 1967). And, of course, "what *appear* to be the same relevant conditions" may in fact be *different* ones.

Instead of algorithms, some writers on computationalism, such as John Haugeland (1985), talk about "automatic formal systems"—essentially, syntactic, i.e., symbol-manipulation, systems (cf. Fetzer 1994: 2–3). Fetzer says that "What turns a purely formal system into a cognitive system ... is the existence of an 'interpretation' in relation to which the well-formed formulae, axioms, and theorems of that formal system become meaningful and either true or false" (Fetzer 1994: 2–3). That's *one* way of turning a syntactic system into a cognitive one. But it's important to see that this is an external, third-person attribution of cognition to the system, for it's an external agent that provides the interpretation (cf. Rapaport 1988). This is one aspect of Daniel Dennett's (1971) "intentional stance".

But another way of turning a syntactic system into a cognitive one—as I have argued in "Syntactic Semantics" and "Understanding Understanding" (Rapaport 1988, 1995)—is to ensure that the formal system is sufficiently rich and has some input–output connections with the external world. (The "some" hedge is to allow for cases of "handicapped" humans; cf. Maloney 1987 and 1989, Ch. 5; Shapiro 1995: 521–522.) Such a rich syntactic system need have no interpretation externally imposed on it. Syntax can give rise to semantics of a holistic, conceptual-role variety. Most likely, some of the formal system's internal symbols (or terms of its language of thought, or nodes of its semantic network) would be internal representations of external entities, causally produced therefrom by perception. And others of its internal symbols (or terms, or nodes) would be concepts of those perceptually-produced symbols. As I argue in "Understanding Understanding" (Rapaport 1995), these would be the system's internal interpretations of the other symbols. This would be the system's first-person "interpretation" of its own symbols.

Fetzer would probably not agree with my analysis of syntax and semantics. Unfortunately, he does not provide arguments for claims such as the following:

> When ... marks [that form the basis for the operation of a causal system without having any meaning for the system] are envisioned as *syntax*, ... they are viewed as the ... bearers of meaning, which presupposes a point of view. In this sense, syntax *is* relative to an interpretation, interpreter or mind.
>
> It is the potential to sustain an interpretation that qualifies marks as elements of a formal system .... (Fetzer 1994: 14.)

But *why* is syntax thus relative? *Who* "views" the marks as "bearers of meaning"? And why do the marks of a formal system need "the potential to sustain an interpretation"? The only way I'll grant this without argument is if we allow the system itself to provide its own interpretation, by allowing it to map some marks into others, which are "understood" by the system in some primitive way. This process of self-interpretation, however, turns out to be purely syntactic and computable. (See Rapaport 1995 for elaboration.)

Let me now turn to Fetzer's positive theory, that minds are not computational systems

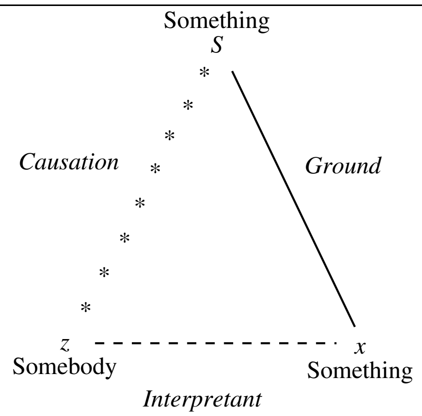

Figure 1: Fetzer's diagram of a semiotic system (from Fetzer 1996: 27).

but *semiotic* systems (Fetzer 1994: 17). A semiotic system, according to Fetzer, is something "for which something can stand for something else in some respect or other" (Fetzer 1994: 17). Thus, a semiotic system consists of three entities and three relations: The three entities are a sign, a sign user (e.g., a mind), and what the sign stands for. The sign could be either an "icon" (which resembles what it stands for), an "index" (which causes or is an effect of what it stands for), or else a "symbol" (which is conventionally associated with what it stands for). None are "merely syntactical marks", i.e., " 'symbols' in the computational sense" (Fetzer 1994: 17). As for the three relations, the sign and what it stands for participate in a "grounding" relation; the sign and its user participate in a "causal" relation; and all three participate in an "interpretant" relation.

If minds are semiotic systems, and semiotic systems are not computational, then neither are minds. Fetzer gives two reasons for the second premise: [1] "*the same sign* may be variously viewed as an icon ..., as an index ..., or as a symbol ..., and ... [2] *inductive reasoning* employing heuristics ... , which are usually reliable but by no means effective procedures, appears to be fundamental to our survival" (Fetzer 1994: 17–18). But, in the first case, I fail to see what the ambiguity of signs has to do with not being computational. And, in the second case, heuristics *are*, as we have seen, effective procedures. In addition, computational theories of inductive reasoning are a major research area in AI (cf. Angluin & Smith 1992; Muggleton & Page, forthcoming). So what are the details of Fetzer's arguments?

