NARROW VERSUS WIDE MECHANISM: 
INCLUDING A RE-EXAMINATION OF TURING'S 
VIEWS ON THE MIND-MACHINE ISSUE *

When René Descartes claims that the human body is a machine, he is proposing a novel way of explaining or modeling the functioning of the body. Among the functions that are to be explained in this new way are the operation of the sense organs, some amount of internal processing of the resulting sensory ideas, the "stamping of these ideas in the memory," and "the internal movements of the appetites and passions" (ibid., p. 108).

I should like you to consider that these functions follow from the mere arrangement of the...organs every bit as naturally as the movements of a clock or other automaton follow from the arrangement of its counter-weights and wheels (ibid.).

Clocks and other mechanical artifacts—such as church organs, water fountains, hydraulically powered statues, and clockwork models of living creatures—were suggesting a new paradigm of explanation in human and animal physiology, in which the functions and (nonvoluntary) movements of the body are viewed as arising "just as [they] would be produced in a machine." Even the difference between a living and a dead body is to be understood in terms of "the difference between, on the one hand, a watch or other automaton...when it is wound up...and, on the other hand, the same watch or machine when it is broken and the principle of its movement ceases to be active." *

* Research on which this article draws was supported in part by University of Canterbury Research Grant No. U6271. I am grateful to Stephen Gaukroger, Diane Proudfoot, and John Sutton for valuable discussion.


3 The Passions of the Soul in The Philosophical Writings of Descartes, Volume 1, pp. 329-30.
Thomas Hobbes’s mechanist philosophy amounts in its essentials to a materialist view of mind coupled with the view that the artifactual mode of explanation—typified by the explanation of the working of a clock in terms of the nature and arrangement of its internal parts—may usefully be transferred to the study of naturally occurring systems, including our bodies and our minds:

...what is the Heart, but a Spring; and the Nerves, but so many Strings; and the Joynets, but so many Wheeles, giving motion to the whole Body, such as was intended by the Artificer?4

Julien Offray de La Mettrie5 likewise propounded a thoroughgoing mechanism, insisting against Descartes that human beings, and not just their bodies, are machines: “Let us then conclude boldly that man is a machine and that there is in the whole universe only one diversely modified substance” (ibid., p. 39). La Mettrie gleefully applied the artifactual mode of explanation to both body and mind:

...the human body is a clock...[T]he heart...is like the mainspring of the machine (ibid., p. 34).

[S]ince all the soul’s faculties depend so much on the specific organisation of the brain and of the whole body that they are clearly nothing but that very organisation, the machine is perfectly explained! ...Some wheels, a few springs more than in the most perfect animals, the brain proportionately closer to the heart and thus receiving more blood... (ibid., p. 26).

The core of the claim, as put forward by the historical mechanists, that such-and-such naturally occurring item is a machine is this: the item’s operation can be accounted for in monistic, materialist terms and in a manner analogous to that in which the operation of an artifact is explained in terms of the nature and arrangement of its parts.6 I shall

6 Other distinctive but secondary claims were endorsed by various mechanists; often such claims were emphasized more by one thinker than by others, and in any case apply more to physical explanation than to psychological or psycho-physiological explanation. These include the following (see J. E. McGuire, “Boyle’s Conception of Nature,” Journal of the History of Ideas, xxxiii (1972): 523-42, p. 523): (1) occult qualities are to be banished from explanations, which must be based on sensory experience in terms of clear and distinct ideas. (2) All natural phenomena arise from matter in motion, or matter and motion. (3) Compound bodies are composed of vortices (Descartes), centers of force (Leibniz), or microscopic corpuscles. (4) Metaphysical principles are to be integrated with experiment. (5) Nature is governed by immutable geometrical laws. (6) Regularities are to be expressed and explained in a mathematical manner. (7) Nature is to be conceived dynamically in terms of motion, rather than statically in terms solely of size and shape. (8) An important feature of Cartesian mechanism was that matter be inert, in the sense that every change in the matter’s motion is accounted for in terms of contact action. (In emphasizing the inertness, in this sense, of matter Descartes stood against “renaissance naturalism,”
refer to the proposition that the mind is a machine in the core sense of ‘machine’ embraced by the historical mechanists as historical mechanism.7

I. TWENTIETH-CENTURY MECHANISM

Under the influence of work by Alan Turing and, to a lesser extent, Alonzo Church, mechanism has taken something of a wrong turn in the twentieth century, or, at any rate, it has been steered for no very good reason in one particular direction to the exclusion of others. In 1936, Turing published his account of what he (later) called “logical computing machines,” which Church, in a review, dubbed “Turing machines.”8 A Turing machine consists of a potentially infinite paper tape, on which is written a finite number of discrete (for example, binary) symbols, and a scanner that moves back and forth along the tape symbol by symbol, reading what it finds and writing further symbols. Turing proved that one particular Turing machine, which he referred to as a “universal computing machine,” can be programmed to perform any task that any other Turing machine can perform. Twelve years later, thanks to developments in high-speed automatic switching, the universal computing machine became a reality.

