B. JACK COPELAND

WHAT IS COMPUTATION?

ABSTRACT. To compute is to execute an algorithm. More precisely, to say that a device or organ computes is to say that there exists a modelling relationship of a certain kind between it and a formal specification of an algorithm and supporting architecture. The key issue is to delimit the phrase 'of a certain kind'. I call this the problem of distinguishing between standard and nonstandard models of computation. The successful drawing of this distinction guards Turing's 1936 analysis of computation against a difficulty that has persistently been raised against it, and undercuts various objections that have been made to the computational theory of mind.

1.

In 1936 Turing published his now famous analysis of the concept of computation.[1] It is true to say that this analysis has become standard in mathematical logic and the sciences. However, there is in the philosophical literature a certain class of problem cases which forms the basis of an objection to Turing's analysis. The thrust of the objection is that although Turing's account may be necessary it is not sufficient. If it is taken to be sufficient then too many entities turn out to be computers. The objection carries an embarrassing implication for computational theories of mind: such theories are devoid of empirical content. If virtually anything meets the requirements for being a computational system then wherein lies the explanatory force of the claim that the brain is such a system?

I aim to meet the objection. According to the point of view to be put forward here, to compute is to execute an algorithm. More precisely, to say that a device or organ computes is to say that there exists a modelling relationship of a certain kind between it and a formal specification of an algorithm and supporting architecture. The key issue is to delimit the phrase 'of a certain sort'. For want of a better terminology I describe the problem to be addressed as that of distinguishing between standard and nonstandard models of computation. Prominent attempts to deal with the problem involve modification of Turing's analysis (for example Goel 1991 and 1992, Searle 1990 and 1992, Smith forthcoming, Sterelny 1990). In my view this is both undesirable and unnecessary. My intention here is to uphold the sufficiency of Turing's analysis.[2]

Synthese **108:** 335–359, 1996.

A recent presentation of the objection is to be found in Searle (1992, see also Searle 1990, 25–27).

> [T]he original definitions given by Alan Turing ... [form] the standard definition of computation ... [pp. 205–6]. On the standard ... definition of computation it is hard to see how to avoid the following results: 1. For any object there is some description of that object such that under that description the object is a digital computer. 2. For any program and for any sufficiently complex object, there is some description of the object under which it is implementing the program. Thus for example the wall behind my back is right now implementing the Wordstar program, because there is some pattern of molecule movements that is isomorphic with the formal structure of Wordstar. But if the wall is implementing Wordstar, then if it is a big enough wall it is implementing any program, including any program implemented in the brain ... [pp. 208–209]. I think it is possible to block the result of universal realisability by tightening up our definition of computation ... [A] more realistic definition of computation will emphasise such features as the causal relations among program states, programmability and controllability of the mechanism, and situatedness in the real world. [p. 209]

A much discussed problem case, in its essentials identical to Searle's, is due to Ian Hinckfuss. (Hinckfuss presented the problem case – which is now popularly known as 'Hinck's pail' – in discussion at the Australasian Association of Philosophy Conference, Canberra, 1978.) Here is Bill Lycan's engaging description of it.

> Suppose a transparent plastic pail of spring water is sitting in the sun. At the micro level, a vast seething complexity of things are going on: convection currents, frantic breeding of bacteria and other minuscule life forms, and so on. These things in turn require even more frantic activity at the molecular level to sustain them. Now is all this activity not complex enough that, simply by chance, it might realize a human program for a brief period (given suitable correlations between certain micro-events and the requisite input-, output-, and state-symbols of the program)? And if so, must the functionalist not conclude that the water in the pail briefly constitutes the body of a conscious being, and has thoughts and feelings and so on? Indeed, virtually any physical object under any conditions has enough activity going on within it at the molecular level that, if Hinckfuss is right about the pail of water, the functionalist quickly slips into a panpsychism ... (Lycan 1981, 39. See also Lycan 1987.)

I shall argue that Searle is right in his claim that there is some pattern of physical activity in the wall (or bucket, etc) that is 'isomorphic with the formal structure' of any given instance of computation (for example a ten minute session with Wordstar). From this claim a proposition easily mistaken for one entailing Searle's theses 1. and 2. does validly follow.

2.

This section offers an intuitive account of when it is true to say that an entity *e* – real or conceptual, artefact or natural – is computing a given function.

Let f be the function and let α be an architecture-specific algorithm that takes arguments of f as inputs and delivers values of f as outputs.

An algorithm is a 'mechanical' or 'moronic' procedure for achieving a specified result (in the case of α, of course, the specified result is arriving at the values of f). That is to say, an algorithm is a finite list of machine-executable instructions such that anyone or anything that correctly follows the instructions in the specified order is certain to achieve the result in question. To say that α is *specific to* an architecture is to say not only that a machine with this architecture can run α but also that each instruction in α calls explicitly for the performance of some sequence of the primitive (or 'atomic') operations made available in the architecture. (The sequence may consist of a single operation.) Thus an algorithm at least one of whose instructions calls explicitly for a multiplication cannot be specific to an architecture that has addition but not multiplication available as a primitive. A program calling for multiplications can run on such an architecture only because the compiler (or equivalent) replaces each multiplication instruction in the program by a series of addition instructions.

The primitive operations that are available vary from architecture to architecture. To give some examples: – If the architecture in question is a Turing machine then α will be a Turing machine table. The primitive operations include: move the tape left one square; read the symbol beneath the head; replace the symbol beneath the head by 0. If the architecture in question is that defined by a particular assembly language (an assembly language is an architecture-specific programming language) then α will be a program in that language. The primitive operations that are available may include: adding the binary numbers in a specified pair of registers and storing the result in a specified register; shifting the bits in a specified register one place to the right (the bit at the far right 'falls out' and 0 is fed in at the left); taking the logical conjunction of whatever binary strings occur in a specified pair of registers and storing the result in a specified register (the logical conjunction of, for example, 1100 and 1010 is 1000). Where the architecture in question is a neural net with a particular pattern of connectivity and certain weights on the connections, α consists of step-by-step applications of a certain propagation rule and a certain activation rule. (A propagation rule calculates the total weighted input into an artificial 'neuron' and an activation rule calculates what the activity level of the 'neuron' is to be, given its total weighted input.) In such an architecture steps are, of course, carried out in parallel: the execution of an algorithm is not necessarily a sequential procedure.[3]

Suppose one has a formal specification of the architecture in question and of the algorithm α, call it SPEC. For definiteness, let SPEC take the form of a set of axioms, although nothing in what follows turns on the use of the axiomatic method as opposed to some other style of formalisation. (In the next section I give a simplified example of such an axiom set.) So on the one hand we have SPEC, a description of a machine, and on the other we have the entity e. How do we bridge the gap and say that e *is* such a machine (at the time in question)? The bridge is effected by means of a system of *labelling* for e. (The concept of a labelling of a device is also used by Gandy 1980 and Deutsch 1985.) For example, labels may be eight binary digits long and be associated with groupings of subdevices in the following way: the possible states – measured in volts, say – of these subdevices are divided into two mutually exclusive classes and a state is labelled '0' if it falls into the first, '1' if it falls into the second. (No privilege is accorded to the binary system: labels may equally well be n- tuples of real numbers.) Thus a labelling scheme for an entity consists of two parts: (1) the designation of certain parts of the entity as label-bearers, and (2) the method for specifying the label borne by each label-bearing part at any given time.