Consider Fetzer's Figure 1 (Fetzer 1994: 19), reproduced here as Figure 1. It raises more questions than it answers: What is the causation relation between sign-user z and sign S? Which causes which? What is the grounding relation between sign S and thing x that S stands for? The diagram suggests that sign S is grounded by thing x that it stands for, which, in turn, suggests that the two binary relations "is grounded by" and "stands for" are the same. But Fetzer says that sign S stands for thing x *for* sign-user z, which is a 3-place relation that is not shown or easily showable in Figure 1. What is shown instead is a binary "interpretant" relation between sign-user z and thing x; yet earlier we were told that the interpretant relation was a 3-place one. I offer a slightly clearer diagram in Figure 2.

So why are semiotic systems and computational systems disjoint?

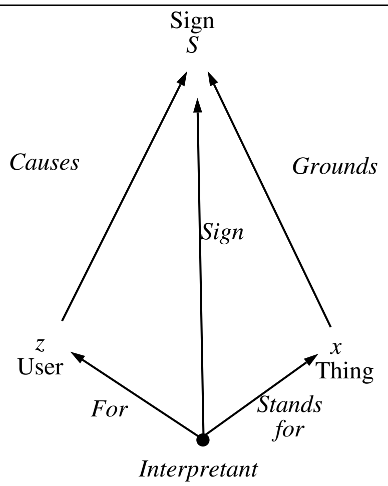

Figure 2: Another diagram of a semiotic system: sign-user z causes sign S; thing x grounds sign S; and the 3-place interpretant relation is that sign S stands for thing x for sign-user z.

> ... the marks that are manipulated by means of programs might be meaningful for the *users* of that system ... but are not therefore meaningful for use **by** that system itself. (Fetzer 1996: 10, my boldface; cf. Fetzer 1994: 18.)

But why are they not meaningful for the system? I have argued in "Syntactic Semantics" and "Understanding Understanding" that with enough structure, they *can* be meaningful for the system. For example, consider a program that (a) computes the greatest common divisor of two natural numbers, (b) that has a "knowledge base" of information about arithmetic and about what a greatest common divisor is, and (c) that has natural-language competence (i.e., the ability to interact with the user in natural language; cf. Shapiro & Rapaport 1991). Such an AI program can be asked what it's doing and how it does it; it can answer that it is computing greatest common divisors, and it can explain what they are and how it computes them, in exactly the same sort of way that a human student who has just learned how to compute them can answer these questions. Not only does the user of such a system ascribe an interpretation *to* it, according to which the *user* says that the system is computing greatest common divisors, but the system *itself* can be said to "understand" what it's doing. It could even turn around and ascribe *to the user* an interpretation of the *user's* greatest-common-divisor–computing behavior!

Such a possibility shows that it is incorrect to say, as Fetzer does, that "the marks ... are not ... meaningful for use by that system itself". Fetzer says that this is because the "grounding relationship between these marks and that for which they stand" is absent (Fetzer 1994: 18). But why is it absent in, say, my greatest-common-divisor–computing computational agent but not in the human student who computes greatest common divisors? And does Fetzer really think that it is the *grounding* relation that is absent, or rather the relation of sign S standing for thing x *for computer* z that is absent? It would seem that the point he wants to make is that although sign S might stand for thing x, it does not do so for the computer z, but only for a human user of the computer.

At this point, Fetzer refers to his Figure 2 (reproduced here as Figure 3), which is supposed

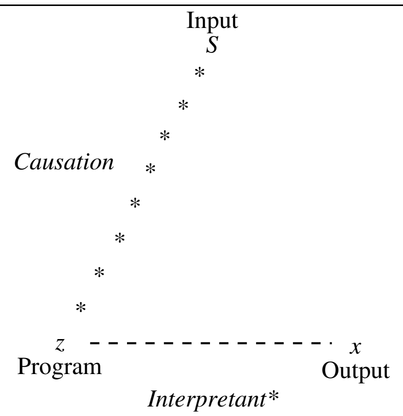

Figure 3: Fetzer's diagram of a (computational) symbol system (from Fetzer 1996: 27).

to be like Figure 1 except that the grounding relation between sign S and thing x is missing. But there are other differences: Sign S is now called an 'input', thing x is called an 'output', and sign-user z is called a program—presumably, a program for a mental algorithm. But who ever said that "the marks by means of which" digital computers "operate" are only the input? Or that what they stand for are the output? (And, anyway, what does it mean for sign S to stand for thing x yet not be grounded by x?)

Consider Cassie, a computational cognitive agent implemented in the SNePS knowledge-representation and reasoning system by our research group at SUNY Buffalo (Shapiro & Rapaport 1987). As a reader of narratives, Cassie's input is English sentences (Shapiro & Rapaport 1995). Her output is other English sentences expressing her understanding of the input or answering a question that was input. The marks by means of which Cassie operates are not these sentences, but the nodes of her internal semantic-network "mind". They are not the input (though they may be causally produced by the input).

