ACADEMIC PRESS

Consciousness and Cognition xxx (2002) xxx–xxx

Consciousness and Cognition

www.academicpress.com

Counterfactuals cannot count: A rejoinder to David Chalmers☆

Mark Bishop*

Cybernetics Intelligence Research Group, Department of Cybernetics, University of Reading, Berkshire, UK

Received 3 July 2002

Abstract

The initial argument presented herein is not significantly original—it is a simple reflection upon a notion of computation originally developed by Putnam (Putnam, 1988). See also Searle (1990) and criticised by Chalmers et al. (see Chalmers (1994, 1996a,b) see also the special issue, *What is Computation*?, in Minds and machines, 4:4, November 1994). In what follows, instead of seeking to justify Putnam's conclusion that every open system implements every Finite State Automaton (FSA) and hence that psychological states of the brain cannot be functional states of a computer, I will establish the weaker result that, over a finite time window every open system implements the trace of FSA **Q**, as it executes program (**P**) on input (**I**). If correct the resulting bold philosophical claim is that phenomenal states—such as feelings and visual experiences—can never be understood or explained functionally.

☆ The argument developed herein followed discussions with David Chalmers regarding the authors papers, 'Dancing with Pixies,' presented at ASSC4 (Brussels 2000) and 'Counterfactuals Can't Count,' presented at ASSC5 (North Carolina 2001). An earlier treatment of this material, including background to the debate, will appear as, Bishop, J. M. (2002). Dancing with Pixies. In J. Preston & J. M. Bishop (Eds.), Views into the Chinese Room, Oxford: OUP.

* Fax: +44-118-931-8220.
E-mail address: j.m.bishop@reading.ac.uk.

1053-8100/02/$ - see front matter © 2002 Published by Elsevier Science (USA).
PII: S1053-8100(02)00023-5

1. Introduction

In the first part of this paper I introduce Discrete State Machines (DSMs) and show how, with input to them defined, their behaviour is described by a simple unbranching sequence of state transitions analogous to that of an inputless FSA. Then I review Putnam's 1988 argument that purports to show how every open physical system implements every inputless FSA. This argument is subsequently applied to a robotic system that is claimed to instantiate genuine phenomenal states as it operates—if the robot system instantiates such states then so must any open system.

This type of argument has previously been criticised on the grounds that Putnam's open system does not correctly implement counterfactuals.[1] However, in a novel extension I demonstrate that, in the execution of the robot's control program with known input, unless we allow non-executed state transition sequences to directly influence the generation of phenomenal states, the lack of counterfactuals in Putnam's open system cannot rescue functionalism; counterfactuals cannot count.

2. Discrete state machines

In his 1950 paper, Computing Machinery and Intelligence, Turing defined DSMs as, ''machines that move in sudden jumps or clicks from one quite definite state to another'' (Turing, 1950, p. 439), and explained that modern digital computers fall within the class of them. An example DSM from Turing is that of a wheel machine that clicks round through 120° once a second, but may be stopped by the application of a lever-brake mechanism. In addition, if the machine stops in a pre-specified location at one of the three possible positions, it will cause a lamp to come on. Input to the machine is thus the position of the lever-brake, {brake on; brake off}, and the output of the machine is the lamp state, {lamp on; lamp off}.

Such a machine can be described abstractly in the following manner. Its internal (computational) states are labelled (arbitrarily) by a mapping function f that maps from the physical state of the machine (i.e., what position the wheel is in) to the computational state of the machine, $q \in \{q_1\ q_2\ q_3\}$. The input to the DSM, the brake position, is described by an input signal, $i \in \{i_0$ brake off$; i_1$ brake on$\}$. Hence, the next internal state of the machine is determined solely by its current state and its current input as follows:

	$\boldsymbol{q_1}$	$\boldsymbol{q_2}$	$\boldsymbol{q_3}$
$\boldsymbol{i_0}$	q_2	q_3	q_1
$\boldsymbol{i_1}$	q_1	q_2	q_3

With its output being determined by:

[1] Strong FSA conditional statements of the form, '*if* the FSA is in state *a*, it will transit into state *b*, however it finds itself in the first state.'