The idea, of course, is that the labels constitute a 'code' such that spatial or temporal sequences of labels have semantical interpretations. In explaining the behaviour and function of the labelled entity one ascribes the semantical interpretation associated with the labels directly to the labelled states themselves. To take a simple example, if the voltage across one member of a pair of flipflops is 600 mV and across the second is 100 mV the pair may be labelled $\langle$High, Low$\rangle$ or $\langle 1, 0\rangle$ and be described as 'representing' or 'storing' the number two.

When the formal axioms in SPEC are true of an entity e under a labelling scheme L the ordered pair $\langle e, L\rangle$ will be called a *model* of SPEC. Here, then, is the promised intuitive account:

Entity e is computing function f if and only if there exist a labelling scheme L and a formal specification SPEC (of an architecture and an algorithm specific to the architecture that takes arguments of f as inputs and delivers values of f as outputs) such that $\langle e, L\rangle$ is a model of SPEC.

(A sentence of the form 'There exists a function f such that ... ' may be true irrespective of whether it is known to be true. I use the phrase 'there exist a labelling scheme and a formal specification such that' in the same way.)

To gain a feel for what is being said here consider the hoary problem of whether the solar system computes solutions to its own equations of motion (Fodor 1975, 74). (A grander version asks whether the universe is a computer dedicated to computing its own behaviour.) Certainly in some loose sense the planets 'follow' the relevant law of motion (Kepler's law, say). However, this hardly suffices to show that the planets execute an algorithm. For one thing, Kepler's law simply is not a list of instructions each of which calls for the performance of one or another of the primitive operations of some given architecture. Kepler's law states a functional relationship f between certain magnitudes but it is obviously not itself an algorithm that takes arguments of f as inputs and delivers values of f as outputs. Such algorithms do exist, of course. At least one of them requires a supporting architecture that makes available the operation of shifting the bits in a register one place to the right. Let SPEC be a specification of this algorithm and its supporting architecture. Is the solar system an example of the type of computing machine described by SPEC? To answer 'yes' is to suppose that the solar system is a register machine – is to suppose that the solar system consists of an interconnected structure of binary registers that are responsive to the shift operation and the various other operations demanded by the algorithm, for example binary addition and logical conjunction. Such a supposition no doubt strikes you as ludicrous. Perhaps those who seriously entertain the thought that the solar system computes will respond that it was never an algorithm like this one that they had in mind. Well and good. The foregoing account of computation presents them with a challenge: if they want to persist in the claim that the solar system is computing the function f then they must describe for us the solar system's computational architecture and detail the algorithm by which the solar system arrives at values of f.

Despite its naturalness the above account of computation will not do. This follows immediately from a result that I will call *Searle's Theorem*:

> For any entity x (with a sufficiently large number of discriminable parts) and for any architecture-algorithm specification y there exists a labelling scheme L such that $\langle x, L\rangle$ is a model of y.

In Section 4 I will prove Searle's Theorem. Searle's challenge to those who think the concept of computation significant – and it is a good challenge – is to modify the foregoing account in such a way as to avoid this trivialisation. I show in Section 5 that this can be done without departing from Turing's characterisation of computation.

Interestingly enough Turing's friend and colleague Max Newman used what is essentially a notational variant of Searle's Theorem as the basis of a devastating objection to Russell's causal theory of perception (Newman 1928).[4] In *The Analysis of Matter* Russell argued that although our perceptions tell us nothing about the 'intrinsic character' of the external stimuli that occasion them, we may nevertheless infer a great deal about 'the *structure* of stimuli' (Russell 1927, 227).

> [I]t would seem that, wherever we infer from perceptions, it is only structure that we can validly infer; and structure is what can be expressed by mathematical logic ... (Russell 1927, 254)

Newman's objection is as follows.

> A point to be emphasised is that ... no ... information about the aggregate A, except its cardinal number, is contained in the statement that there exists a system of relations, with A as field, whose structure is an assigned one. For given any aggregate A, a system of relations between its members can be found having any assigned structure compatible with the cardinal number of A. (Newman 1928, 140)
>
> These statements [of Russell's] can only mean, I think, that our knowledge of the external world takes this form: The world consists of objects, forming an aggregate whose structure with regard to a certain relation R is known, say W; but of the relation R nothing is known ... but its existence; that is, all we can say is, '*There is* a relation R such that the structure of the external world with reference to R is W'. Now ... such a statement expresses only a trivial property of the world. Any collection of things can be organised so as to have the structure W, provided there are the right number of them. (Newman 1928, 144)

We know of Russell's response to Newman's objection from a letter included in the second volume of Russell's autobiography:

> Dear Newman
>
> Many thanks for sending me the off-print ... I read it with great interest and some dismay. You make it entirely obvious that my statements to the effect that nothing is known about the physical world except its structure are either false or trivial, and I am somewhat ashamed at not having noticed the point for myself. It is of course obvious, as you point out, that the only effective assertion about the physical world involved in saying that it is susceptible to such and such a structure is an assertion about its cardinal number. (Russell 1968, 176)

3.

This section explains by means of an example what is meant by an axiomatic specification of a machine architecture. I will consider a simple machine M whose central processor consists of three eight-bit registers: an instruction register I, a data buffer D and an accumulator A. To give an example of the sort of thing that M does, if the instruction to perform an addition enters I, the machine adds the contents of D to the contents of A and stores the result in A. For simplicity I will ignore all matters concerning input, output, data transfer between the cpu and memory, and program storage and control, and I will omit the specification of any particular algorithm or program. M is a von Neumann machine (i.e. a particular type of serial processor). However, I emphasise that M appears here merely as an illustration. As I have already said, the account of computation under investigation applies both to serial and parallel architectures, including connectionist networks. (It is interesting, incidentally, that Turing himself seems to have been the first to consider building computing machines out of simple, neuron-like elements connected together into networks in a largely random manner; see Turing 1948, Copeland and Proudfoot 1996.)

The behaviour of M is described by means of a primitive term 'ACTION-IS'. ('ACTION-IS' is a member of the same family as Belnap's stit construction and Segerberg's δ operator.[5]) The axioms for M (or rather for M simplified in the ways just mentioned) are as follows. The function $\bar{x}$ is to be read 'the contents of (register) x' and $\Rightarrow$ is to be read 'becomes'.

$Ax1$ If $\bar{I} = 00000001$ ACTION-IS $(\bar{A} \Rightarrow \bar{D})$

$Ax2$ If $\bar{I} = 00000010$ ACTION-IS $(\bar{A} \Rightarrow \bar{A} + \bar{D})$

$Ax3$ If $\bar{I} = 00000011$ ACTION-IS $(\bar{A} \Rightarrow \bar{A} \times \bar{D})$

$Ax4$ If $\bar{I} = 00000100$ ACTION-IS $(\bar{A} \Rightarrow \bar{A} + \bar{\bar{D}})$

And so on for the remaining instructions in the machine's order code. Thus $Ax1$ says that the displayed binary digits are the instruction to wipe the accumulator and transfer the contents of D to the accumulator; and $Ax4$ says that the displayed digits are the instruction to add the contents of the register whose address is stored in D to the contents of the accumulator and store the result in the accumulator.