Do these marks mean anything? Yes: Some of them stand for, or are grounded by, whatever in the external world caused them to be built. But Cassie can have no direct access to such external things, so this isn't what they mean *for her*. Do they mean anything for Cassie? Again, yes: Their meaning for her is their location in Cassie's entire semantic network. Insofar as some of these marks are directly caused by external objects, and others are concepts of those marks, then those others stand in a grounding relation to the directly caused ones. But all of this is internal to—and meaningful to—Cassie (cf. Ehrlich 1995, Rapaport 1995). And it is all syntactic symbol manipulation. And it is all computable.

I conclude that computationalism properly treated withstands Fetzer's objections and that the semiotic approach is consistent with it.

3 BRINGSJORD'S TREATMENT OF COMPUTATIONALISM.

In "Computationalism Is Dead; Now What?" (1996), Selmer Bringsjord analyzes computationalism into four postulates and a theorem:

Computationalism consists of the following four propositions.

CTT A function f is effectively computable if and only if f is Turing-computable.

P=aTM Persons are Turing machines.

TT The Turing Test is valid.

P-BUILD Computationalists will succeed in building persons.

TT-BUILD Computationalists will succeed in building Turing Test-passing artifacts. (This proposition is presumably entailed by its predecessor.)

(Bringsjord 1996: 6.)[6]

As a true believer in computationalism properly treated, I accept all five, with one small caveat about **P=aTM**: I would prefer to say, not that *persons* are Turing machines, but that *minds* are—even better, that cognition is comput*able*; in fact, putting it this way links it more closely to Bringsjord's **CTT**.

Bringsjord, however, denies **P-BUILD**. (Since he accepts **TT-BUILD**, it follows that he must reject **P=aTM**.) He does not, however, offer us a definition of 'person'. In Chapter IX, "Introspection", of his book, *What Robots Can and Can't Be* (Bringsjord 1992), to which he refers us for the defense of his denial of **P-BUILD**, he argues that persons satisfy "hyper-weak incorrigibilism" but that robots cannot. I shall argue that robots *can* satisfy this feature, and I will conclude with some observations on the nature of personhood that suggest a more general way in which **P-BUILD** can be defended.

Hyper-weak incorrigibilism is essentially the following thesis (Bringsjord 1992: 333–335):

1. Let F be a contingent property (i.e., one such that it's possible for something to have it and possible for something (else) to lack it).
2. Let s be a cognitive agent.
3. Let $\mathbf{C}'$ be a set of "psychological" or "Cartesian" properties (such as *being sad*, *being in pain*, *being happy*, or *seeming to see a ghost*).
4. Let $\mathbf{C}''$ be defined as follows: If $F' \in \mathbf{C}'$, then *seeming to have* $F' \in \mathbf{C}''$.
5. Suppose $F \in \mathbf{C}''$ (i.e., F is of the form: *seeming to have* F', for some $F' \in \mathbf{C}'$; thus *seeming to be sad*, *seeming to be in pain*, *seeming to be happy* (and *seeming to seem to see a ghost*?) are all in $\mathbf{C}''$).
6. Then it is necessary that if s believes that Fs, then Fs (i.e., it is necessary that if you believe that you seem to have F', then you seem to have F').

Now, Bringsjord's argument concerns (1) "logicist cognitive engineering" (as opposed to connectionist cognitive engineering), where cognitive engineering is roughly the interdisciplinary attempt to build a (computational) cognitive agent, and (2) a proposition he calls $\mathbf{AI_{LOGFOUND}}$, viz:

[6] Page references are to the version of 21 May 1996.

> If [logicist cognitive engineering] is going to succeed, then the robots to be eventually produced by this research effort will be such that
>
> (i) if there is some significant mental property G that persons have, these robots must also have G;
>
> (ii) the objects of their "beliefs" (hopes, fears, etc.)—the objects of their *propositional attitudes*—are represented by formulas of some symbol system, and these formulas will be present in these robots' knowledge bases; and
>
> (iii) they will be physical instantiations of automata (the physical substrate of which will be something *like* current silicon hardware, but may be something as extravagant as optically-based parallel hardware).
>
> (Bringsjord 1992: 330–331.)