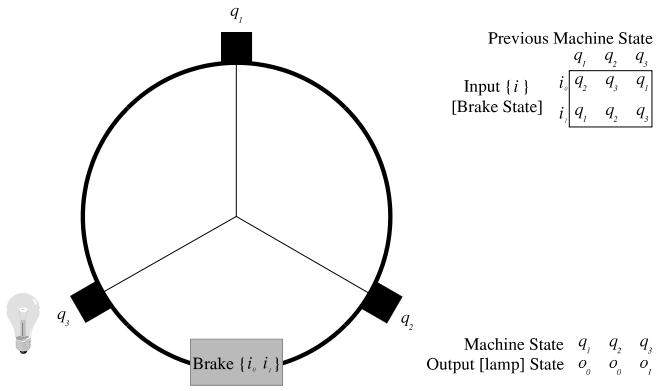

Fig. 1. Turing's, 'Discrete State Wheel Machine.'

State :	$\boldsymbol{q_1}$	$\boldsymbol{q_2}$	$\boldsymbol{q_3}$
Output :	o_0	o_0	$o_{1[\text{lamp on}]}$

Thus, with input undefined, the state transition diagram in Fig. 1 describes the DSM's behaviour.

Fig. 2 has several branch points where the next state of the machine is determined by a state transition contingent on the current input. However, as shown in Fig. 3, for any specific input value there are no such branching state transitions. The machine's output (lamp on/off) is determined purely by its initial state, ($q \in \{q_1\ q_2\ q_3\}$), and the system input value, ($i \in \{i_0\ i_1\}$).

Knowledge of the specific input to the machine's state transition table (program) has thus *collapsed its combinatorial structure*. Further, over a given time period, say $[t_1 \ldots t_7]$, all loops can be removed from the state diagram to form a linear path of state transits. The machine now functions, *like clockwork*, e.g.,

INPUT STATE 0: $\langle \boldsymbol{q_1\ q_2\ q_3\ q_1\ q_2\ q_3\ q_1} \rangle$ OR $\langle \boldsymbol{q_2\ q_3\ q_1\ q_2\ q_3\ q_1\ q_1} \rangle$
OR $\langle \boldsymbol{q_3\ q_1\ q_2\ q_3\ q_1\ q_2\ q_3} \rangle$

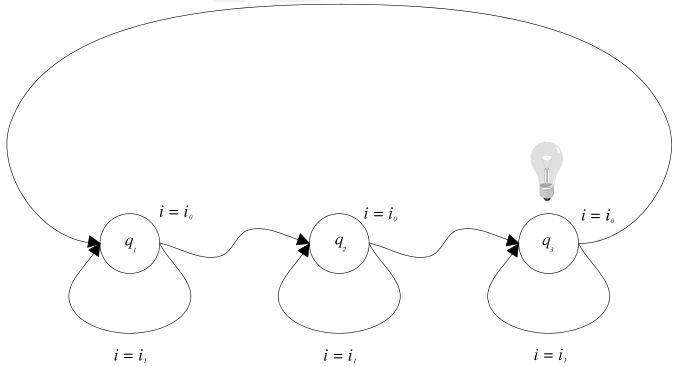

Fig. 2. State transition diagram of Turing's wheel machine—input undefined.

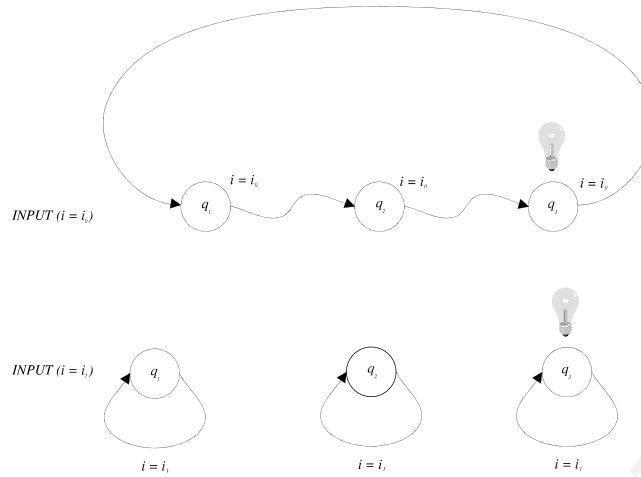

Fig. 3. State transition diagram of Turing's wheel machine—input defined.

OR

INPUT STATE 1: $\langle q_1\ q_1\ q_1\ q_1\ q_1\ q_1\ q_1\rangle$ OR $\langle q_2\ q_2\ q_2\ q_2\ q_2\ q_2\ q_2\rangle$ OR $\langle q_3\ q_3\ q_3\ q_3\ q_3\ q_3\ q_3\rangle$

The following argument endeavours to show that for any Discrete State Machine which, it is claimed, instantiates mental (phenomenal) states purely in virtue of its execution of a suitable state transition table, *or program*, we can generate a corresponding state transition sequence using any open physical system. That this conclusion leads to a form of panpsychism, where all parts of matter involve phenomenal consciousness, is clear, as if such state transition sequences are effectively found in all open systems, then phenomenal states must be equally ubiquitous in matter.