The intended interpretation of a statement of the form 'if X ACTION-IS Y' is that the occurrence of X produces or brings about the action Y. Such a statement is of the same logical 'strength' as the statement that X causes Y, in that each supports the counterfactual 'if X had occurred then Y would have occurred'. However, the truth-condition of statements of the form 'if X ACTION-IS Y' *cannot be couched in terms of physical causation*, for the construction must be applicable not only to real hardware but also to merely conceptual machines. For example, we wish to say that each action of a Turing machine is the result of its configuration (i.e. the combination of its state and the scanned symbol). Unless we intend to speak metaphorically, the phrase 'is the result of' cannot be replaced here by 'is caused by', for the machine is a purely abstract entity.

It is worth emphasising that 'if X ACTION-IS Y' expresses a notion of consequence or dependency that is stronger than that expressed by the material implication $X \supset Y$. Suppose that in the course of M's operations the instruction 00000100 happens never to enter I; the axioms nevertheless tell us which action would have ensued if it had. Yet since any material implication with a false antecedent is true, the following implications are all true in the envisaged circumstances:

$$\bar{I} = 00000100 \supset \bar{A} \Rightarrow \bar{A} + \bar{\bar{D}}$$

$$\bar{I} = 00000100 \supset \bar{A} \Rightarrow \bar{D}$$

$$\bar{I} = 00000100 \supset \bar{A} \Rightarrow \bar{A} + \bar{D}$$

$$\bar{I} = 00000100 \supset \bar{A} \Rightarrow \bar{A} \times \bar{D}$$

Clearly the set of material implications of this form that are true of M gives no guidance as to how M is designed to behave. Furthermore, since any material implication with a true consequent is true it is equally the case that if at some point in the computation the action $\bar{A} \Rightarrow \bar{A} + \bar{\bar{D}}$ is performed then the following are all true:

$$\bar{I} = 00000001 \supset \bar{A} \Rightarrow \bar{A} + \bar{\bar{D}}$$

$$\bar{I} = 00000010 \supset \bar{A} \Rightarrow \bar{A} + \bar{\bar{D}}$$

etc.

Yet the corresponding 'if ··· ACTION-IS - - -' statements are all false.

As is well known, the problem of analysing counterfactual-supporting statements is a hard one – so hard, indeed, that many logicians like to pretend that a first-order extensional language suffices for all scientific purposes (see for example Quine 1960). Yet an adequate theory of computation cannot be couched in a purely extensional language.

Quine proposes to paraphrase counterfactual-supporting statements by means of a relative term 'M' ('alike in structure'). For example, 'x is soluble' is to be paraphrased as ('in rough outline'): $\exists y$ (Mxy & y dissolves) (Quine 1960, 224). This proposal cannot work in the present context, since the computing device *has no* relevant structure over and above that detailed by the very axioms in the architecture-algorithm specification that are to be paraphrased. In the jargon, there are 'multiple physical realisations' of one and the same computing device. For example, M may be realised by means of valves or transistors or cogs and levers à la Babbage. The only feature that such disparate physical entities need have in common is that all are realisations of M.

Turing himself gives expression to this strong dependency relationship by means of the phrase 'completely determined by': each action of a Turing machine is completely determined by its configuration. Here is his elegant presentation of his analysis of computation.

> We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions $q_1, q_2, \ldots, q_k$ which will be called "m-configurations". The machine is supplied with a "tape" (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the r-th, bearing the symbol $S(r)$ which is "in the machine". We may call this square the "scanned square". The symbol on the scanned square may be called the "scanned symbol" ... The possible behaviour of the machine at any moment is determined by the m-configuration q_n and the scanned symbol $S(r)$. This pair q_n, $S(r)$ will be called the "configuration": thus the configuration determines the possible behaviour of the machine ... If at each stage the motion of a machine ... is *completely* determined by the configuration, we shall call the machine an "automatic machine" ... In some of the configurations in which the scanned square is blank (i.e. bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to the right or left. In addition to any of these operations the m-configuration may be changed ... It is my contention that these operations include all those which are used in the computation of a number. (Turing 1936, 231–2)

4.

In this section I outline the proof of the theorem toward which Searle gestures in the quotation given in Section 1, viz.:

> For any entity x (with a sufficiently large number of discriminable parts) and for any architecture-algorithm specification y there exists a labelling scheme L such that $\langle x, L\rangle$ is a model of y.

The strategy is to pick some arbitrary physical entity e with a comfortably large number of discriminable parts – Searle's wall or Hinck's pail will do nicely – and show that there exists a labelling of e which enables one to interpret the axioms for M in such a way that each of them is true of e. One then performs a universal generalisation: the proof makes no essential use of features peculiar to e or M and so the result holds for any architecture-algorithm specification and for any entity (with a sufficiently large number of discriminable parts).

The first thing to be done is to settle on a way of correlating binary numbers with physical structure. Let's simply grant Searle, Hinckfuss, Lycan et al. a method that enables one to correlate unique binary numbers with regions of whatever physical object is in question. For instance, if the wall has a high polymer content then the following simple method can be used: when the number of polymer chains that end in a given space S is odd then S tokens 0, and when the number is even S tokens 1.[6]

There is no hope of taking the three registers I, D and A to be three particular regions of the wall. There are two fairly obvious reasons for

TABLE I

Cycle	$\bar{I}$	$\bar{D}$	$\bar{A}$	$\bar{\bar{D}}$
1	i_1	d_1	a_1	x_1
2	i_2	d_2	a_2	x_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	i_n	d_n	a_n	x_n

this. (1) We require the contents of the registers to remain constant unless altered by an instruction. M may, for example, be running a program whose function is to raise the number in D to a given power. The program does this by repeatedly adding the contents of D to A. Clearly the contents of D must remain constant throughout the computation. There is no reason to expect the selected physical property (or properties) to remain constant in a given region for precisely the required period. (2) We require the contents of the registers to change appropriately during the execution of a program. We cannot expect a mere wall to have physical properties that change in a way responsive to the sequencing of instructions in whatever program M happens to be running. Thus there is no region that can serve as the referent of 'I'. It would be equally absurd to expect the physics of the wall to be responsive to the demands of binary arithmetic; so there is no region that can serve as the referent of 'A'. The solution is to use entities of greater abstraction as referents of the terms 'I', 'A' and 'D'. (I call this procedure 'Searlification'.) Each of these terms is to be interpreted as a function whose value at any point in the computation is the contents of the register at that point. (A similar construction serves to interpret the indirect address '$\bar{\bar{D}}$'.) If an explicit treatment were being given of those of M's axioms that contain terms designating an input register and an output register then these terms would be dealt with in just the same way.

To make matters specific let's suppose that M (the real M) is running a particular program, say Wordstar, and let's consider a particular run of the program lasting, say, for n of M's clock-cycles. (M performs one instruction per cycle or 'beat' of its internal clock.) Table I shows the contents of I, A, D and $\bar{D}$ at the end of each cycle of the run (thus the column headed $\bar{A}$ shows the contents of the accumulator *after* that cycle's instruction has been executed). The result is a table of *labels*. Columns two through five display not physical states of components of M but binary numbers that label such states.