He uses hyper-weak incorrigibilism to refute $\mathbf{AI_{LOGFOUND}}$ as follows: From the claim that hyper-weak incorrigibilism is a significant mental property that persons have (which I will not for the moment deny) and from (i), "it follows ... that the flashy robot (call it 'r') to be eventually produced by Cognitive Engineering will be able to introspect infallibly with respect to $\mathbf{C}''$" (Bringsjord 1992: 341). Therefore, by instantiating hyper-weak incorrigibilism to robot r, we get:

(1) $$\forall F[(F \text{ is contingent} \wedge F \in \mathbf{C}'') \supset \Box(B_r Fr \supset Fr)].$$

By (ii), this implies:

(2) $$\forall F[(F \text{ is contingent} \wedge F \in \mathbf{C}'') \supset \Box((\ll Fr \gg \ \in D(\mathcal{B})) \supset Fr)],$$

where: $\ll Fr \gg$ denotes the first-order formula corresponding to the proposition 'Fr', $\mathcal{B}$ is a set of first-order formulas that r "believes initially", and $D(\mathcal{B}) = \{\alpha \mid \mathcal{B} \vdash \alpha\}$ is the robot's knowledge base (Bringsjord 1992: 340). *The underlying assumption here is that for robot* r *to believe a proposition is for a first-order representation of the proposition to be an element of robot* r*'s knowledge base*, or, as it's sometimes called, the robot's "belief box".

Next, let $F^* \in \mathbf{C}''$; Bringsjord suggests taking *seeming to be in pain* as F^*. Hence,

(3) $$\Box[(\ll F^* r \gg \ \in D(\mathcal{B})) \supset F^* r].$$

(E.g., necessarily, if the formula 'r seems to be in pain' is in robot r's belief box, then r seems to be in pain.)

Suppose that $\neg F^* r$, e.g., that it is *not* the case that the robot seems to be in pain. (Bringsjord doesn't specify this assumption, but I think that it is needed for what follows.) Now, "since it's physically possible that the hardware substrate of r fail, and since, in turn, it's physically possible that this failure be the cause of $\ll F^* r \gg \ \in D(\mathcal{B})$" (Bringsjord 1992: 342), we have:

(4) $$\Diamond(\ll F^* r \gg \ \in D(\mathcal{B}) \wedge \neg F^* r),$$

which contradicts (3). Therefore, $\mathbf{AI_{LOGFOUND}}$ is false. Therefore, logicist cognitive engineering will fail.

There are a number of questions I have about this argument. For one thing, what does it mean for a cognitive agent to *seem* to have a property? It could mean that it seems *to someone* *else* that the agent has the property; e.g., it might seem to *me* that the robot is in pain if it *acts* (as if) in pain. But for *seeming* to have a property to make sense in hyper-weak incorrigibilism, I think it has to mean that it seems *to the agent* that the agent itself has the property.[7] But, then, what does this mean for the robot? Does it mean that there is a "seeming box" such that if the first-order formula expressing that the robot has the property is in the seeming-box, then it seems to the robot that it has the property? Not only does this make *seeming* to have a property much like *believing* oneself to have it, but I suggest (without argument) that that's just what it is. At any rate, let's suppose so for the sake of argument.[8] Then the brunt of hyper-weak incorrigibilism is this:

(5) $$\Box(B_r B_r F'r \supset B_r F'r),$$

where $F' \in \mathbf{C}'$; e.g., necessarily, if the robot believes that it believes itself to be in pain, then it believes itself to be in pain.

This leads to another problem I have with Bringsjord's argument: The assumption underlying the inference to the belief-box version of hyper-weak incorrigibilism (2) is somewhat simplistic. Sophisticated computational cognitive agents need not equate belief in a proposition P with the mere presence of a representation of P in the knowledge base. For example, for reasons independent of the present considerations (see Shapiro & Rapaport 1992; Rapaport, Shapiro, & Wiebe, forthcoming), we represent Cassie's believing a proposition in two distinct ways: One way is by "asserting" the node representing the propositional object of her mental act of believing. This is a "flag" that *is* roughly equivalent to placing the node in a "belief box", as in the (simplified) semantic-network representation using the assertion flag of Cassie's belief that John is rich, as shown in Figure 4. The other way is to represent explicitly that Cassie herself believes the proposition in question, as in the (simplified) semantic-network representation using an explicit "I (Cassie) believe that" operator, as shown in Figure 5. One reason for allowing these two different representations is that we want to be able to represent that Cassie *believes* that she believes something even if she believes that it's *not* the case. This can be done roughly as shown in Figure 6. Here, Cassie believes that she herself believes `M1`, but she doesn't: `M2` represents her believing that she herself believes `M1`; `M3` represents her belief that ¬`M1`.

But now, given our interpretation of seeming to be in pain (say) as believing oneself to be in pain, what happens on the simplistic belief-box theory? The consequent of (5) becomes:

(6) $$\ll Fr \gg \ \in D(\mathcal{B});$$

i.e., a formula representing the proposition that r is in pain is in r's belief box. But what is the antecedent of (5)? The question is: How does one represent nested beliefs in a belief-box theory? Bringsjord says that we need an epistemic logic and that $\ll B_r Fr \gg \ \in D(\mathcal{B})$—but then how does $\ll B_r Fr \gg$ being in $D(\mathcal{B})$ relate to $\ll Fr \gg$ being in $D(\mathcal{B})$? Bringsjord doesn't say, so let me speculate that, in fact, nested beliefs collapse: $B_r B_r Fr$ just becomes $\ll Fr \gg \ \in D(\mathcal{B})$. But then (5) is a tautology. Hence, any robot satisfies hyper-weak incorrigibilism.