3. Putnam's argument

Tucked away in an appendix to Hilary Putnam's 1988 book, 'Representation and Reality', is a short argument that endeavours to prove that any open physical system is a realisation of every abstract inputless Finite State Automaton and hence that Functionalism fails to provide an adequate foundation for the study of the mind.

Central to Putnam's original argument is the observation that every open physical system, **S**, is in different maximal states[2] at every discrete instant and is characterised by a discrete series of non-cyclic[3] modal state transitions, $[s_1, s_2 \ldots s_t \ldots s_n]$. To simplify the following discussion of Putnam's claim, I will replace Putnam's arbitrary

[2] Putnam (1988, p. 122).
[3] Putnam (1988, p. 121).

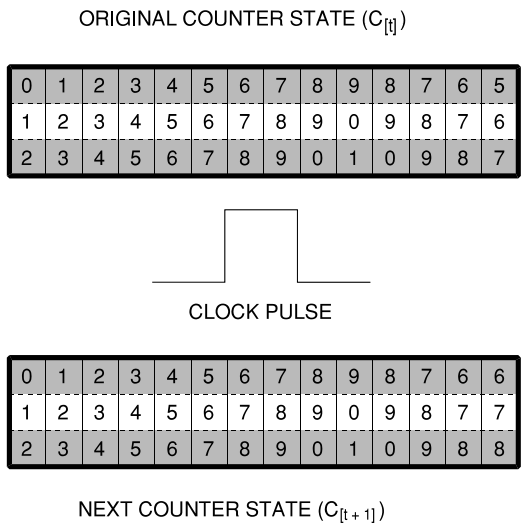

Fig. 4. A 'non-cyclic' counting machine.

open physical system, **S**, with an Infinite Counting Machine (ICM) generating the non-cyclic counter state sequence $[c_1, c_2 \ldots c_t \ldots c_n]$ in place of the physical system state sequence $[s_1, s_2 \ldots s_t \ldots s_n]$[4] (see Fig. 4).

It is clear that given counter state $[c_k]$ at time $[t_k]$, it is trivial to predict its next state $[c_{k+1}]$ at time $[t_{k+1}]$ NB. This transition from state $[c_k]$ to $[c_{k+1}]$ is both regular and carries full modal force—that the counting machine is in state $[c_k]$ defines and contains the provision to force it to transit to $[c_{k+1}]$ at the next clock interval.

Any inputless FSA is characterised by its state transition table, defining, given its current state, its subsequent state. Imagine, without loss of generality, that the state transition table for FSA **Q** calls for the automaton to go through the following sequence of states in the interval $[t_1 \ldots t_6]$:

$$\langle q_1\ q_2\ q_3\ q_1\ q_2\ q_3 \rangle$$

Next let us suppose we are given a counting machine, **C**, which goes through the sequence of states, $[c_1, c_2, c_3, c_4, c_5, c_6]$ in the interval $[t_1 \ldots t_6]$. We wish to find a mapping between counter states and FSA states such that, during the time interval under observation, the counting machine obeys **Q**'s state transition table by going

[4] Putnam uses an open physical system simply as a device to generate a non-cyclic series of discrete states. Chalmers (1996a) argues that for such a system to reliably transit a sequence of states it must include a natural clock (such as a source of radioactive decay), however he also concedes that, "probably most physical systems satisfy such a requirement" (Chalmers, 1996a, p. 316). Nonetheless, to simplify this exposition of Putnam's mapping I initially employ the conceptually simpler *ICM*, (an example of a finite counting machine being the milometer of a car). However, to fully generalise the results of the argument, in subsequent sections I will resort to using Putnam's *open physical system*.

through a sequence of states, which the state mapping function will label $[\boldsymbol{q_1}\ \boldsymbol{q_2}\ \boldsymbol{q_3}\ \boldsymbol{q_1}\ \boldsymbol{q_2}\ \boldsymbol{q_3}]$.