Each cycle c ends (we may suppose) at a precise moment t_c. This will be a moment in the history of the wall. Call such moments the designated

TABLE II

T	$\lvert\bar{I}\rvert$	$\lvert\bar{D}\rvert$	$\lvert\bar{A}\rvert$	$\lvert\bar{\bar{D}}\rvert$
t_1	$\lvert i_1\rvert$	$\lvert d_1\rvert$	$\lvert a_1\rvert$	$\lvert x_1\rvert$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_n	$\lvert i_n\rvert$	$\lvert d_n\rvert$	$\lvert a_n\rvert$	$\lvert x_n\rvert$

moments. Using the hypothesised method for correlating binary numbers with regions of the wall one can assign distinct regions of the wall as it is at a designated moment t_c to each of i_c, d_c, a_c and x_c. Making these assignments yields a second table. Unlike Table I this is a table not of labels but of regions of the wall. I write $|i_1|$ to name the region assigned to i_1, and so on; $|i_1|$ may be pronounced 'the molecular encoding of the binary number i_1'. The terms in the first column refer to the designated moments for this particular run of the program, in the second column to the molecular encodings of the contents of I, and so on.

Tables 1 and 2 taken together constitute the desired labelling of the wall. The next three paragraphs develop some terminology required for the demonstration that there exists an interpretation of M's axioms such that each is true of the wall so labelled. (Readers who are prepared to take the details for granted may wish to skip them.)

Let **I** be the function whose domain is the set T (i.e. the set of designated moments for this run of the program) and whose range is the set $|\bar{I}|$, and similarly for **D**, **A**, and **X**. **I** will be used to interpret occurrences of the symbol 'I' in the axioms, **D** to interpret occurrences of 'D' and **A** of 'A'. **X** interprets the indirect address used in $Ax4$.

t' will be used to denote the designated moment immediately preceding a given designated moment t. Notice that $\mathbf{A}(t')$ represents the state of the accumulator which immediately precedes the state $\mathbf{A}(t)$.

Let t_j be any designated moment such that i_j = 00000010 (i.e. such that i_j is the instruction to add the contents of D to the contents of A). Consider the set of all ordered pairs $\langle\langle\mathbf{A}(t_j'), \mathbf{D}(t_j)\rangle, \mathbf{A}(t_j)\rangle$, one for each such moment t_j. By construction, this set is a function. Call it PLUS. Similarly, let t_k be a designated moment such that $i_k = 00000011$ (i.e. the instruction to multiply the contents of D by the contents of A). Consider the set of all ordered pairs $\langle\langle\mathbf{A}(t_k'), \mathbf{D}(t_k)\rangle, \mathbf{A}(t_k)\rangle$. Again this set is a function; call it TIMES. Lastly, let t_l be a designated moment such that $i_l = 00000100$ (i.e. the instruction to add the contents of the register whose address is stored in D to the contents of A). The function PLUS† is the set $\langle\langle \mathbf{A}(t_l'), \mathbf{X}(t_l)\rangle, \mathbf{A}(t_l)\rangle$.

The displayed axioms for M are interpreted as follows (the quantifier ranges over designated moments):

$$Ax1\colon \forall t(\mathbf{I}(t) = |00000001| \supset \mathbf{A}(t) = \mathbf{D}(t))$$

$$Ax2\colon \forall t(\mathbf{I}(t) = |00000010| \supset \mathbf{A}(t) = \text{PLUS}(\langle A(t'), \mathbf{D}(t)\rangle))$$

$$Ax3\colon \forall t(\mathbf{I}(t) = |00000011| \supset \mathbf{A}(t) = \text{TIMES}(\langle \mathbf{A}(t'), \mathbf{D}(t)\rangle))$$

$$Ax4\colon \forall t(\mathbf{I}(t) = |00000100| \supset \mathbf{A}(t) = \text{PLUS}^\dagger(\langle \mathbf{A}(t'), \mathbf{D}(t)\rangle)).$$

It is easily verified that each of these statements is true.

Under the interpretation provided, then, M's axioms are true of the wall. This is the technical analogue of Searle's claim that 'there is some pattern of molecule movements that is isomorphic with the formal structure of Wordstar' (Searle 1992, 209). Since the construction I am calling Searlification is quite generally applicable the argument will go through no matter what entity and what architecture-algorithm specification are under consideration (subject only to the usual fine print about the cardinality of the parts). This completes the proof of Searle's Theorem.

Reflection on the model presented in this section will yield the promised criteria for guarding the analysis given above of the predicate 'is computing the function f' from models of an architecture-algorithm specification that are of the wrong kind to sustain the analysis.

5.

A nonstandard interpretation of a theory is an interpretation that does not respect the intended meanings of the terms of the theory.[7] (When mathematical logicians speak of nonstandard interpretations of number theory, analysis and set theory they have in mind interpretations whose domains are not isomorphic to the domain of the intended interpretation. In the more general sense of the term 'nonstandard interpretation' in use here the domain of such an interpretation may or may not be isomorphic to the domain of the intended interpretation.) For example, a nonstandard interpretation of some statements concerning European geography might assign the number 1 as referent of the symbol 'London' and the number 16 as the referent of 'Moscow'. Sentences of the form 'a is north of b' might be assigned truth conditions of the form 'the referent of "b" < the referent of "a" '. In this modelling the sentence 'Moscow is north of London' is true, but is no longer about Moscow and London.

What is perhaps the most spectacular nonstandard interpretation of them all is due to Skolem. Building upon earlier work by Löwenheim he proved that if there is some interpretation under which the axioms of a theory are true, then they are true under an interpretation whose domain is the set of natural numbers (for any first-order theory whose language is countable and contains the identity predicate).[8] The result of applying this theorem to the theory of real numbers is known as Skolem's paradox. It is an axiom (or theorem) of real number theory that the cardinality of $\mathbb{N}$ (the set of all natural numbers) is less than the cardinality of $\mathbb{R}$ (the set of all real numbers). (That is, the set of all real numbers is larger than any set of natural numbers.) Yet the Löwenheim–Skolem theorem tells us that this axiom is true under an interpretation that countenances nothing but natural numbers and sets thereof. How, one might ask, can a sentence entailing the existence of a set larger than any set of natural numbers be true in a universe where there are no sets other than sets of natural numbers? Or to take a more mundane example, how can the sentence 'Skolem is Norwegian', which entails '$\exists x$ (x is Norwegian)', be true under an interpretation that countenances no Norwegians, only Swedes?

