Does a Rock Implement Every Finite-State Automaton?

David J. Chalmers

Abstract

Putnam has argued that computational functionalism cannot serve as a foundation for the study of the mind, as every ordinary open physical system implements every finite-state automaton. I argue that Putnam's argument fails, but that it points out the need for a better understanding of the bridge between the theory of computation and the theory of physical systems: the relation of implementation. It also raises questions about the classes of automata that can serve as a basis for understanding the mind. I develop an account of implementation, linked to an appropriate class of automata, such that the requirement that a system implement a given automaton places a very strong constraint on the system. This clears the way for computation to play a central role in the analysis of mind.

1 Introduction

The theory of computation is often thought to underwrite the theory of mind. In cognitive science, it is widely believed that intelligent behavior is enabled by the fact that the mind or the brain implements some abstract automaton: perhaps a Turing machine, a program, an abstract neural network, or a finite-state automaton. The ambitions of artificial intelligence rest on a related claim of computational sufficiency, holding that there is a class of automata such that any implementation of an automaton in that class will possess a mind. A similar claim is often made about many specific mental properties, including properties characteristic of human mentality: that is, it is claimed that there exists a class of automata such that any implementation of an automaton in that class will have the mental property in question. In this way, it is hoped that computation will provide a powerful formalism for the replication and explanation of mentality.

In an appendix to his book *Representation and Reality* (Putnam 1988, pp. 120-125), Hilary Putnam argues for a conclusion that would destroy these ambitions. Specifically, he claims that *every ordinary open system realizes every abstract finite automaton.* He puts this forward as a theorem, and offers a detailed proof. If this is right, a simple system such as a rock implements any automaton one might imagine. Together with the thesis of computational sufficiency, this would imply that a rock has a mind, and possesses many properties characteristic of human mentality. If Putnam's result is correct, then, we must either embrace an extreme form of panpsychism or reject the principle on which the hopes of artificial intelligence rest. Putnam himself takes the result to show that computational functionalism cannot provide a foundation for a theory of mind.[1]

In what follows I will argue that Putnam's argument does not establish such a conclusion, and that the foundations of artificial intelligence are not in danger. His argument nevertheless points to the need for a better understanding of implementation, the relation that connects the theory of computation with the theory of physical systems. (Putnam calls this relation 'realization'.) An analysis of Putnam's argument leads us to a better understanding of implementation, as well as to insights into the class of automata that are relevant in computational theories of cognition. As objections are raised and modifications are made, the discussion that follows will touch on a number of formalisms in the theory of computation and various constraints on implementation, before settling on a formalism and an account of implementation that avoid the most serious problems.

2 Putnam's Argument

Putnam precedes his proof with discussion of some physical principles. There is a Principle of Physical Continuity, which is not important for our purposes (it plays a minor role in the argument in an application I will accept), and a more central Principle of Noncyclical Behavior. The latter principle holds that every ordinary system is in different maximal states at different times. (A "maximal" state is a total state of the system, specifying the system's physical makeup in perfect detail). Putnam argues for this on the basis that every such system is exposed to electromagnetic and gravitational radiation from a natural clock. I will accept the principle in the discussion that follows. Even if it does not hold across the board (arguably, signals from a number of sources might cancel each other's effects, leading to a cycle in behavior), the more limited result that every noncyclic system implements every finite-state automaton would still be a strong one.

Putnam is centrally concerned with finite-state automata, or FSAs, and in particular FSAs without input or output, or *inputless FSAs*. (He extends his results to FSAs with inputs and outputs later.) Such an FSA is specified by a set of formal states $\{S_1, \ldots, S_n\}$, and by a set of state-transition relations which specify for each state the state that must follow. He does not offer an explicit account of the conditions under which a physical system implements an FSA, but from his discussion it is clear that he requires something like the following.

> A physical system implements an inputless FSA in a given time-period if there is a mapping f from physical states of the system to formal states of the FSA such that: if the system is in physical state p during the time-period, this causes it to transit into a state q such that formal state $f(p)$ transits to formal state $f(q)$ in the specification of the FSA.

I will later argue that this account is not quite right, but it captures the central facet of the notion of implementation: the requirement that in order for a physical system to implement a formal automaton, causal relations among states of a physical system must mirror formal relations among states of the automaton.

Putnam considers an arbitrary system S—say, a rock—over an arbitrary time period, such as from 12:00 to 12:07. Using the expression $St(S, t)$ to denote the maximal state of S at t, he defines states of S in the following way.

> Let the beginnings of the intervals during which S is to be in one of its stages [...] be $t_1, t_2, \ldots, t_n$ (in the example given, $n = 7$, and the times in question are $t_1 = 12:00$, $t_2 = 12:01$, $t_3 = 12:02$, $t_4 = 12:03$, $t_5 = 12:04$,

> $t_6 = 12:05$, $t_7 = 12:06$). The end of the real-time interval during which we wish S to "obey" this table we call t_{n+1} ($= t_8 = 12:07$, in our example). For each of the states t_i to $t_{i+1}, i = 1, 2, \ldots, n$, define a (nonmaximal) *interval state* s_i which is the "region" in phase space consisting of all the maximal states $St(S, t)$ with $t_i \leq t < t_{i+1}$. (I.e., S is in s_i just in case S is in one of the maximal states in this "region". Note that the system S is in s_1 from t_1 to t_2, in s_2 from t_2 to $t_3, \ldots$, in s_n from t_n to t_{n+1}. (Left endpoint included in all cases but not the right—this is a convention to ensure the "machine" is in exactly one of the s_i at a given time.) The disjointness of the states s_i is guaranteed by the Principle of Noncyclical Behavior. (pp. 122-123.)

The states are defined, then, so that the system goes through states $s_1, s_2, \ldots, s_n$ in succession, in time intervals following $t_1, t_2, \ldots, t_n$ respectively.

We can choose an arbitrary (inputless) FSA for the purposes of the argument. Putnam chooses (claiming no loss of generality) an FSA that transits back-and-forth between two states A and B, which is therefore required to go through the sequence $ABABABA$ in the time-period in question.

The argument that the system implements the FSA is straightforward. We simply define physical state a to be the disjunction $s_1 \vee s_3 \vee s_5 \vee s_7$, and state b to be $s_2 \vee s_4 \vee s_6$, and we define the mapping f so that $f(a) = A$ and $f(b) = B$. (Putnam's presentation does not distinguish between physical states a and b and formal states A and B; I introduce the distinction for clarity.) It is easy to see that during the period in question, the physical system cycles through the states $abababa$, corresponding to formal states $ABABABA$ as required.