[7]This use of 'the agent itself' is a "quasi-indicator"; see Castañeda 1966; Rapaport 1986; Rapaport, Shapiro, & Wiebe, forthcoming.

[8]In discussion, Bringsjord has offered the following counterexample to my identification of seeming to have a property with believing oneself to have it: In the case of optical illusions in which appearance differs from reality, such as the Müller–Lyer illusion, in which two lines of equal length appear to have different lengths, it seems to me that the lines have different lengths even while it is the case that I sincerely believe them to have the same length. I agree that this shows that seeming to have a property can differ from believing oneself to have it. However, arguably this is not a case of what Bringsjord calls 'Cartesian' properties. For Cartesian properties, I think that seeming to have a property *is* believing oneself to have it.

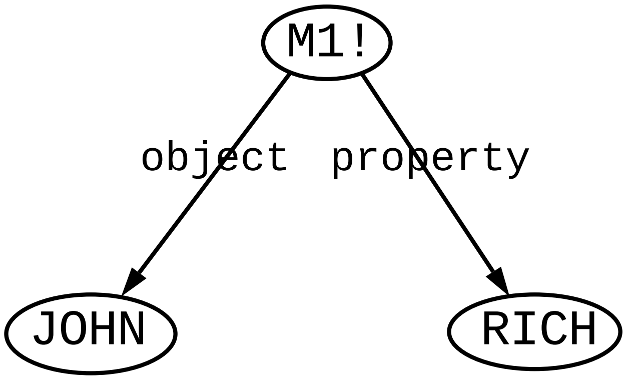

Figure 4: `M1` is a simplified SNePS representation of the proposition that John is rich, using an assertion flag (`!`) to indicate that Cassie believes it.

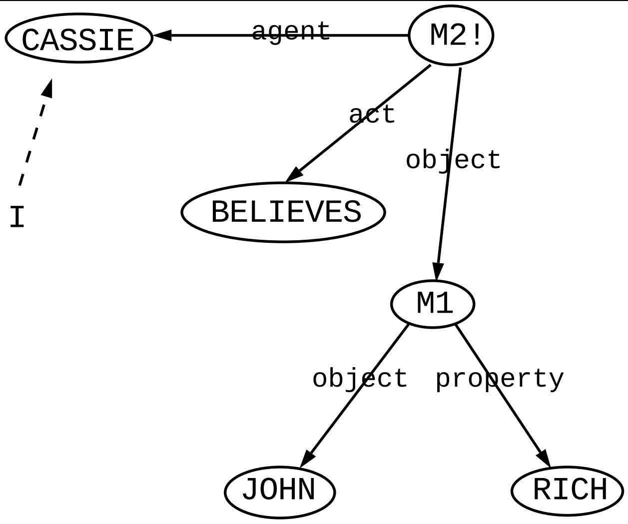

Figure 5: `M2` is a simplified SNePS representation of the proposition that Cassie believes `M1`. The "I-pointer" to the node that represents Cassie indicates that the node labeled '`Cassie`' is her self-concept. She would express `M2` as "I believe that John is rich". Since `M2` is asserted, Cassie believes that she (herself) believes that John is rich. If `M1` were asserted, then Cassie would believe that John is rich. But `M1` is not asserted, so it's not the case that Cassie believes that John is rich (even though she believes that she believes it). (See Rapaport, Shapiro, & Wiebe (forthcoming) for more on the "I-pointer".)

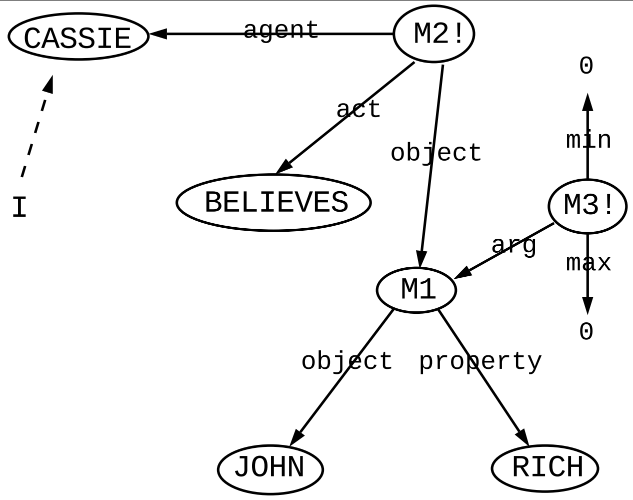

Figure 6: `M3` represents the proposition that ¬`M1`. The `min-max-arg` case frame says that at least 0 and at most 0 of the propositions pointed to by the `arg` arc are true. `M3` is asserted, so Cassie believes ¬`M1`. Since `M2` is also asserted, Cassie believes that she believes `M1`, but she also believes that ¬`M1`. (See Shapiro & Rapaport 1987, Martins & Shapiro 1988 for details.)