There is no real difficulty here. Perhaps the latter interpretation assigns the following truth-condition:

'Skolem is Norwegian' is true iff the referent of 'Skolem' $\in K$

where K is some subset of the domain (perhaps the set of unmarried members of the domain). Provided the object assigned as the referent of the symbol 'Skolem' is indeed in K, the sentence 'Skolem is Norwegian' is true under the interpretation; but it is clearly no longer about Skolem and Norwegians. In the case of an interpretation within the natural numbers of real number theory, some set of naturals is assigned as the referent of the symbol '$\mathbb{R}$'. The sentence 'card $\mathbb{N}$ < card $\mathbb{R}$' is made true by the expedient of ensuring that the model contains no $1-1$ function that maps this set onto whatever set is assigned as referent of '$\mathbb{N}$'. (Similarly one could make the sentence 'Moscow is the most northerly European city' true in the model mentioned earlier by choosing not to include any numbers greater than 16 in the domain of the model.) In summary, while axioms containing the term '$\mathbb{R}$' are true under the nonstandard interpretation I have sketched, the axioms thus interpreted are no longer about the set of all real numbers.

The interpretation I have described for the axioms of M is a nonstandard one. We may say that the wall "computes" and mean by this nothing more than that the axioms for M, a computer, are true under the interpretation I have described. However, it would be an elementary error to infer from this that the wall computes in any genuine sense – just as it would be an error to infer that some countable sets are uncountable from the fact that the axiom 'card $\mathbb{N} <$ card $\mathbb{R}$' is true in Skolem's countable model of number theory. The intended meaning of this axiom is that $\mathbb{R}$ is uncountable (i.e. is larger than any set of natural numbers), but this does not mean that if the axiom is true under a nonstandard interpretation then whatever $\mathbb{R}$ refers to under that interpretation is uncountable. Similarly, under their intended interpretation the axioms for M add up to the proposition that M is a von Neumann computer, but this does not mean that if the axioms are true under a nonstandard interpretation then whatever they are true of under that interpretation is a von Neumann computer.

Where the pair $\langle e, L\rangle$ is a model for some SPEC *but only under a nonstandard interpretation* of SPEC I will say that $\langle e, L\rangle$ is a nonstandard model of SPEC. A model that is not nonstandard will be called *honest*. Searle's Theorem shows that according to the analysis given in Section 2 the predicate 'is computing the function f' is trivially true of very many entities. I suggest the following modification to the analysis:

Entity e is computing function f if and only if there exist a labelling scheme L and a formal specification SPEC (of an architecture and an algorithm specific to the architecture that takes arguments of f as inputs and delivers values of f as outputs) such that $\langle e, L\rangle$ is an *honest* model of SPEC.

The claim, then, is that the model constructed in Section 4 is a nonstandard one. Why exactly? What are the grounds for insisting that the model fails to respect the intended meanings of the terms of the axiomatic theory? These are threefold. First, the axiomatic theory under consideration and those like it – the various dynamic logics, for example – are intended (in a phrase of Segerberg's (1989, 248)) as logics of computer *action*. However, all the computational activity occurred *outside* the wall, in the course of obtaining Table I, which is simply a record of the activity within the cpu of the machine that actually performed the computation. Once Table I is secured the labelling scheme is constructed from it ex post facto. The wall under this novel description is at most a passive 'scoreboard' and is no more an active participant in the computation than the scoreboard is an active player in a game of billiards. The axioms for M certainly contain the term 'ACTION-IS' but the fact that the axioms (as interpreted) are true of the wall goes no way toward showing that the wall *acted* in accordance with the instructions in the algorithm. The wall so acted only if the referent of '$\mathbb{R}$' in Skolem's countable model is uncountable!

Second, the nonstandard interpretation introduces unintended temporal specificity into the theory. The labelling scheme used in the model is, of course, incomplete in that the scheme identifies the regions of the wall that bear labels at the designated moments $t_1, \ldots, t_n$ but provides no information concerning which regions of the wall are the label-bearers at times prior to t_1 or subsequent to t_n. This is in sharp contrast to the unmarked case, where the labelling scheme remains applicable throughout the lifetime of the entity (assuming no hardware modifications), certain fixed regions of the entity, or disjunctions of such, being designated ab initio as label-bearers. This incompleteness in the labelling scheme shows up in the interpretation of the construction 'if $\cdots$ ACTION-IS - - -', via the presence of the universal quantifier ranging over the designated moments $t_1, \ldots, t_n$. The axioms as interpreted have no entailments concerning times lying outside this range, whereas under their intended interpretation the axioms are (and entail) conditional statements that are true at any moment during the normal functioning of the device. (Is the wall perhaps in a state of malfunction at all times prior to t_1 and subsequent to t_n? Hardly. At the whim of the modeller the wall can be made to run the program again, say through the moments $t_{n+1}, \ldots, t_{2n}$.) Moreover, under the nonstandard interpretation the whole axiomatisation is, so to speak, in the past tense, whereas to describe a physical entity as a computing machine of a certain kind is to envisage being able to *pre*dict aspects of its physical behaviour on the basis of its architecture-algorithm specification and its labelling scheme. That is to say, the intended meaning of the term 'ACTION-IS' certainly involves no restriction to past actions. Yet it is necessarily the case that under the nonstandard interpretation each axiom is a statement about the past. It is, of course, the ex post facto nature of the labelling scheme that introduces these unwanted temporal specificities into the interpretation.

Third, the construction 'if $\cdots$ ACTION-IS - - -' is interpreted by means of material implication (the truth-condition assigned to each axiom is a universally quantified material implication). Nor can a better interpretation be found for this construction within the context of an attempt to show that the wall serves to model M's axioms. There are possible worlds differing minimally from the actual world in which the axioms as interpreted have true antecedents and false consequents. Consider:

$$\forall t(\mathbf{I}(t) = |00000011| \supset \mathbf{A}(t) = \text{TIMES}(\langle \mathbf{A}(t'), \mathbf{D}(t) \rangle)).$$

Suppose that in the actual world $\mathbf{I}(t) = |00000010|$. That is to say, the region which is the value of **I** at the moment t is a molecular encoding of the binary number 00000010 (this number being M's instruction to add $\bar{D}$ to $\bar{A}$). Take a possible world w in which the properties of this same region are just sufficiently different to make it a molecular encoding of the binary number differing from the foregoing only in its least significant digit. The regions denoted by $\mathbf{A}(t)$, $\mathbf{A}(t')$ and $\mathbf{D}(t)$ code the same numbers in w as they do in the actual world. (If quantum mechanics is even roughly true then such a world is not only nomologically possible but may also be indistinguishable from the actual world through all history up to a time t minus delta.) Say for definiteness that $\mathbf{A}(t') = |00000001|$ and $\mathbf{D}(t) = |00000010|$. So from Table II $\mathbf{A}(t) = |00000011|$. Suppose further that the computation represented in Table II is such that

$$\exists u(\mathbf{I}(u) = |00000011|\,\&\,\mathbf{A}(u') = |00000001|\,\&\,\mathbf{D}(u) = |00000010|).$$

Since TIMES is a function it follows that

$$\text{TIMES}(\langle \mathbf{A}(t'), \mathbf{D}(t)\rangle) = |00000010|.$$

So it is true in w that $\mathbf{I}(t) = |00000011|$ and false in w that $\mathbf{A}(t) = \text{TIMES}(\langle \mathbf{A}(t'), \mathbf{D}(t)\rangle)$. There is then, no conditional connective $\blacksquare\!\!\rightarrow$ with a possible worlds semantics such that

$$\forall t(\mathbf{I}(t) = |00000011| \blacksquare\!\!\rightarrow \mathbf{A}(t) = \text{TIMES}(\langle \mathbf{A}(t'), \mathbf{D}(t)\rangle))$$

is true. The same goes for the remaining axioms.