Putnam further argues that the system's being in state a *causes* it to transit into state b and vice versa, using the Principle of Continuity to show that given that the system was in state a at a time t with $t_{n-1} \leq t < t_n$ (and given boundary conditions at t), it was *determined* that the system would be in state b at times t with $t_n \leq t < t_{n+1}$. I will postpone discussion of this part of the argument.

The argument is general. Given any inputless FSA and any noncyclic physical system, we can map physical states of the system over an arbitrarily long period of time to formal states of the FSA by the same method. We associate the initial physical state of the system with an initial state of the FSA and associate subsequent physical states of the system with subsequent states of the FSA, where the FSA state-evolution is determined by its state-transition rules. The implementation mapping is determined by taking the disjunction of associated physical states for each state of the FSA, and mapping that disjunctive state to the FSA state in question. Under this mapping, it is easy to see that the evolution of the physical states precisely mirrors evolution of the FSA states. Therefore every noncyclic physical system implements every (inputless) FSA.

3 Analysis

Some object to this argument on the grounds that it requires "unnatural" physical states, involving arbitrary disjunctions. It is difficult to make this objection precise, however, as there does not seem to be an objective distinction between "natural" and "unnatural" states that can do the relevant work; and an objective distinction is needed, if having a mind is to be an objective matter. I will not pursue this line, as I think the problems lie elsewhere. It is also sometimes objected that Putnam employs "time-varying" states, but this is based on a misreading of the argument. The states themselves are quite invariant over time. Time has simply been used to fix reference to these atemporal states.

The problem, I think, is that Putnam's system does not satisfy the right kind of state-transition conditionals. The conditionals involved in the definition of implementation are not ordinary material conditionals, saying that on all those occasions in which the system happens to be in state p in the given time period, state q follows. Rather, these conditionals have modal force, and in particular are required to support counterfactuals: *if* the system were to be in state p, *then* it would transit into state q. This expresses the requirement that the connection between connected states must be *reliable* or *lawful*, and not simply a matter of happenstance. It is required that *however* the system comes to be in state p, it transits into state q (perhaps with some restriction ruling out extraordinary environmental circumstances; perhaps not). We can call this sort of conditional a *strong* conditional.

It is not quite clear whether Putnam intends his conditionals to have this sort of modal force. He requires that the relation be a *causal* relation, and goes to some lengths to argue that his construction indeed satisfies the causal relation by arguing that its being in state a fully determines its transition into state B, so he may have some such requirement in mind. In any case, I will argue that his system does not satisfy the strong conditionals in the way that implementation of an automaton requires. There are two ways in which Putnam's system fails to satisfy the strong conditionals. The first concerns the state-transitions that are actually exhibited. The second concerns unexhibited state-transitions.

To see the first point, consider the transition from state a to state b that the system actually exhibits. For the system to be a true implementation, this transition must be reliable. But Putnam's system is open to all sorts of environmental influences by its very definition as an open system (indeed, Putnam exploits this fact to establish his Principle of Noncyclical Behavior). It follows that if environmental circumstances had been slightly different, the system's behavior would have been quite different. Putnam's construction establishes nothing about the system's behavior under such circumstances. The construction is entirely specific to the environmental conditions as they were during the time-period in question. It follows that his construction does not satisfy the relevant strong conditionals. Although on this particular run the system happened to transit from state a to state b, if conditions had been slightly different it might have transited from state a to state c, or done something different again.

In particular, it is quite possible for Putnam's system to be in state a at all times between t_{n-1} and t_n, while failing to be in state b at all times between t_n and t_{n-1}. If environmental conditions had been slightly different at and after time t_n, the system might have been buffeted into an entirely different state well before t_{n+1}. The relevant strong conditionals are therefore not satisfied. It will not do to argue that the system counts as being in state

b if it is in that state at the *beginning* of the relevant time-interval. If so, then the same goes for state a, and a similar argument can be given. We only need to note the possibility of environmental buffeting soon after t_{n-1}, so that although the system counts as being in state a in the given interval, it is never in state b in the following interval.

Putnam argues (p. 123) that the system's being in state a throughout the first interval *determines* the transition into state b in the next interval, but his argument falls short of establishing what is required. In particular, he only establishes that:

> ...given the information that the system was in state A at t, and given that the boundary condition at t was B_t, a mathematically omniscient being can determine from the Principle of Continuity that the system S must have been in $St(S, t)$, and can further determine, given the boundary conditions at subsequent times and the other laws of nature, how S evolves in the whole time interval under consideration. (p. 123)

What is important here is the clause "given boundary conditions at subsequent times". This is illegitimate information for the omniscient being in question to use. What is required is that together with the laws of nature, the state (and perhaps boundary conditions) *during the first interval* determines the state during the next interval. It is illegitimate to appeal to the boundary conditions throughout the next interval. To be sure, if we can independently specify these conditions, then the next state will be entirely fixed, as there will be no room for external buffeting, but this is no more interesting than noting that if we fix the physical state during the next time interval, the physical state will be fixed. This argument therefore fails to establish what is required, and the strong conditional is unsatisfied.

The second failure of the conditionals is perhaps more interesting. To qualify as an implementation, it is not only required that the formal state-transitions that are manifested *on this run* (say, the transitions from A to B and back) are mirrored in the structure of the physical system. It is required that *every* state-transition be so mirrored, including the many that many not be manifested on a particular run. It may be, for instance, that the FSA in question is specified by the state-transitions $A \rightarrow B$, $B \rightarrow A$, $C \rightarrow D$, $D \rightarrow E$, and so on. In this case, we require that the unmanifested features of the machine-table are reflected in the physical structure of the system, so that it will be the case that *if* the system had been in state C, it would have transited to state D, and so on.

Perhaps it might be objected that there is little point having such extra states when the machine eternally cycles between two states and will never reach the other states. However, an FSA will often specify the machine's behavior for more than one initial state; and more importantly the fact that there is a cycle in the state-space is a contingent feature of Putnam's example, and his argument is supposed to be perfectly general. Generally, a finite-state machine that is of practical interest will not exhaust all its states in a given run, nor will it cycle.

The requirement that unmanifested state-transitions also be reflected in the physical structure of the machine requires a slight change to our definition of implementation. It should read:

> A physical system implements an inputless FSA in a given time-period if there is a mapping f from physical states of the system onto formal states of the FSA such that: for every formal state-transition $P \rightarrow Q$ in the specification of the FSA, if the physical system is in a state p such that $f(p) = P$, this causes it to transit into a state q such that $f(q) = Q$.