Let me trade on the kind words of Bringsjord in his footnote 3, in which he praises our work on Cassie, and suggest how Cassie would fare in this case. First, hyper-weak incorrigibilism would have to be part of her "unconscious" processing mechanism, much as her reasoning ability is—i.e., hyper-weak incorrigibilism wouldn't be encoded explicitly as a "conscious" belief but would govern the way she thinks. Second, what it would mean is this: Whenever Cassie believed (via the assertion flag) that she believed (represented via the "I (Cassie) believe that" operator) that she is in pain (say), then she would believe (via the assertion flag) that she *was* in pain—i.e., asserting "Cassie believes that she herself is in pain" would automatically assert that Cassie was in pain. In Figure 7, hyper-weak incorrigibilism would mean that if `M12` were asserted, then `M10` would automatically also be asserted. Now suppose that Cassie believes (via assertion) that she is *not* in pain (`M11` in Figure 7). And suppose that a hardware failure asserts a Cassie-believes-that belief that she herself *is* in pain (`M12` in Figure 7). Hyper-weak incorrigibilism would then assert that she is in pain (i.e., it would cause an assertion operator to be applied to `M10` in Figure 7), contradicting her explicit (asserted) belief that she is not in pain; i.e., `M10` and `M11`, both asserted, are inconsistent.

What actually happens with Cassie? *At this point, the SNePS belief-revision system* (SNeBR; Martins & Shapiro 1988, Martins & Cravo 1991, Cravo & Martins 1993, Ehrlich 1995) *is invoked,* alerting Cassie to the fact that her beliefs are inconsistent, *and she would have to give one of them up*. Whichever one she gives up will maintain hyper-weak incorrigibilism, for either she will correct the hardware failure and unassert `M12`, the Cassie-believes-that belief that she is in pain, or she will decide that she *is* in pain after all.

Thus, I don't think that Bringsjord has made his case that $\mathbf{AI_{LOGFOUND}}$ and therefore logicist cognitive engineering fail, and hence that hyper-weak incorrigibilism rules out Turing-Test–passing artifacts from personhood. Moreover, why couldn't an analogous neural hardware

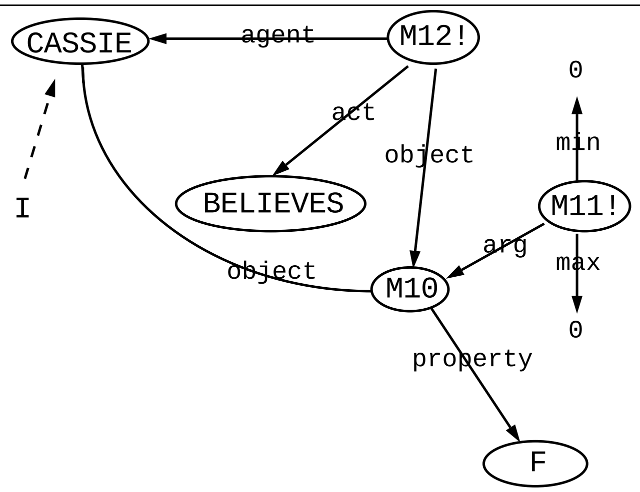

Figure 7: M10 represents that Cassie is F; M12 represents that Cassie believes that she herself is F; M11 represents ¬M10. By hyper-weak incorrigibilism, if M12 is asserted—i.e., if Cassie believes M12; i.e., if Cassie believes that she herself believes that she herself is F; i.e., if Cassie believes that she herself seems to be F—then M10 must be asserted—i.e., Cassie believes that she herself is F; i.e., Cassie seems to be F. If M11 is asserted, then Cassie believes ¬M10; i.e., Cassie believes that she herself is not F; i.e., Cassie does not seem to be F. If both M10 and M11 are asserted, then Cassie has inconsistent beliefs, and SNeBR will be invoked (see text) (see text).

(i.e., brain) failure cause a similar problem for us? Maybe hyper-weak incorrigibilism isn't a necessary feature of persons after all. (Indeed, if hyper-weak incorrigibilism were implemented in Cassie's mind as an explicit "rule" that she believed, then, when she realized that her beliefs were inconsistent, she could simply reject *it* instead of either M12 or M10!)

In any case, I reject clause (ii) of **AI$_{\text{LOGFOUND}}$** (the clause that posits belief boxes). Replacing it with a more sophisticated belief-representation mechanism can, I believe, save computationalism. At the very least, it shows that **P-BUILD** has not yet been refuted.