Why does it matter that 'if ··· ACTION-IS - - -' has been interpreted as material implication? Because, as previously remarked, the axioms under that interpretation fail to support assertions about the counterfactual behaviour of M. Whereas given the intended meaning of the 'if ··· ACTION-IS - - -' construction the axioms do licence assertions about the counterfactual behaviour of M. (For example, under its intended interpretation axiom 3 licences the assertion that if 00000011 had been in the instruction register at t then the machine would have performed the replacement shown in the axiom.) 'If ··· ACTION-IS - - -' expresses a stronger relationship than material implication and that is the third reason to say that the interpretation fails to respect the intended meanings of terms occurring in the axioms. The strong dependency relationship that Turing refers to in the course of explaining the concept of a computing machine is simply absent from the 'machine' consisting of labelled regions of the wall. Only if Turing had made no mention of this strong dependency relationship would Searle's criticism of him in the above quotation be just.

In summary, I suggest two necessary conditions for honesty. First, the labelling scheme must not be ex post facto. This requirement guards the class of honest models from intruders that fail to respect the intended meanings of the terms of the axiomatic theory in ways of the sort outlined in the first and second of the above three criticisms. Second, the interpretation associated with the model must secure the truth of appropriate counterfactuals concerning the machine's behaviour. Either of these two requirements suffices to debunk the alleged problem cases.

6.

This section criticises some alternative answers to the question 'What is computation?'.

1. Deutsch gives the following characterisation of computation:

> [A] computing machine is any physical system whose dynamical evolution takes it from one of a set of 'input' states to one of a set of 'output' states. The states are labelled in some canonical way, the machine is prepared in a state with a given input label and then, following some motion, the output state is measured ... [T]he measured output label is a definite function f of the prepared input label; ... the machine is said to 'compute' the function f. (Deutsch 1985, 97)

This account falls easy prey to Searle's Theorem. Hinck's pail is a physical system whose dynamical evolution takes it from an 'input' state to an 'output' state. The states are labelled in some canonical way by means of a suitable encoding/decoding function. Someone who is, so to speak, told exactly where to look in the bucket can measure the 'output' state and determine its label. So the bucket is computing the function that relates the input labels to the output labels. Of course, the labelling scheme in question is entirely ex post facto (as in Tables I and II).

There is something else very wrong about Deutsch's analysis: it makes no mention of the notion of an algorithm. In my view this notion should lie at the heart of an account of computation. Consider a clockwork clock. The hands show 12. This is the 'input' state. The label is simply the numerical specification of the time shown. The clock runs for n minutes. The ensuing position of the hands is the 'output' state. Is the clock computing the function that relates the input label to the output label? According to Deutsch's account Yes; according to my account No. The elementary steps of an algorithm for deriving the output labels from the input labels might consist, for example, of multiplications, additions and subtractions. No honest way of modelling a specification of such an algorithm and an architecture for implementing it (i.e. an architecture that makes these elementary operations available) will be able to locate such steps in the functioning of the clock. Similar remarks apply to the issue of whether the solar system computes solutions to its own equations of motion (Section 2 above). According to Deutsch's account it does but according to an account that takes the notion of an algorithm and its supporting architecture seriously the solar system does not compute. Nonstandard models of any architecture-algorithm specification are easily obtained but there is no reason to think that the challenge of providing an honest model based on the solar system of a suitable algorithm and supporting architecture can be met.

The examples of the clock and the solar system generalise. Any physical system that is describable as producing 'output' from 'input' – and is there any that is not? – has a labelling in Deutsch's sense, and so on his account the system computes the function relating the input labels to the output labels. In consequence the claim that the brain computes loses its status as a serious empirical hypothesis. Protect Deutsch's account from the radical trivialisation it suffers at the hand of Searle's Theorem (by means of the apparatus detailed in Section 5) and the account will still be too wide.

2. The analysis of computation presented by Cummins is responsive to intuitions such as these concerning the clock and solar system. He writes:

> [F]unctions need not be computed to be satisfied. Set mousetraps satisfy a function from trippings to snappings without computing it, and physical objects of all kinds satisfy mechanical functions without computing them. The planets stay in their orbits without computing them. (Cummins 1989, 91)

He glosses the notion of a function's being satisfied: a device satisfies a function g when 'the arguments and values of g are literally states of' it (Cummins 1989, 89).

> Functions associate values with arguments. To see a device as satisfying a function, therefore, is to see it as having inputs and outputs . . . and to see these as *arguments* and *values*. (Cummins 1989, 164)

Cummins proposes the following analysis of when a function is being computed:

> Computing reduces to program execution, so our problem reduces to explaining what it is to execute a program. The obvious strategy to exploit is the idea that program execution involves *steps*, and to treat each elementary step as a function that the executing system simply satisfies . . . Program execution reduces to step satisfaction. (Cummins 1989, 91–92)

This analysis too is trivialised by Searle's Theorem, as is easily shown by considering the elementary steps of a program for the machine M. A typical example of such a step is

> Add the contents of D to the contents of A and store the result in A.

We may regard this step as taking two binary numbers as input and delivering a single binary number as output. Cummins invites us to treat each elementary step as a function that an executing system can satisfy. In this case the function will be one whose values are label-bearing states of the executing system and whose arguments are pairs of such states. Under the labelling proposed in Section 4 the wall satisfies the function in question. On each clock cycle x at which the register **I** contains the instruction for performing the above step it is the case that

$$\mathrm{PLUS}(\langle \mathbf{A}(t'_x), \mathbf{D}(t_x)\rangle) = \mathbf{A}(t_x).$$

The same goes for all the other elementary steps of the program. So according to Cummins' account the wall is executing the program.

Moreover Cummins regards computation as a causal process:

> [A] computation [isl a causal process specified abstractly as an algorithm. (Cummins and Schwarz 1991, 63)

As I remarked earlier, a causal account of computation lacks the necessary generality. Abstract devices such as Turing machines compute. Consider the case of a programmed simulation of a Turing machine. The virtual Turing machine is computing, but there are no causal relationships between the contents of the virtual machine's tape and the virtual machine's actions. The relevant causal network has activities of the underlying real machine at its nodes, not activities of the virtual machine.

An analysis of computation in terms of causation is intolerably narrow. Even if that fact is disregarded Cummins cannot protect his step-satisfaction account of computation from trivialisation by adding a rider to the effect that if states are to satisfy a function then the value-state must be a *causal outcome* of the argument-state(s). For there exists a labelling of the wall such that label-bearing states satisfy the appropriate functions even in this stronger sense of 'satisfy'. (Or if that's not true then pick a larger wall!) The trick, of course, is to select states for labelling that *are* causally related in the required ways. Staying with the previous example of an elementary step, the state $|a_x|$ must be chosen in such a way that (as well as having the right structural properties to bear the label a_x under the encoding/decoding function being used) it is a causal outcome of the states $|a_{(x-1)}|$, $|d_x|$. The function PLUS is then satisfied in the stronger sense.