The changes are (1) the requirement that the mapping map physical states onto *all* states of the FSA, and (2) the requirement that a conditional be satisfied for each state-transition in the FSA. If these conditions are not satisfied, there is a very clear sense in which the FSA is not fully implemented.

It is clear that Putnam's construction does not satisfy this stronger condition. In fact, for any formal state P that does not appear in the sequence manifested in a given run, his construction leaves undefined the physical states that map onto P. Even if we pick arbitrary uninstantiated physical states to map onto P, there is no reason to believe that these will satisfy the requisite state-transition conditionals. (If the conditional in the definition of implementation is read as a *material* conditional, then it turns out that these conditionals are satisfied vacuously—every instance of p transits to q, because there are no instances of p. Again, the conditional needs to be read modally.)

To put the point slightly differently: from the fact that any system that implements a given FSA must go through a sequence $abababa$ in real time, the converse does not follow. We cannot infer that any system that goes through the sequence $abababa$ implements the FSA. In effect, Putnam has failed to ensure that his system reflects the structure of the FSA. All he has done is ensure that a particular *trace* of the FSA—that is, the states it goes through on a particular run—is reflected by the system's behavior.[2] But much more is required. There are all sorts of inherent possibilities in an FSA that are not reflected in a given trace.

In the body of his book (pp. 96-98), Putnam responds briefly to the charge that his construction does not satisfy a certain kind of counterfactual: in particular, the requirement that if the system *had not* been in state a, it would not have transited into state b (this is a requirement on causation according to the analysis in Lewis 1973). His response is that these counterfactuals are dependent on the vague notion of a similarity metric over possible worlds (at least on Lewis' analysis), and that the notions of similarity and possible worlds are even vaguer than what we are trying to explicate.

The counterfactuals that I require do not have this problem. All that my account of implementation requires is that *if* the system is in state a, it will transit into state b, however it finds itself in the first state. This is in no way a similarity-based counterfactual. It requires that in *all* (or perhaps most) situations in which the system is in state a, it will transit into state b, no matter how similar or dissimilar that situation is from the actual one. This class of counterfactuals is much less vague than the other kind, and is therefore not open to a similar objection. It simply corresponds to the requirement that the transition be lawful. What makes the law of gravity a law is that under *any* circumstances, if two bodies have an appropriate mass, there will be such-and-such a force exerted. If it only happens between 12:00 and 12:07 today, due to unusual environmental conditions, it does not count.

4 A Possible Reply

It turns out that the above objections can be met fairly easily. To ensure that the physical systems satisfies the relevant strong conditionals as well as the material conditionals, we need only require that the system satisfies a pair of undemanding constraints. It remains the case that problems with strong conditionals undermine Putnam's general thesis, but we will have to wait a while to see why.

To overcome the first objection, it is sufficient to require that the system *reliably* transits through a sequence of states $s_1, s_2, \ldots$, irrespective of environmental conditions. (To simplify, I will hereafter assume that time is discrete. Nothing will depend on this.) This is not a difficult requirement: most clocks satisfy it, for instance. Probably most physical systems satisfy such a requirement; perhaps we might find reliable sequences like this in patterns of radiation decay. In any case, let us say a physical system contains a *clock* if it has a subsystem that reliably transits through a sequence like this.

A system containing a clock will circumvent the first objection. If we define the states $s_1, s_2, \ldots$ of the system as those states containing the relevant states of the clock, then the transition from s_n to s_{n+1} will be reliable. If disjunctive states a, b, and so on are defined appropriately, then the transitions between these will satisfy the appropriate strong conditionals.

To reply to the second objection we need to make sure that the system has sufficient extra states to map onto formal states that are not manifested on a given run. We can do this by ensuring that the system contains a *dial*: that is, a subsystem with an arbitrary number of different states, such that when it is put into one of those states it stays in that state come what may. Most physical systems will contain a dial—think of states corresponding to marks on rocks, for instance.

Claim: Every physical system containing a clock and a dial will implement every inputless FSA.

Proof: Label physical states of the system $[i, j]$, where i corresponds to a clock state and j to a dial state. If the system starts in state $[1, j]$ it will reliably transit through states

$[2, j], [3, j]$ and so on.

Assume the physical system on the actual run has dial state 1. The initial state of the system will be $[1, 1]$; we associate this state with an initial state of the FSA. We associate states $[2, 1], [3, 1]$ and so on with subsequent states of the FSA in the obvious way. If at the end of this process some FSA states have not come up, choose an unmanifested state P, and associate state $[1, 2]$ with it. We associate states $[2, 2], [3, 2]$ and so on with the states that follow P in the evolution of the FSA. If this does not exhaust the states of the FSA, choose an unmanifested state and associate state $[3, 1]$ with it, and so on. Eventually all the states of the FSA will be exhausted. For each state of the FSA, there will be a non-empty set of associated physical states $[i, j]$. To obtain the implementation mapping, the disjunction of these states is mapped to the FSA state.

It is easy to see that this system satisfies all the strong conditionals in the strengthened definition of implementation. For every state of the FSA, if the system is (or were to be) in a state that maps onto that formal state, the system will (or would) transit into a state that maps onto the appropriate succeeding formal state. So the result is demonstrated.

There are minor problems with the states $[n, i]$ that come up at the end of the run, where t_n is the final time-step. We need to satisfy conditionals of the form "if it were in state $[n, i]$, it would transit appropriately"—that is, $[n, i]$ would transit to a physical state that maps onto a successor of $f([n, i])$—but we have established nothing about the physical successors to $[n, i]$. We can get around this problem by assuming the clock has an infinite sequence of states, mapping all the states in each sequence, and mapping an infinite disjunction onto each formal state. This is not an especially unrealistic assumption; there are probably such infinite clocks in most continuous systems.

Thus Putnam's result is preserved in only a slightly weakened form. But there is still no problem for the computationalist about the mind. All this has demonstrated is that inputless FSAs are an inappropriate formalism. The structure of an inputless FSA is quite trivial, and it is entirely the wrong sort of thing to describe or specify a mind. Inputless FSAs are almost never invoked in the theory of computation, precisely because of this triviality.

To see the triviality, note that the state-space of an inputless FSA will consist in a single unbranching sequence of states ending in a cycle, or at best in a finite number of such sequences. The latter possibility arises if there is no state from which every state is reachable. It is possible that the various sequences will join at some point, but this is as far as the "structure" of the state-space goes. This is an completely uninteresting kind of structure, as indeed is witnessed by the fact that it is satisfied by a simple combination of a dial and a clock.