But suppose Bringsjord can strengthen his argument to meet my objections. I still think that **P-BUILD** is true, for an entirely different—and more general—reason. Suppose that **TT-BUILD** is true (as both Bringsjord and I hold). I maintain that such Turing-Test–passing artifacts will be such that we will, *in fact* treat them morally as if they were persons—i.e., they will have the moral and perhaps legal status of persons (cf. Asimov 1976, Rapaport 1988). That is, our concept of *person* will be broadened to include not only human persons, but non-human computational ones. In much the same way that our concept of *mind* can be broadened to include not only human minds but other animal and computational ones, thus making *Mind* something like an abstract data type implemented in humans, other animals, and computers, so we will come to see an "abstract data type" *Person* as implemented in both humans and robots. (Legally, corporations already implement it; cf. Willick 1985 and Rapaport 1988, §4.2.) I maintain, that is, that it would be morally wrong to harm an entity that, in virtue of passing the Turing Test, we accept as being intelligent, *even if philosophers like Fetzer, Bringsjord, and Searle are right about propositions like* **P-BUILD**. We already consider it morally wrong to harm non-human animals, and the more intelligent (the more human-like?) the animal, the more it seems to be morally wrong. Surely we would owe intelligent robots no less.[9]

4 REFERENCES.

1. Angluin, Dana, & Smith, Carl H. (1992), "Inductive Inference," in Stuart C. Shapiro (ed.), *Encyclopedia of Artificial Intelligence, 2nd Edition* (New York: John Wiley & Sons): 672–682.
2. Asimov, Isaac (1976), "The Bicentennial Man," in Isaac Asimov, *The Bicentennial Man and Other Stories* (Garden City, NY: Doubleday): 135–172.
3. Bringsjord, Selmer (1992), *What Robots Can and Can't Be* (Dordrecht, The Netherlands: Kluwer Academic Publishers).
4. Bringsjord, Selmer (1996), "Computationalism Is Dead; Now What? Response to Fetzer's 'Minds Are Not Computers: (Most) Thought Processes Are Not Computational Procedures'," paper presented as part of an invited symposium, "Are Minds Computational Systems?", at the 88th Annual Meeting of the Southern Society the 88th Annual Meeting of the Southern Society for Philosophy and Psychology, Nashville, 5 April 1996; revised version (21 May 1996) available at [`http://www.rpi.edu/~brings/SELPAP/fetz.nash.ps`].
5. Castañeda, Hector-Neri (1966), " 'He': A Study in the Logic of Self-Consciousness," *Ratio* 8: 130–157.
6. Cleland, Carol E. (1993), "Is the Church–Turing Thesis True?", *Minds and Machines* 3: 283–312.
7. Cravo, Maria R., & Martins, João P. (1993), "SNePSwD: A Newcomer to the SNePS Family", *Journal of Experimental and Theoretical Artificial Intelligence* 5: 135–148.
8. Dennett, Daniel C. (1971), "Intentional Systems," *Journal of Philosophy* 68: 87–106; reprinted in Daniel C. Dennett, *Brainstorms* (Montgomery, VT: Bradford Books): 3–22.
9. Ehrlich, Karen (1995), "Automatic Vocabulary Expansion through Narrative Context," *Technical Report 95-09* (Buffalo: SUNY Buffalo Department of Computer Science).
10. Fetzer, James H. (1994), "Mental Algorithms: Are Minds Computational Systems?", *Pragmatics and Cognition* 2: 1–29.
11. Fetzer, James H. (1996), "People Are Not Computers: (Most) Thought Processes Are Not Computational Procedures," paper presented as part of an invited symposium, "Are Minds Computational Systems?", at the 88th Annual Meeting of the Southern Society the 88th Annual Meeting of the Southern Society for Philosophy and Psychology, Nashville, 5 April 1996.

[9]An ancestor of this paper was presented as part of an invited symposium, "Are Minds Computational Systems?", at the 88th Annual Meeting of the Southern Society for Philosophy and Psychology, Nashville, 5 April 1996, with other papers by James H. Fetzer and Selmer Bringsjord. I am grateful to Fetzer for organizing the symposium and to Bringsjord, Fetzer, and members of the SNePS Research Group for comments on earlier versions.

12. Findler, Nicholas V. (1993), "Heuristic," in Anthony Ralston & Edwin D. Reilly (eds.), *Encyclopedia of Computer Science, 3rd Edition*, (New York: Van Nostrand Reinhold): 611–612.

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

14. Herman, Gabor T. (1993), "Algorithms, Theory of," in Anthony Ralston & Edwin D. Reilly (eds.), *Encyclopedia of Computer Science, 3rd Edition*, (New York: Van Nostrand Reinhold): 37–39.

15. Korf, Richard E. (1992), "Heuristics," in Stuart C. Shapiro (ed.), *Encyclopedia of Artificial Intelligence, 2nd Edition* (New York: John Wiley & Sons): 611–615.

16. Korfhage, Robert R. (1993), "Algorithm," in Anthony Ralston & Edwin D. Reilly (eds.), *Encyclopedia of Computer Science, 3rd Edition*, (New York: Van Nostrand Reinhold): 27–29.