3. Lycan's own way of dealing with Hinck's pail is not, as it stands, successful. He writes:

> [W]e see... why Block's group organisms are admissible as sentient beings but Hinckfuss's pail of water is not: the homunculi-head ... incorporate[sl Φ-ers, Ψ-ers, and "-ers" of countless other types, courtesy of the bureaucrats who are doing all the work; the pail of water does not contain any "-ers" of any kind that is mentioned in a homunctionalist

> program, precisely because it is not organised in the relevant way, even if the de facto motions of some of the molecules in the pail happen to ape the motions that would be made by an organism that *was* functionally organised on the human model. (Lycan 1981, 41; see also 1987, 34)

As ever, let M's axioms serve as an example – simplistic but adequate for the purpose in hand – of a functionalist architectural specification. Lycan's thought is that the specification requires certain kinds of ers that the pail cannot supply. But let $\text{er}_1 \ldots \text{er}_n$ be a list of the various ers involved in M's architecture (instruction register, accumulator, etc.) and consider a model founded on the pail of the sort described in Section 4. Under the interpretation associated with the model a referent for each term 'er_1', ..., 'er_n' is indeed furnished by the pail; so Lycan's claim is false. (To be sure each of M's registers is realised in a highly distributed way in the pail or wall, but then so would they be in a connectionist modelling of M.) Lycan's attempt to defuse the problem cases simply underestimates the Skolemesque tactics that underlie them.

7.

It is the computational theory of mind, and not extant definitions of computation, that is the principal target of Searle and Hinckfuss. Numerous other colourful problem cases are to be found in the critical literature on computational functionalism. All the various cases can be divided into two groups. Cases in Group I are typified by Hinck's pail and the Wordstar wall. All can be dealt with by means of the distinction between nonstandard and honest models developed here. The cases in Group II are typified by Block's ingenious and well-known thought experiments involving the economy of Bolivia and the population of China. Group II cases pose a prima facie challenge to computational functionalism but have no tendency – not even a prima facie one – to show that Turing's analysis of computation is trivialisable and so are not germane to the present discussion. Nevertheless it is worthwhile briefly examining one of Block's examples in order to establish that it does indeed pose no challenge to the account of computation offered here (and nor, for that matter, to Deutsch's or Cummins' account). I shall suggest moreover that the Group II cases in fact have no force against computational functionalist theories of mind.

If computational functionalism is true then there exists an architecture-algorithm specification SPEC-BLOCK such that if for any system x and labelling L $\langle x, L\rangle$ is an honest model of SPEC-BLOCK then x is cognitively equivalent to Block. 'Suppose', says Block, 'we convert the government of China to functionalism, and we convince its officials that it would enormously enhance their international prestige to realize a human mind for an hour. We provide each of the billion people in China ... with a specially designed two-way radio that connects them in the appropriate way to other persons ... ' (Block 1978, 279). If each of the billion people is told to obey an appropriate instruction and does so faithfully then the resulting realisation of SPEC-BLOCK will be a system that is cognitively equivalent to Block. (In Block's own description of the scenario each person is given an instruction corresponding to a single line in a Turing machine table; the precise nature of the instructions will vary depending on the computational architecture in question.)

Block writes:

> Remember that a machine table specifies a set of conditionals of the form: if the machine is in S_i and receives input I_j it emits output O_k and goes into S_l. Any system that has a set of inputs, outputs and states related in the way described realizes that machine table, even if it exists only for an instant. For the hour the Chinese system is "on", it *does* have a set of inputs, outputs and states of which such conditionals are true. (Block 1978, 279)

What Block says here is unpersuasive. For there exists a labelling of the wall such that these indicative conditionals are true of the wall, yet the wall does not realise a program or machine table. As Searle's Theorem shows, it is simply false that 'any system that has a set of inputs, outputs and states related in the way described realises that machine table'. The points Block ought to emphasise are that the labelling of the Chinese system is not ex post facto and that the physical design of the system underwrites all the relevant counterfactuals.

There can be no dispute over the fact that Block's system would compute. It genuinely runs the algorithm, genuinely computes the function involved. The cutting edge of hardware engineering consists in the search for hitherto unconsidered ways of realising computational architectures (witness the progression from mercury delay lines to cathode ray tube storage to semiconductors). To match Block's fantastical computer with another: if it turns out that the state transitions in the digestive system of the Chinese silkworm can be manipulated so that – under a certain labelling – the digestive system forms a practicable and ultra cheap realisation of the architecture-algorithm specification commonly used by manufacturers of handheld calculators, then this fact will be seized upon by engineers eager to exploit it, not regarded as a counterexample to any account of computation that entails it. Block's hitherto unconsidered way of physically realising an architecture-algorithm specification is simply that.

Is Block's fantastical system really a counterexample to computational functionalism? No. The system is a genuine realisation of SPEC-BLOCK, but it is a realisation that could exist only in fairyland. I have no firm intuitions about fairyland, save that one should expect the bizarre; yet Block's argument against functionalism is essentially an appeal to an intuition that he expects us to have concerning the system, namely that it 'lack[s] mentality' (Block 1978, 277). I certainly have no inclination to insist that in fairyland this brain-of-slaves would lack mentality.

8.

Searle's concession (in the passage quoted in Section 1) that it is 'possible to block ... universal realisability' – by which he means block the result that 'everything would be a digital computer' (Searle 1992, 208) – seems to me to overturn his main argument.[9] Here is Searle's summary of the relevant portion of his argument:

> The point is not that the claim 'The brain is a digital computer' is simply false. Rather, it does not get up to the level of falsehood. It does not have a clear sense. The question 'Is the brain a digital computer?' is ill defined. If it asks, 'Can we assign a computational interpretation to the brain?' the answer is trivially yes, because we can assign a computational interpretation to anything. If it asks, 'Are brain processes intrinsically computational?' the answer is trivially no, because nothing is intrinsically computational ... (Searle 1992, 225)

Searle offers the claim that brains are not '*intrinsically* digital computers' as a consequence of his claim that 'syntax is not intrinsic to physics' (Searle 1992, 208, 225). The latter claim is certainly true: the labels are not intrinsic to the physics of the labelled device. There are no discrete binary states intrinsic to the physics of my Macintosh. Binary labels are attached according to whether certain non-discrete variables (e.g. voltage, degree of magnetisation) fall within one or the other of two ranges. If the truism that syntax is not intrinsic to physics implies that brains are not 'intrinsically digital computers' then by parity it implies that no entity is intrinsically a digital computer. In Searle's sense there is nothing 'intrinsically computational' about neuron-firings; and nor is there anything 'intrinsically computational' about micro-chip events. Searle is telling us no more than that if the brain is a computer, then it is so only in the sense in which all other computers are computers. This is hardly interesting.

It is Searle's first way of interpreting the question 'Is the brain a digital computer?' that is the important one: 'Can we assign a computational interpretation to the brain?'. (Searle is careful to distinguish the question 'Can we assign a computational interpretation to the brain?' from the question 'Can the operations of the brain be simulated on a digital computer?' (Searle 1992, 200).) His claim that the answer to the question thus interpreted is 'trivially yes' is mistaken: 'universal realisability' is false, as I have argued.