This suggests that we need a more interesting formalism than inputless FSAs. In particular, to put stronger constraints on structure we need to move to FSAs with *input* and perhaps with output. The addition of input changes the formalism from trivial to non-trivial. Where there is input, there can be branching behavior. A formal state can be succeeded by various different formal states, depending on the input. Furthermore, the presence of input gives the formalism a kind of combinatorial structure. Later states depend not just on a single state, but on a combination of state and input.

This formalism is much more appropriate for capturing the dynamics of cognitive systems. Humans do not have a single path of states along which their lives are determined. Even if they do, as some fatalistic views might suggest, this path does not exhaust their description. For any given sequence of states that a human goes through, it remains the case that *if* things in the world had gone slightly differently, they would have functioned in an interestingly different way. Omitting this potentiality leaves out a vital part of the description of human functioning.[3] A wind-up toy or perhaps a videotape of my life could go through the same sequence of states, but it would not be a cognitive system. Cognition requires at least the possibility of functioning in more than one way.

It might be objected that there are or might be people who have effectively *no* capacity for input and outputs—blind, deaf paralytics for instance – who nevertheless have minds, and are cognitive systems in a quite reasonable sense. According to this objection, such people cannot be cognitive in virtue of implementing an automaton with input or output, for they have no such input or output themselves.

I think this would be to misdescribe the situation, however. These people can be seen to implement an automaton with input and output. It remains the case when they are in a given state that *if* they were to receive a given input, *then* their state would transit appropriately. It is just that they are currently not receiving any inputs (at best they are receiving a single "null" input) due to the malfunctioning of their sensory apparatus. Of course this means that the inputs in question have to be understood as stimulations of the optic nerve, or of the visual cortex or some other internal system, rather than as patterns of photons (mutatis mutandis for other sensory modalities), but this is no problem. It simply moves the boundary inward. The malfunctioning of their sensory system means that their visual cortex does not often get stimulated, but *if* it were to be stimulated, their internal state would change in interesting ways. We can deal with outputs in a similar way.

It is this very sensitivity or potentiality that demonstrates that there is a cognitive system present. If there were only one sequence of states along which a system could "work", it would not be a person but a wind-up toy. The sensitivity to change shows that real cognitive *mechanisms* are present.

An alternative way to handle this sort of case will be outlined later, in terms of automata with internal combinatorial structure. Either way, it turns out that structureless FSAs without input or output are inadequate for the specification of a cognitive system.

5 Finite-State Automata with Input and Output

Putnam extends his result to the case of FSAs with input and output. He notes that there is no hope of extending the universal realization result to this case. To implement an FSA with input and output, a physical system must at least have the capacity for input and output. Furthermore, it must have something like the right *kind* of input and output, and of input/output dependencies. Putnam notes (p. 124) that inputs and outputs to physical systems cannot map to inputs and outputs of automata arbitrarily, as is the case with internal states. There are generally restrictions on the mapping, depending on our purposes.

Instead, Putnam puts forward a more limited result that is still significant. He argues that a finite-state automaton with input and output is realized by every physical system with the right input/output dependencies. That is, if the physical system gets the input and output right, it gets the FSA right.

To see that this is still significant, note that there are vastly different FSAs with the same input/output pattern. Think of an FSA that outputs zero on every input, for instance, and an FSA that upon receiving an input n goes into an exhaustive calculation to determine whether the number is prime, outputs zero if it is prime, and zero if it is not (it also outputs zero on each step during its calculation). These FSAs have precisely the same input/output dependencies. Nevertheless, one would think that most simple systems that implement the first machine certainly do not implement the second, with its vastly more complex internal structure.

Putnam also notes that if this result is correct, then (computational) *functionalism* about mentality will imply *behaviorism* about mentality. If mentality is dependent on implementing the right automaton, and if an automaton is implemented by any system with the right input and output, then mentality is dependent only on input/output patterns. But functionalism is generally taken to stand in contrast to behaviorism, and to lack many of behaviorism's problems. If Putnam's result is correct, functionalism is no better off. As he summarizes:

> Thus we obtain that the assumption that something is a "realization" of a given automaton description (possesses a specified "functional organization") is equivalent to the statement that it behaves as if it had that description. In short, "functionalism", if it were correct, would imply behaviorism! If it is true that to possess mental states is simply to possess a certain "functional organization", then it is also true that to possess mental states is simply to possess certain behavior dispositions! (pp. 124-125)

Putnam's argument for this conclusion is a straightforward variation of his earlier argument. Given an arbitrary FSA, take a system S with the right input/output dependencies.

For the usual reasons S will go through a sequence of states s_1, s_2, and so on. Associate s_1 with the relevant initial state of the FSA, and associate s_2 and subsequent physical states with the relevant subsequent formal states (that is, those FSA states determined by the earlier states and inputs in question). To every FSA state, we map the relevant disjunction of associated physical states. The system therefore goes through relevant states a, b, c that correspond under this mapping to FSA states A, B, C; these states transit appropriately given the previous state and the input, and appropriate output is produced at each step. So, Putnam argues, the system implements the FSA.

To evaluate this claim, we need to spell out conditions of implementation for an FSA with inputs and outputs. An FSA with input and output is specified by a a set of states, a set of inputs, a set of outputs, and a set of state-transitions $(I, S) \rightarrow (S', O)$ for each input/state pair (I, S), specifying the next state S' and the output O that will be produced by that state and input. (Perhaps the output should depend only on the previous state and not on the input, but the generalization does not hurt.) We can then straightforwardly extend our previous account of implementation as follows. Of course, we require that the relevant strong conditionals are satisfied.

> A physical system implements an FSA (with input and output) in a given time-period if there is a mapping f from physical states of the system onto formal states of the FSA and from inputs and outputs to the system onto inputs and outputs of the FSA such that: for every formal state-transition $(I, S) \rightarrow (S', O)$ in the specification of the FSA, if the physical system is in a state s and receiving input i such that $f(s) = S$ and $f(i) = I$, this causes it to transit into a state s' such that $f(s') = S'$ and to produce output o such that $f(o) = O$.

(There will generally be some further restrictions on the mapping from inputs to inputs and outputs to outputs, but I will not go into this here.)