17. Maloney, J. Christopher (1987), "The Right Stuff: The Mundane Matter of Mind," *Synthese* 70: 349–372.

18. Maloney, J. Christopher (1989), *The Mundane Matter of the Mental Language* (Cambridge, UK: Cambridge University Press).

19. Martins, João P., & Cravo Maria R. (1991), "How to Change Your Mind," *Noûs* 25: 537–551.

20. Martins, João, & Shapiro, Stuart C. (1988), "A Model for Belief Revision," *Artificial Intelligence* 35: 25–79.

21. McCarthy, John (1968), "Programs with Common Sense," in Marvin Minsky (ed.), *Semantic Information Processing* (Cambridge, MA: MIT Press): 403–418; reprinted in Ronald J. Brachman & Hector J. Levesque (eds.), *Readings in Knowledge Representation* (Los Altos, CA: Morgan Kaufmann, 1985): 299–307.

22. Muggleton, Stephen, & Page, David (forthcoming), Special Issue on Inductive Logic Programming, *Journal of Logic Programming*.

23. Penrose, Roger (1989), *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics* (Oxford: Oxford University Press).

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

25. Quillian, M. Ross (1967), "Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities," *Behavioral Science* 12: 410–430; reprinted in Ronald J. Brachman & Hector J. Levesque (eds.), *Readings in Knowledge Representation* (Los Altos, CA: Morgan Kaufmann, 1985): 97–118.

26. Rapaport, William J. (1986), "Logical Foundations for Belief Representation," *Cognitive Science* 10: 371–422.

27. Rapaport, William J. (1988), "Syntactic Semantics: Foundations of Computational Natural-Language Understanding," in James H. Fetzer (ed.), *Aspects of Artificial Intelligence* (Dordrecht, Holland: Kluwer Academic Publishers): 81–131; reprinted in Eric Dietrich (ed.), *Thinking Computers and Virtual Persons: Essays on the Intentionality of Machines* (San Diego: Academic Press, 1994): 225–273.

28. Rapaport, William J. (1993), "Cognitive Science," in Anthony Ralston & Edwin D. Reilly (eds.), *Encyclopedia of Computer Science, 3rd Edition*, (New York: Van Nostrand Reinhold): 185–189.

29. Rapaport, William J. (1995), "Understanding Understanding: Syntactic Semantics and Computational Cognition," in James E. Tomberlin (ed.), *Philosophical Perspectives*, Vol. 9: *AI, Connectionism, and Philosophical Psychology* (Atascadero, CA: Ridgeview): 49–88.

30. Rapaport, William J.; Shapiro, Stuart C.; & Wiebe, Janyce M. (forthcoming), "Quasi-Indexicals and Knowledge Reports," *Cognitive Science*; preprinted as *Technical Report 95-49B* (Buffalo: SUNY Buffalo Department of Computer Science, 1995) and *Technical Report 95-17* (Buffalo: SUNY Buffalo Center for Cognitive Science, 1995).

31. Shapiro, Stuart C. (1989), "The CASSIE Projects: An Approach to Natural Language Competence," *Proceedings of the 4th Portuguese Conference on Artificial Intelligence (Lisbon)* (Springer-Verlag): 362–380.

32. Shapiro, Stuart C. (1992), "Artificial Intelligence," in Stuart C. Shapiro (ed.), *Encyclopedia of Artificial Intelligence, 2nd Edition* (New York: John Wiley & Sons): 54–57; revised version appears in Anthony Ralston & Edwin D. Reilly (eds.), *Encyclopedia of Computer Science, 3rd Edition*, (New York: Van Nostrand Reinhold): 87–90.

33. Shapiro, Stuart C. (1995), "Computationalism," *Minds and Machines* 5: 517–524.

34. Shapiro, Stuart C., & Rapaport, William J. (1987), "SNePS Considered as a Fully Intensional Propositional Semantic Network," in Nick Cercone & Gordon McCalla (eds.), *The Knowledge Frontier: Essays in the Representation of Knowledge* (New York: Springer-Verlag): 262–315.

35. Shapiro, Stuart C., & Rapaport, William J. (1991), "Models and Minds: Knowledge Representation for Natural-Language Competence," in Robert Cummins & John Pollock (eds.), *Philosophy and AI: Essays at the Interface* (Cambridge, MA: MIT Press): 215–259.

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

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

38. Soare, Robert I. (1996), "Computability and Recursion," *Bulletin of Symbolic Logic*, forthcoming.

39. Willick, Marshal S. (1985), "Constitutional Law and Artificial Intelligence: The Potential Legal Recognition of Computers as 'Persons'," *Proceedings of the of the 9th International Joint Conference on Artificial Intelligence (IJCAI–85, Los Angeles)* (Los Altos, CA: Morgan Kaufmann): 1271–1273.