4. Putnam's mapping

It is trivial to observe that if we map FSA state $[\boldsymbol{q_1}]$ to the disjunction of counting machine states, $[c_1\ v\ c_4]$, FSA state $[\boldsymbol{q_2}]$ to the disjunction of counting machine states, $[c_2\ v\ c_5]$ and FSA state $[\boldsymbol{q_3}]$ to the disjunction of counting machine states, $[c_3\ v\ c_6]$, then the counting machine will fully implement **Q**. Further, given any counting machine state $[c_a] \in \{c_1, c_4\}$, at time $[t_a]$, we can predict the FSA will enter state $[\boldsymbol{q_2}]$ at time $[t_b]$.

To show that being in state $[\boldsymbol{q_1}]$ at time $[t_1]$ caused the counting machine to enter state $[\boldsymbol{q_2}]$ at $[t_2]$ we observe that at $[t_1]$ the counting machine is in state $[c_1]$, (which the mapping function labels FSA state $[\boldsymbol{q_1}]$), and that being in state $[c_1]$ at $[t_1]$ causes the counting machine to enter state $[c_2]$, (which the mapping function labels FSA state $[\boldsymbol{q_2}]$) at $[t_2]$. Hence, given the current state of the counting machine at time $[t]$, we can predict its future state and hence how the states of **Q** evolve over the time interval under observation.

Note, after Chalmers, that the counting system above will only implement a particular execution run of the FSA. Hence, Chalmers remains unfazed at this result because he states that inputless FSA's are simply an inappropriate formalism for a Computationalist Theory of Mind.

> To see the triviality, note that the state space of an inputless FSA will consist of 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 a completely uninteresting kind of structure, as indeed is witnessed by the fact that it is satisfied by a simple combination of a clock and dial. (Chalmers, 1996a, p. 318)

But Putnam extends his result to the case of FSA's with input and output, by arguing that an FSA with input and output is realised by every open physical system with the right input/output dependencies—if the physical system has the right input/output then it instantiates the FSA correctly. Patently, this is a restriction on his original claim, but nonetheless it remains a significant result that, if correct, suggests functionalism implies behaviourism.[5]

Putnam's original argument, using an open physical system to generate a series of non-repeating system states, (equivalent to the non-cyclic counting machine states used above), runs as follows. For any arbitrary FSA, take an open physical system with the right input/output dependencies. For example a rock with a number of marks on it encoding the input vector, (**I**), (where (**I**) encodes the finite set of input

[5] "Any open system with the correct input/output dependencies implements the FSA with input/output. Hence, every FSA with input (I) and output (O) is implemented by any physical system with the same input/output dependencies. Hence mentality is contingent only on input and output and functionalism implies behaviourism." (Putnam, 1988, pp. 124–125).

values, $\{i_1, i_2 \ldots i_n\}$) and another set of marks encoding the output vector (**O**), (where (**O**) encodes the finite set of output values, $\{o_1, o_2 \ldots o_n\}$). Associate rock state $[s_1]$ at $[t_1]$ with the relevant initial state of the FSA, and rock state $[s_2]$ at $[t_2]$ with the subsequent state of the FSA etc. Putnam claims it is clear that by mapping each FSA state with the disjunction of associated rock states we ensure the system goes through the relevant state sequence $[s_1, s_2, s_3]$ that corresponds, using this mapping, to the relevant FSA state sequence, $[q_a, q_b, q_c]$, with system output encoded by the other marks on the rock.

However, as for Turing's DSM, the addition of input now makes the formalism non-trivial. There can now be branching in the execution trace, as the next FSA state is contingent upon its current state and the system input. This gives the system a combinatorial structure. However, as Chalmers states, Putnam's revised construction does not fully encapsulate this structure—rather it merely manifests one trace of the FSA with a specific input/output dependency. However, as we observed with Turing's simple three state DSM, when input to the system is defined over a specific time interval the combinatorial state structure of the FSA collapses to a bounded linear path, which can be generated using a Putnam style mapping and any open physical system.

5. Simulating phenomenal states over a finite time window

Consider a mobile robot whose behaviour is controlled by a program (**P**), acting upon input (**I**), generating output (**O**), running on computer hardware (**Q**), over the specified time interval $[t_1 \ldots t_k]$. Assume that after switch-on at time $[t_1]$, the robot experiences a series of phenomenal states until it is switched off at time $[t_k]$.