Does Putnam's construction satisfy this definition? It will have trouble with reliability and with uninstantiated formal states, of course, but as before we can stipulate that it contains a clock and a dial to avoid that problem. There is still a serious problem. We have established that the system transits appropriately in response to the inputs and outputs that it receives. However, we have no guarantee of its transition behavior if the inputs and outputs had been different. Again, in order to satisfy the strong conditionals, it is necessary that *if* it had been in a state/input pair that did not come up in the given run, then it would have transited appropriately.

Nothing in Putnam's construction ensures that this condition is satisfied. In fact there is every reason to believe that it will not be. If the sequence of internal states $s_1, s_2, \ldots$ is deterministic or reliable in itself, as it will be in the clock/dial case and as nothing in Putnam's stipulation rules out, then the system will produce precisely the *same* transition behavior for *any* inputs. There is none of the sensitivity that an FSA with input and output is required to have.

Once again, the strong conditionals go unsatisfied. Once again, this is because the structure of the FSA is not fully reflected in the structure of the physical system. All that is represented are a few state-transitions, those that come up on the run in question. If a state-transition in the machine table is not play a role in that run, then it will correspond to nothing in the structure of the physical system. The system therefore fails to implement the FSA.

Because of the combinatorial nature of these state-transitions, it is not as easy to get around this failure as it was before. We cannot simply disjoin states to our heart's content. For every state that takes part in such a disjunction, it is required that for *every* input, it transits appropriately. We cannot simply map the new physical state that follows a given state and input onto the requisite formal state, as it will often be the case that the same physical state can be produced as the result of two distinct state/input pairs. Because of this, the iterative mapping procedure described above will not be well-defined.

Still, this suggests yet another maneuver that might salvage something of Putnam's argument. If we require that for every initial physical state and every sequence of inputs, the system goes into a distinct physical state, the mapping above will be well-defined. Furthermore, this condition is not *too* hard to satisfy. It is satisfied by a system that keeps a record of all its inputs, for instance.

Let us say a physical system contains a *input memory* if it has a subsystem that goes into a distinct state for every possible sequence of inputs (think of it as keeping a list of the inputs so far, perhaps). It then turns out that every physical system with an input memory and a dial implements every FSA with the right input/output dependencies.

To see this, denote states of the physical system by labels $[j, i_1, ..., i_n]$ (where n can take any value from zero up). The dial state is represented by j, and the sequence of inputs in the recorder is represented by the sequence $i_1, \ldots, i_n$. To construct the mapping, assume the dial is in fact set to 1, and associate state [1] with the appropriate state S of the FSA. For every possible input sequence $i_1, \ldots, i_n$, we associate physical state $[1, i_1, \ldots, i_n]$ with the FSA state that we get by starting at state S and feeding inputs $i_1, \ldots, i_n$ (or the formal correlates of these) to the FSA. If there are FSA states that are not reached anywhere in this "tree", we pick such a state, associate it with physical state [2], and start the process again, and repeat until FSA states are exhausted. We map disjunctions of physical states to formal states in the obvious way, and the construction is complete.

It is easy to see that under this mapping, all the relevant strong conditionals are satisfied. If the system is in a physical correlate of a given formal state, and receives some input, it will always transit appropriately. As long as we require that the system has the right input/output *dependencies* (and not just the right input/output patterns in a particular run), then the system will also satisfy the right strong conditionals concerning outputs. It follows that the system implements the automaton.

By requiring the addition of an input memory, we have moved well beyond universal realization, but the result is still troubling. Input memories are not difficult to instantiate. A system that retains a separate mark for every input will achieve it, for instance. It follows for instance that any implementation of a simple zero-outputting FSA, supplemented with a dial and a input memory, will implement the complex primality-testing FSA. This result is quite counterintuitive, but it seems that we are stuck with it.

It also means that if AI is dependent on this FSA formalism, then functionalism still *almost* reduces to behaviorism. We know that everybody with the right behavior will realize a given automaton as long as they carry a videocamera to record inputs and a dial in their pocket. If so, the functionalist premise implies that any two behaviorally equivalent people will be mentally equivalent, as long as they are carrying input memories and dials. The caveat is not much help for non-behavioristic functionalism.

(There is an interesting alternative argument for the reduction of functionalism to behaviorism that Putnam does not mention. There is a plausible general principle about the computational basis of cognition:

> (++) If a cognitive system implements two FSAs F and G both of which are sufficient to produce the system's behavioral function, and G implements F, then the cognitive properties of the system arise in virtue of its implementing F (as the added structure of G is just irrelevant implementational detail).

However, by the minimization principle for finite-state automata (Hopcroft and Ullman 1979, p. 67), any two FSAs with the same input/output dependencies implement a common simpler FSA with the same dependencies. Take any two behaviorally equivalent cognitive systems whose cognitive properties arise by virtue of implementing an FSA. By the reduction theorem and (++), these will have the same cognitive properties. It follows that if functionalism is not to reduce to behaviorism, we must either reject (++) or reject FSAs as the basis for cognition.)

It is true that we are no longer in danger of panpsychism, as most systems will not even have the right behavior. However, the result may imply that every FSA will be implemented by a "giant lookup table" (see Block 1981) consisting of a tree with an output listed for each series of inputs. The matter is slightly unclear, as any such tree in our world will be finite; whether a finite tree implements the FSA depends on whether we allow the state-transition conditionals to fail for states at the end of a given time-period (that is, at the outer edge of the tree). But is a human being with a finite lifetime any better off? In any case, even the result that an infinite lookup-table would be as conscious as you and me is troubling.

6 Combinatorial State Automata

It might be tempting to respond to this construction by arguing that somehow these inputs, states, and outputs do not have the right sort of connections between them. After all, there is a reasonable sense in which the state of the recorder is causally irrelevant to the system's function; one might try to introduce some causal relevance condition, or some other requirement on the relation between states. I am not sure that this would work, however; there would be problems with FSAs that always output zero despite complex calculations, for example. I leave the possibility open, because otherwise we are left with the counterintuitive result that the zero-outputter with a dial and a recorder implements the prime-tester, but I will focus here on another way to handle the matter.

The real moral of the above discussion is that even simple FSAs with inputs and outputs are not constrained enough to capture the kind of complex structure that computation and cognition involve. The trouble is that the internal states of these FSAs are *monadic*, lacking any internal structure, whereas the internal states of most computational and cognitive systems have all sorts of complex structure. Generally these states are divisible into components which interact locally and globally according to complex principles. Just as the structure of the system is not captured by a monadic state description, neither are the state-transitions captured by a monadic state-change. There may be all sorts of local dependencies that go into the functioning of such a system.