The rock-bottom issue in cognitive science is precisely whether, and to what extent, the brain *can* be assigned 'a computational interpretation'. If the argument presented here is correct, this is an empirical issue. It is always an empirical question whether or not there exists a labelling of some given naturally occurring system such that the system forms an honest model of some architecture-algorithm specification.[10] And notwithstanding the truism that 'syntax is not intrinsic to physics' the discovery of this architecture-algorithm specification and labelling may be the key to understanding the system's organisation and function.

NOTES

[1] I am grateful to Philip Catton, Mike Resnik, John Searle, Krister Segerberg, Kim Sterelny and Tim Williamson for comments on earlier versions of this material.

[2] But not the necessity of Turing's analysis *as usually interpreted*, that is, as embodying the so-called Church-Turing thesis. See my 1996b, 1997 and note 3 below.

[3] Where the weights, thresholds, activation levels etc. are specifiable by means either of rational numbers or else real numbers that are computable in Turing's sense of 'computable number' then the step-by-step procedure is what I call a *classical* algorithm; where other real numbers may figure as weights etc., the step-by-step procedure is a *nonclassical* algorithm. Any Turing-equivalent device can execute only classical algorithms. The class of architectures capable of executing nonclassical algorithms is diverse and includes purely digital machines. I develop the classical/nonclassical distinction in 1996b and 1997.

[4] I am grateful to Philip Catton for drawing Newman's article to my attention. Newman's objection to Russell is discussed by Demopoulos and Friedman 1985.

[5] See, for example, Belnap 1996, Segerberg 1996.

[6] I owe this suggestion to Philip Catton.

[7] See also Copeland 1979, 1986 and 1994.

[8] Chang and Keisler 1973, 66–68.

[9] I discuss other aspects of Searle's attack on cognitive science in 1993a and 1993b chapters 6 and 10.

[10] For another argument, quite different in approach from the one presented here, in support of the claim that the question 'Is the brain a digital computer?' is an empirical one, see Proudfoot and Copeland 1994.

REFERENCES

Belnap, N. D.: 1996, 'Agents in Branching Time', in Copeland 1996a.

Block, N.: 1978, 'Troubles with Functionalism', in Savage 1978, pp. 261–325.

Chang, C. C. and Keisler, H. J.: 1973, *Model Theory*, North Holland, Amsterdam.

Copeland, B. J.: 1979, 'On When a Semantics is not a Semantics', *Journal of Philosophical Logic* **8**, 399–413.

Copeland, B. J.: 1986, 'What is a Semantics for Classical Negation?', *Mind* **XCV**, 478–90.

Copeland, B. J.: 1993a, 'The Curious Case of the Chinese Gym', *Synthese* **95**, 173–86.

Copeland, B. J.: 1993b, *Artificial Intelligence: a Philosophical Introduction*, Blackwell, Oxford.

Copeland, B. J.: 1994, 'Vagueness and Bivalence', *Proceedings of the Aristotelian Society*, **LXVIII**, 193–200.

Copeland, B. J. (ed.): 1996a, *Logic and Reality: Essays on the Legacy of Arthur Prior*, Oxford University Press, Oxford.

Copeland, B. J.: 1996b, 'The Broad Concept of Computation', Forthcoming.

Copeland, B. J.: 1997, *Turing's Machines*, Oxford University Press, Oxford.

Copeland, B. J. and Proudfoot, D.: 1996, 'On Alan Turing's Anticipation of Connectionism', *Synthese* **108**, this issue.

Cummins, R.: 1989, *Meaning and Mental Representation*, MIT Press, Cambridge, Mass.

Cummins, R. and Schwarz, G.: 1991, 'Connectionism, Computation, and Cognition', in Horgan and Tienson 1991, 60–73.

Demopoulos, W. and Friedman, M.: 1985, 'Bertrand Russell's *The Analysis of Matter*: Its Historical Context and Contemporary Interest', *Philosophy of Science* **52**, 621–39.

Deutsch, D.: 1985, 'Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer', *Proceedings of the Royal Society*, Series A, **400**, 97–117.

Fodor, J. A.: 1975, *The Language of Thought*, Thomas Y. Crowell, New York.

Gandy, R.: 1980, 'Church's Thesis and Principles for Mechanisms', in J. Barwise, H. J. Keisler, and K. Kunen (eds): 1980, *The Kleene Symposium*, North-Holland, Amsterdam, pp. 123–148.

Goel, V.: 1991, 'Notationality and the Information Processing Mind', *Minds and Machines* **1**, 129–165.

Goel, V.: 1992, 'Are Computational Explanations Vacuous?', *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Erlbaum, Hillsdale, New Jersey, pp. 647–52.

Horgan, T. and Tienson, J.: 1991, *Connectionism and the Philosophy of Mind*, Kluwer, Dordrecht.

Lycan, W. G.: 1981, 'Form, Function, and Feel', *Journal of Philosophy* **78**, 24–50.

Lycan, W. G.: 1987, *Consciousness*, MIT Press, Cambridge, Mass.

Newman, M. H. A.: 1928, 'Mr. Russell's "Causal Theory of Perception" ', *Mind* **37**, 137–48.

Proudfoot, D. and Copeland, B. J.: 1994, 'Turing, Wittgenstein and the Science of the Mind', *Australasian Journal of Philosophy* **72**, 497–519.

Quine, W. V. O.: 1960, *Word and Object*, MIT Press, Cambridge, Mass.

Russell, B.: 1927, *The Analysis of Matter*, Kegan Paul, Trench, Trubner, London.

Russell, B.: 1968, *The Autobiography of Bertrand Russell*, Vol. 2, Allen & Unwin, London.

Savage, C. W. (ed.): 1978, *Perception and Cognition: Issues in the Foundations of Psychology*, University of Minnesota Press, Minneapolis.

Searle, J.: 1990, 'Is the Brain a Digital Computer?', *Proceedings and Addresses of the American Philosophical Association* **64**, 21–37.

Searle, J.: 1992, *The Rediscovery of the Mind*, MIT Press, Cambridge, Mass.

Segerberg, K.: 1989, 'Getting Started: Beginnings in the Logic of Action'. Atti del Convegno Internazionale di Storia della Logica, *Le teorie delle modalità*; San Gimignano 5–8 dicembre 1987. CLUEB, Bologna, Italy.

Segerberg, K.: 1996, 'To Do and Not To Do', in Copeland 1996a.

Smith B. C.: forthcoming, *A View from Somewhere*, MIT Press, Cambridge, Mass.

Sterelny, K.: 1990, *The Representational Theory of Mind*, Blackwell, Oxford.

Turing, A. M.: 1936, 'On Computable Numbers, with an Application to the Entscheidungsproblem', *Proceedings of the London Mathematical Society*, Series 2, **42** (1936–37), 230–65.

Turing, A. M.: 1948, 'Intelligent Machinery', National Physical Laboratory Report, Reprinted in Meltzer, B. and Michie, D. (eds), *Machine Intelligence*, Vol. 5, Edinburgh University Press, Edinburgh, pp. 3–23.

Philosophy Department
University of Canterbury
Private Bag 4800
Christchurch
New Zealand

bjcopeland@canterbury.ac.nz