During the specified interval, $[t_1 \ldots t_k]$, the robot's input states are defined by data from its sensors forming the input set $(\mathbf{I}) = \{i_1\ i_2\ i_3 \ldots i_k\}$ with its output states defining the actuator commands controlling its external behaviour, which together form the output set $(\mathbf{O}) = \{o_1\ o_2\ o_3 \ldots o_k\}$. Let us now concede that the robot's control program, (**P**), instantiates a series of phenomenal states in **Q**, caused by interaction with its environment as the program executes. What is it in the robot that manifests this property? Unless we allow these phenomenal states to extend outside of the robots computational hardware, **Q**, they must be manifested solely by it. That is, the claimed phenomenal properties of the system must be solely realised by **Q** (**P**,**I**); the computer **Q**, executing program (**P**), on input (**I**).

Now as **Q** executes (**P**) over the interval $[t_1 \ldots t_k]$, **Q** (**P**, **I**) generates a specific set of computational states, **S**, ($\mathbf{S} = \{s_1, s_2 \ldots s_t\}$), at the discrete clock intervals of the CPU, **Q**. Due to the universal realisability of Turing Machine programs,[6] the particular underlying computational engine, **Q**, is irrelevant to the generation of the computational states $\{s_1, s_2 \ldots s_t\}$. Whether **Q** had at its heart a CPU made from toilet rolls[7] or the latest Intel processor, the associated computational states

[6] All digital computers are in a sense equivalent—the underlying hardware that executes a specific computation does not affect the result of that computation.

[7] Weizenbaum (1976, p. 51).

are the same and it is **Q**'s generation of these states that results in its instantiation of phenomenal experience.

However, we have already seen from Putnam how the computational states resulting from the execution of any given FSA, **Q**, (with specific input/output) can be mapped onto an open physical system, (e.g., a simple counting machine). Thus, by relaxing the requirement that the physical system instantiates the full combinatorial state structure of a program with general input, to the relatively trivial requirement that it instantiate the correct state transitions for a specific execution trace, we sidestep the need for an exponential increase in state space. Thus, with the input set defined, over the specified time interval $[t_1 \ldots t_k]$ we are able to fully generate the computational states governing the robot's behaviour and hence instantiate its phenomenal states, via a simple counting machine or, following Putnam, any open physical system.

6. Counterfactuals can't count

The above argument has been criticised by David Chalmers and others[8] on the grounds that the Putnam FSA analogue, (the open physical system/non-cyclic counter described earlier), does not genuinely replicate an FSA with input. In particular, it lacks the ability to correctly implement counterfactuals. Specifically, 'if a state-transition in the machine table does 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.' It would seem that even though specific computational states may not be entered on a specific execution run of the FSA, the mere possibility that they could be if system input was different is required in a genuine functional isomorph of the FSA.

However, in the next section, by employing a variation on Chalmers' Fading Qualia Argument (FQA), I will show that in the context of an FSA with input fixed a priori, state transition behaviour reduces to that of an inputless FSA, and Putnam's FSA open system analogue fully holds.

7. A clockwork robot

Consider the operation of two robots over the time interval $[t_1 \ldots t_k]$, with defined input (**I**), ($\mathbf{I} = [i_1 \ldots i_k]$). A program designed to Chalmers' specification, replicating the fine-grained functional organisation of a system known to have phenomenal states, controls one robot, R_1; the second, R_2, generates the computational states of R_1 via an open physical system using Putnam's mapping.

Although the input to both robots is fully defined a priori, ($\mathbf{I} = [i_1 \ldots i_k]$), and hence the series of computational states transited by both systems over the specified

[8] In discussions with: Peter Fletcher at the University of Keele, (UK, 1999); David Chalmers at ASSC5 (North Carolina 2001).

time interval is identical, the *counterfactual hypothesis* is that only R_1 would experience phenomenal states, as only R_1 correctly implements counterfactuals and maintains behaviour that is sensitive to the input as its program execution trace,[9] T_{R1} (**I**), evolves.

However, consider what happens if, contingent upon the input (**I**), at each conditional branch point in the execution trace T_{R1}(**I**), we delete a state transition sequence that is not entered,[10] then iteratively repeat this procedure until Chalmers' robot, R_1, with full input-sensitivity, is step-by-step transformed into a copy of the second robot R_2, such that its resulting behaviour is also determined solely by a series of linear state transitions. We can imagine that throughout this replacement procedure the robot repeatedly reports the colour of a red square fixed within its sensor field; that is, the robot input vector (**I**) remains constant and encodes the red square, (**I** = [red square]).