Turing machines and cellular automata are good examples of computational devices whose states have internal structure. In cognition, it is obvious that the brain has a highly complex structure, with its functioning consisting in a complex pattern of interaction among billions of neurons and other parts. There is also plausibly complex structure in cognitive processing at coarser levels, structure that is central to the explanation of human thought and behavior.

To capture all this, I will introduce the notion of a *combinatorial state automaton*, or CSA. A CSA is much like an FSA, except that its states have combinatorial structure. An internal state of a CSA is a *vector* $[S^1, S^2, \ldots, S^n]$, where the ith component of the vector can take on a finite number of different values, or *substates*. The components of the vectors can be thought of by analogy with squares in a Turing machine or cells in a cellular automaton. The substates correspond to symbols in those squares or particular values for the cells. Inputs and outputs can also have a complex structure if this is desired. State-transition rules are determined by specifying for each component of the state vector a function by which its new substate depends on the old overall state vector and input vector (it will frequently depend on only a few "nearby" components of these vectors), and the same for each element of the output vector. The vectors involved may be finite or infinite, but I will stick to the finite case.

The conditions for implementation of a CSA are analogous to those for an FSA, with appropriate modifications. The main change is that internal states of the physical system must be specified as vectors, where each element of the vector corresponds to an independent element of the physical system. I will stipulate that each element in such a decomposition corresponds to a distinct physical region in the system, although there are other alternatives. The same goes for the complex structure of inputs and outputs, if required. The definition of implementation is as follows:

> A physical system implements a given CSA if there is a decomposition of its internal states into substates $[s^1, s^2, \ldots, s^n]$, and a mapping f from these substates onto corresponding substates S^j of the CSA, along with similar mappings for inputs and outputs, such that: for every formal state transition $([I^1, \ldots, I^k], [S^1, \ldots, S^n]) \rightarrow ([S'^1, \ldots, S'^n], [O^1, \ldots, O^l])$ of the CSA, if the system is in internal state $[s^1, \ldots, s^n]$ and receiving input $[i^1, \ldots, i^n]$ such that the physical states and inputs map to the formal states and inputs, this causes it to enter an internal state and produce an output that map appropriately to the required formal state and output.

Often the state-transitions of a CSA will be defined in terms of *local* dependencies, as when a substate depends only on a few neighboring substates and perhaps on a few inputs rather than on the entire previous state and input vectors (this will be so for cellular automata and Turing machines, for instance). In this case, we can require that the appropriate restricted conditional holds: that is, if the physical system is in the (few) specified previous substates and receiving the specified inputs, this causes it to transit appropriately.

It will be observed that (finite) CSAs are no more powerful that FSAs. For every CSA, there is an FSA that can simulate it. However, the implementation conditions for CSAs are much more constrained than those for the corresponding FSA. A implementation of a CSA is required to consist in a complex causal interaction among a number of separate parts. Not only do states have to be subdivisible, but the many state-transition conditionals imply that the system must have fine-grained causal structure than an implementation of the corresponding FSA might well lack.

Because of these constrained implementation conditions, CSAs are much less vulnerable to Putnam-style objections that FSAs. Unlike FSA implementations, CSA implementations are required to have complex internal structure and complex dependencies among their parts. For a complex CSA, the right sort of complex structure will be found in very few physical systems. What does the work is precisely the combinatorial nature of the states, through the requirement that the parts of an internal state are independently variable. This imposes a constrained structure of dependencies on the system that arbitrary systems have no hope of passing.

The CSA formalism also provides another solution to the problem of the blind and deaf paralytic mentioned earlier. A non-trivial computational basis for cognition in this case may be specified by a CSA description, where the CSA need not have inputs and outputs. Even if a CSA lacks inputs and outputs, its combinatorial structure enables it to evade Putnam's objections. We can therefore argue that it is in virtue of implementing this CSA that the person possesses mentality. Certainly such a person will still have complex combinatorial structure in their brain, and there will be all sorts of dependencies such as "*if* this neuron were in a different state, then...". The complexity and sensitivity implied by this formalism can therefore go a long way toward capturing the complex inner life of such a person.

CSAs are a plausible candidate for principles of computational sufficiency, then, due to the fine-grained causal structure they impose on an implementation. They are also much more appropriate for the purposes of cognitive *explanation* than FSAs. Where an FSA description of a process will consist in an unenlightening sequence of monadic states, a CSA description of the same process will provide detailed information about the internal structure and about patterns of interaction between various parts, giving the potential for a much better understanding of the process in question.

CSAs, more than many other computational formalisms, directly reflect certain properties of cognitive dynamics that such a formalism must handle, in order to provide a foundation for a theory of mind. Unlike monadic FSAs but like natural cognitive systems, their internal states have complex structure. Unlike Turing machines and cellular automata (at least as these are most commonly understood), these can take in input and produce output at every time-step, and area therefor quite suitable for the modeling of ongoing situated cognitive function. Nevertheless, the CSA framework is perfectly general. An FSA, a Turing machine, or a cellular automaton can be straightforwardly described as a CSA.

Given an account of implementation for CSAs, it is easy to derive an account of implementation for Turing machines, cellular automata, and so on. We simply have to translate one of these machines into the CSA formalism, which is straightforward. For an implementation of the machine in question, we simply require implementation of the corresponding CSAs. We can derive implementation conditions for programs in languages such as LISP and C by similar methods. Here the translation into the CSA formalism is more complex, but there is no problem in principle.

7 Open questions

The above account of what it is to implement a CSA is useful but not perfect. There are two problems that I can see: the first is that it lets in a few too many systems; the second is that it may let in too few. Neither of these problems are debilitating for the account, but both points suggest the need for a future, more perfect account. Here, I leave both of these as open questions for future work.

1. It turns out that even the CSA framework lets in some Putnam-style false implementations—of a sort. The false implementations here include no systems that we are ever likely to encounter, but they include certain astronomically large systems. Thus we still have counterexamples within the realm of logical possibility, if not within the realm of practical possibility. I give details in what follows.

In attempting to construct Putnam-style counterexamples, one might first try an extension of the "input memory" strategy, by building in an input memory into every component state. For a three-component FSA, for example, one might try an implementation that sends state $[a, b, c]$ with input I into state $[aI, bI, cI]$ (where a, b, c are strings, and aI is a concatenation of a and I); then, we map the strings aI, bI, cI onto the appropriate CSA substates, depending on the substates that a, b, and c mapped to. But it is easy to see that this "implementation" fails to have the requisite combinatorial structure. When we recombine component states here with other component states, we will get the wrong results. For example, *any* state of the form $[a, -, -]$ will transit into one of the form $[aI, -, -]$, but the CSA state-transitions will almost certainly require a different first component in the resulting state depending on the other components in the original state.