Between R_1 and R_2 there are a number of intermediate robots $[R_1 \ldots R_{(i-1)}R_{(i)}R_{(i+1)} \ldots R_2]$—what is it like to be them? As we transform R_1 into R_2, how does its phenomenal perception vary? Imagine that initially R_1 was having bright experience of the red square, the counterfactual hypothesis would be that at some point in the transformation $R_{(i)}$'s phenomenal experience must cease to be bright red. However, the only difference between $R_{(i-1)}$ and $R_{(i)}$, given the bright red square remains fixed as input, is that a sequence of states that was not transited in $T_{R(i-1)}$(**I**), has been deleted from $R_{(i)}$.

It is clear that this scenario is implausible, for otherwise we have a robot, $R_{(i)}$, whose phenomenal experience of input (**I**) is in someway contingent upon the mere presence of potential state transition sequences that are never transited during FSA operation.[11] Hence the counterfactual hypothesis is false and the phenomenal experience of R_1 and R_2 must be the same.

Yet is this version of the FQA valid? David Chalmers has argued that it is not. In contrast to the original FQA, in this scenario although the first robot, R_1, is sensitive to its input, the second, R_2, is clearly not ('it merely acts like a clockwork toy').

It is clear that removing sensitivity to input per se cannot affect $R_{(i)}$'s phenomenal experience, for consider what would happen to R_1 if the link between its frame store[12] and its visual sensor is damaged, such that its frame store constantly maintains digital representation of a red square, irrespective of the true image that falls on the optical

[9] Program execution trace, T_{R1}(**I**), is the sequence of state transitions generated by the FSA controlling robot R_1, given input (**I**).

[10] This is achieved by replacing one input-sensitive branching state transition, (cf. Fig. 2), with a simple linear state transition contingent on the input, (cf. Fig. 3).

[11] In other words, the mere presence of segments of program code that are not executed by the robot's computer would have direct casual influence over the phenomenal states instantiated in the machine.

[12] A device that maintains a digital representation of an image obtained by a visual sensor such as a TV camera.

sensor. This would result in R_1, like R_2, becoming insensitive to its visual input.[13] When asked, R_1 will now 'act like clockwork' and always report that the object in its visual field is a red square, irrespective of its true shape and hue.

The lack of input-sensitivity will either deflate the phenomenal experience of R_1 or have no effect (with any phenomenal states in the latter case analogous to the human experience of a red hallucination). But as R_1's control program is unchanged and the data it reads from its frame store is exactly the same as usual, (a set of binary numbers), R_1 will continue to function as normal and its phenomenal experience must be unchanged. Hence, in the execution of an FSA with fixed constant input, the lack of input-sensitivity per se will not influence the phenomenal states instantiated by the system.

8. Conclusion

For any computing machine, **Q**, executing program (**P**), with known input (**I**), over the specified time interval $[t_1 \ldots t_k]$, only a formal [and repeatable] series of state transitions occur within its hardware. The generation of the state transitions must be responsible for the generation of the machines phenomenal properties. In this paper we have seen why, following Turing's observations on universal realisability, the underlying hardware that instantiates computational state transitions is unimportant and hence, after Putnam, that in principle a series of such transitions could be implemented by any open physical system. Finally we observed that, with input to **Q** (**P**, **I**) defined, the mere potential for counterfactual reasoning,[14] cannot influence the phenomenal states instantiated by the system. Thus, over a finite interval, if a computational system **Q** (**P**, **I**) does instantiate genuine phenomenal states, then so will any open physical system; implying functionalism will fail, crediting too many things with the phenomenal quality of mind.

Acknowledgments

The author gratefully thanks Dr. Slawomir Nasuto and the anonymous reviewers for their helpful comments on the original manuscript.

References

Chalmers, D. (1994). On implementing a computation. *Minds and Machines, 4*, 391–402.
Chalmers, D. (1996a). Does a rock implement every finite-state automaton. *Synthese, 108*, 309–333.

[13] In the earlier discussion R_1 is incrementally modified but its input data is constant. Here the FSA functions as designed, but its incoming data is modified, (it is forced to take on a specified value irrespective of the image on the robots optical sensor).

[14] Strong FSA conditional state transitions.

279 Chalmers, D. (1996b). *The conscious mind: in search of a fundamental theory*. Oxford: Oxford
280 University.
281 Putnam, H. (1988). *Representation and reality*. Bradford Books.
282 Searle, J. (1990). Is the brain a digital computer. *Proceedings of the American Philosophical*
283 *Association, 64*, 21–37.
284 Turing, A. (1950). Computing machinery and intelligence. *MIND, 49*, 433–460.
285 Weizenbaum, J. (1976). *Computer power and human reason: from judgement to calculation*. San
286 Francisco: W.H. Freeman.