For a Putnam-style counterexample to be possible, every component state must be sensitive to every previous component state. The most straightforwad way to do this is as follows: build an implementation in which state $[a, b, c]$ with input I transits into state $[abcI, abcI, abcI]$ (where $abcI$ is a concatenation of a, b, c, and I). Now, we are assured that for every resultant component state, there is a unique candidate for the preceding state and input. Then we can construct the natural mapping from strings $abcI$ (in various positions) onto substates of the CSA, without fear of troubles with recombination. A recombined state such as $[a, b', c']$ will transit into a new state with unique component states in every position, each of which can be mapped to the appropriate CSA substate.

But this sensitivity comes at a price. A system like this will suffer from an enormous combinatorial explosion, getting three times bigger at every time-step. If the strings that make up each component have length L at one point, within 100 time-steps they will have length $3^{100}L$, which is about 5.10^{47} times larger. In a very short time, the system will be larger than the known universe! CSAs that are candidates to be bases for cognition will have many more than three components, so the situation there will only be worse. Here, the "implementing" system will reach the boundaries of the universe in number of steps corresponding to a fraction of a second in the life of a brain. So there is no chance that any realistic system could ever qualify as an implementation in this manner.

It is easy to see that any Putnam-style implementation will suffer from such an explosion. To achieve the requisite sensitivity "blindly", every component state must be sensitive to the value of every preceding component state. If a system had step t has n components each with k possible values, then at $t+1$ each component must be able to carry k^n possible values, at $t+2$ each component must be able to carry k^{n^2} possible values, and so on. Given that encoding k^m values takes a system whose size is proportional to m, we can see that the size of the system must increase by a factor of n at every time step.

No practically possible system has this structure, and it is likely that no physically possible system has it either, as this sort of unbounded explosion is probably incompatible with the physical structure of the universe. It follows that the threat of vacuity and of panpsychism have long since disappeared; the only physically possible systems that implement a non-trivial CSA will do so in virtue of having the right sort of fine-grained causal structure among their components. If the CSA is complex enough to be a plausible basis for mentality, the implementations will themselves be complex enough to be plausible candidates for mentality.

Still, the mere logical possibility of these false implementations is troubling. Intuitively, it seems that the exponentially-increasing system, for all its size, lacks the right sort of specific structure to count as really implementing the corresponding computation. Of course, one might bite the bullet and say that in a system that large, there is enough going on that we can truly find all computations implemented, just as one can find all possible books in Borges' Library of Babel. It might or might not follow that the corresponding minds could be found there too, depending on whether one takes the sufficiency of computational structure for mentality to hold with logical necessity or natural necessity. If it is only a natural necessity, then no conclusions follow from the consideration of a mere logical possibility, but if it is a logical necessity, the stronger conclusion would follow.

On reflection, I think that these systems should be ruled out as implementations, on grounds of lacking the right sort of causal structure. But this means that the account I have given of implementing a CSA is imperfect as it stands. It needs some sort of revision or addition, specifying an additional constraint on causal structure. To find this constraint, we need to reflect upon and analyse the features that the false implementations intuitively lack. One way to cash out the intuitions might be to add a "uniformity" clause, specifying that underlying each formal state-transition of a CSA there needs to be a uniform causal mechanism, as opposed to the highly disjunctive causal mechanism found in the false implementations. Another might be to add a causal relevance clause along the lines suggested before. But it is not obvious just how these suggestions should be spelled out in detail, so I leave the project as a challenge for the future.

In the meantime, we can rest reasonably content with the knowledge that the account as it stands provides satisfactory results within the class of physically possible systems. If a physically possible system implements a CSA on this account, it will do so by virtue of having the right sort of systematic, non-trivial causal dependencies between its component states. But the difficulties in the realm of logical possibility remain something of a challenge.

2. The second worry concerns an area in which the implementation conditions seem not too weak, but too strong. I refer to the constraint that each component in a CSA's state-vector must correspond to a distinct region in the physical system. (This constraint arises from the stipulated definition of a decomposition of a physical system.) One might think that the constraint is somewhat arbitrary. Why cannot components of a CSA be

implemented by overlapping physical states, for example?

Some such independence condition on components is required, however. Otherwise, the conditions for implementing a CSA would collapse into the conditions for implementing the corresponding FSA, and the structure of the CSA would be lost. To see this, consider an arbitrary CSA with an n-element state vector, and the corresponding FSA obtained by collapsing each complex state into a single monadic state, with appropriate state-transitions. Let T be a system that implements the FSA under a mapping f. To see that it implements the CSA under the unrestricted condition: let T^i_j be the substate of the CSA where the ith element takes on its jth value. Let t^i_j be the disjunction of states of T such that $f(t^i_j)$ is an FSA state corresponding to CSA state T^i_j (there will be many such FSA states, of course). Then $t^i_1, t^i_2, \ldots$ will be the states of the physical system corresponding to substates $T^i_1, T^i_2, \ldots$ of the CSA, and the implementation will hold under the appropriate mapping.

To see what is going on here, note that when the independence condition is left aside, a decomposition of internal states of a physical system into substates is just a set of non-maximal states t^i_j such that the states $t^i_1, t^i_2, \ldots$ (representing the various substates of the ith element) are mutually disjoint and together exhaustive of maximal states of the system, for each i. We can think of these as states wherein a parameter has a particular value, with a separate parameter for each element of the vector. If we allow each value of each parameter to be determined by an arbitrary global mapping, we lose the sense that the system is functioning by interaction between independent components, and we therefore lose the extra structure that is vital to the nature of a CSA.

In order to have a true implementation of a CSA, we must require that the various parameters are in some strong sense *independent*, corresponding to separable components of a system. The easiest way to do this is to require that for a system to implement a CSA, the physical differences relevant to the variation in a given parameter—must be restricted to a limited physical region, with different physical regions for each element. That is, the values of a "parameter" supervene on a distinct region for each parameter.

It is possible, however, that a weaker "independence" condition could suffice. After all, not all existing computational systems have disjoint regions for each computational component. Witness systems with virtual memory, where the value of a given CSA "component" can be represented in various parts of the system depending on where there is free space, simply by using a pointer to the correct memory location. Cases such as this one might be handled in terms of more flexible independence conditions on components. An alternative would be to handle such cases by a two-stage definition of implementation, noting that the system implements some more complex CSA under the usual conditions, and the complex CSA bears an appropriate relation to the original CSA. The burden here would rest on the latter relation, but that could be analyzed purely mathematically (in terms of pointer values and the like), avoiding the problems with physical systems and causation.

In any case, the conditions that we have outlined provide at the very least *sufficient* conditions for the implementation of an automaton, admitting a wide class of devices that are appropriate, and excluding counterintuitive "implementations" such as Putnam's. The question of how far these conditions can be loosened without letting in inappropriate systems remains open.

8 Minds, Brains, and Programs

It is worth mentioning an argument by Searle for a thesis similar to Putnam's (Searle 1990). He argues that any physical system can be seen to implement any computation: computational properties are not intrinsic to physics, so that computational descriptions are observer-relative. Under the right interpretation, for example, his wall might be seen to implement the Wordstar program.

From what we have seen here, this argument fails. Whether or not computational properties are intrinsic to physics, the implementation relation between abstract automata and physical systems is perfectly objective. Even if there is a correspondence between states of the wall and states of the Wordstar program, and even if we do not worry about inputs and outputs, it is almost certain that the wall states will not satisfy the relevant strong state-transition conditionals, as the wall will not possess the requisite causal organization. Implementation is a determinate matter, as we have seen, and it puts a very strong constraint on the relevant class of physical systems.

There is a weak sense in which computational descriptions are "observer-relative", however. This reflects the fact that most physical systems will implement more than one abstract automaton. Every system will implement a trivial 1-state CSA, for instance, most systems will implement a 2-state CSA, and so on. So an observer is free to choose between a number of different computational "descriptions" of a system. This poses no problems of vacuity, however. The question of whether or not a given system is accurately "described" by a given computational description remains an objective one. The vacuity problem would arise only if every system impleented *every* CSA, and that is not the case. A given complex CSA will only be implemented by a small fraction of physical systems.

Indeed, it is a commonplace idea that a given physical system can implement more than one abstract computation. A single computer can simultaneously implement a number of programs, for example. One can even have a situation where a program is indirectly implemented, by implementing one program that implements another in turn (where the implementation relation between programs is cashed out in the natural way). It is natural to expect that the brain implements a large number of different automata, depending on what part of the brain we are focusing on and on the level of detail we are concerned with. Descriptions of brain processes in terms of implementations of these automata remain both objective and informative, although which description we appeal to at a given time may depend on our purposes.

There may even be instances in which a single system implements two independent automata, both of which fall into the class sufficient for the embodiment of a mind. A sufficiently powerful future computer running two complex AI programs simultaneously might do this, for example. In this case, we would naturally expect two minds to arise, not one. This suggests that if minds arise in virtue of implementing CSAs, then mental properties should be ascribed to the (system, CSA) pair, rather than to the system alone.

Alternatively, it might be more in line with AI terminology to redefine the term "system" to denote the pair in question, in order that a single object can support more than one system. Either way, a given physical hunk of matter can be associated with more than one mind.

This suggests a natural response to Searle's (1980) "Chinese Room" argument. In the Chinese room, where a human is simulating a computer program, there are two distinct CSAs being implemented. The causal organization of the human's brain supports one CSA, and the causal organization between the marks on paper supports another one. We should not let the fact that the human is facilitating the causal dynamics of the second implementation blind ourselves to the fact that the causal structure is there—there are counterfactual-supporting state-transition relations between the marks on paper, just as there are between silicon chips or neurons—and that this structure is quite distinct from the structure in the first CSA implementation. We therefore have a situation analogous to that in which a computer implements two independent programs. There are two distinct causal and computational systems here, and we should expect two distinct minds to be associated with them.

9 Conclusion

Computational descriptions of physical systems need not be vacuous. We have seen that there is a well-motivated formalism, that of combinatorial state automata, and an associated account of implementation, such that the automata in question are implemented approximately when we would expect them to be: when the causal organization of a physical system mirrors the formal organization of an automaton. In this way, we establish a bridge between the formal automata of computation theory and the physical systems of everyday life. We also open the way to a computational foundation for the theory of mind.

This is only half the story, of course, and the easy half. The harder part is to take advantage of this bridge, showing that the physical properties that a computational description fixes are the properties in virtue of which minds arise. It is not implausible that minds arise in virtue of causal organization, but neither is it obvious. It is also plausible but not obvious that the discrete CSA framework can capture the precise causal organization (perhaps continuous, perhaps even non-computable) on which mentality depends. This second half is likely to be a long story, although see Chalmers (1994b) for my own attempt at putting things together. In any case, an account of computation and implementation clears the way.[4]

References

Block, N. 1981. Psychologism and behaviorism. *Philosophical Review* 90:5-43.

Chalmers, D.J. 1994a. On implementing a computation. *Minds and Machines* 4.

Chalmers, D.J. 1994b. A computational foundation for the study of cognition. Philosophy-Neuroscience-Psychology Technical Report 94-03, Washington University.

Hopcroft, J.E., and Ullman, J.D. 1979. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley.

Lewis, D. 1973. Causation. *Journal of Philosophy* 70:556-67.

Maudlin, T. 1989. Computation and consciousness. *Journal of Philosophy* 86:407-32.

Putnam, H. 1988. *Representation and Reality*. MIT Press.

Searle, J.R. 1980. Minds, brains and programs. *Behavioral and Brain Sciences* 3:417-57.

Searle, J.R. 1990. Is the brain a digital computer? *Proceedings and Addresses of the American Philosophical Association* 64:21-37.

Department of Philosophy

Washington University

St. Louis, Missouri 63130.

Notes

[1]Searle (1990) argues for a similar conclusion, although his argument is much less detailed than Putnam's. I comment on it briefly toward the end of the paper.

[2]I owe this observation to Joseph O'Rourke.

[3]Maudlin (1989) raises some very interesting questions about along these lines, questioning the relevance of counterfactual-involving conditionals to conscious experience. For instance, if one blocks the connection that supports some counterfactual conditional in a currently static part of the system, is it plausible that this could change or remove the system's conscious experience? Maudlin suggests that it could not. We have seen, however, that such conditionals are constitutive of computational organization. Using an elaborate construction, Maudlin makes an argument along these lines to the conclusion that cognition is not based in computation. This argument requires an in-depth treatment in its own right.

[4]The ideas in this paper arose out of discussion and argument with a number of others, largely over the Internet. Thanks are due to David Joslin, Daryl McCullough, Joseph O'Rourke, Calvin Ostrum, and Jerry Seligman.