The effect of Turing’s invention on mechanist thinking was marked. The classic paper of J. R. Lucas9 published in 1961 affords an excellent window on the state of affairs that existed twenty-five years after Turing’s invention. Lucas wished to refute mechanism: “Mechanism is false, that is,...minds cannot be explained as machines” (ibid., p. 112). To do so, it was, he thought, sufficient to argue that the powers of the mind exceed those of a universal Turing machine, for “any system which [is] not a Turing machine [is] not a machine within the meaning of the act” (ibid., p. 126).10 (Strictly, of course, Lucas should have said that any sys-
tem that cannot be simulated, or exactly mimicked, by a Turing machine is not a machine within the meaning of the act, for this was surely his intention; but the practice of suppressing the word ‘simulated’ and its cognates in such a context as this is a common one, and will sometimes be followed here.) Within twenty-five years of Turing’s invention, the idea that the mind is a Turing machine had become central to the mechanist conception of mind, so much so that one of mechanism’s foremost critics needed to take no cognizance of alternative mechanist conceptions. Indeed, by the time Lucas was writing, it had become endemic to mechanist thinking that there could be no alternative mechanist conception. The absence of any alternative mechanist conception was, and still is, held to be entailed by a principle supported by the logical discoveries of Turing and Church. A typical formulation of this supposed principle is: the class of possible operations that can be carried out by information-processing machinery is identical to the class of operations that can be carried out by a universal Turing machine. Turing’s biographer, Andrew Hodges, gives expression to the common perception of matters when he writes:

Alan had...discovered something almost...miraculous, the idea of a universal machine that could take over the work of any machine.... So there could be a single machine which, by reading the descriptions of other machines placed upon its “tape,” could perform the equivalent of human mental activity (ibid., p. 109).

The view that the universal Turing machine is in some appropriate sense maximal among machines is widespread in the philosophical, cognitive, and biological literature. For example, the following version of the view is from a manifesto of the artificial-life movement:

There are certain behaviors that are “uncomputable”—behaviors for which no formal specification can be given for a machine that will exhibit that behavior. The classic example of this sort of limitation is Turing’s famous Halting Problem: can we give a formal specification for a machine which, when provided with the description of any other [sic] machine together with its initial state, will...determine whether or not that machine will reach its halt state? Turing proved that no such machine can be specified.

As we shall see, it is far from the case that Turing proved any such thing.

Those who have equated machines “within the meaning of the act” with Turing machines would no doubt resist any account of themselves as revisionists who have replaced historical mechanism with a narrower thesis of their own devising. They may claim that they have merely been explicit about a point to which historical mechanism was always committed, unbeknownst to its sixteenth- and seventeenth-century supporters: had La Mettrie, for example, known of the logical discoveries of Turing and Church, he would have accepted that his mechanism entails that the mind is a Turing machine. I shall argue that the view that modern Turing-machine mechanism is simply a clarified version of historical mechanism is in error. One can uphold historical mechanism and deny Turing-machine mechanism without contradicting oneself. Let me coin the term narrow mechanism for the view that the mind is (strictly, can be simulated by) a Turing machine. A wide mechanist, on the other hand, holds that the mind is a machine but countenances the possibility of information-processing machines that cannot be mimicked by a universal Turing machine, and allows in particular that the mind may be such a machine. I shall argue that the widespread acceptance of narrow mechanism among mechanists represents an unwarranted circumscription of the mechanist tradition. The view that this circumscription is somehow necessitated by the work of Church and Turing, as encapsulated in the Church-Turing thesis, is a muddle. Specifically, I shall be claiming that: (1) mechanism does not entail narrow mechanism; (2) Turing himself, a mechanist par excellence, was not a narrow mechanist; (3) neither the Church-Turing thesis, nor any other formal or semiformal result of Church or Turing, favors narrow over wide mechanism; (4) typical arguments for narrow mechanism are vitiates by what I have called elsewhere the Church-Turing fallacy.

Lately, there have been encouraging signs that the grip of narrow mechanism is loosening. The newly emerging field known as UMC (unconventional models of computation) explores computational approaches to cognition which transgress the boundaries of narrow

---

13 Elsewhere, I term such machines hypercomputers—see my and D. Proudfoot’s “Alan Turing’s Forgotten Ideas in Computer Science,” Scientific American, CCLXXX (April 1999): 99-103.

mechanism. Also, the recent dynamical hypothesis in cognitive science is a wide-mechanist hypothesis. The new dynamicists distance themselves from the mainstream computational approach by pointing out that the dynamical hypothesis countenances dynamical (and, specifically, cognitive) systems whose behavior cannot—even in principle—be calculated by a Turing machine.

II. COMPUTERS AND COMPUTERS

It has often been remarked that when Turing uses the word ‘computer’ in his early papers, he does not employ it in its modern sense. Many passages make this obvious, as does the following example:

Computers always spend just as long in writing numbers down and deciding what to do next as they do in actual multiplications, and it is just the same with ACE...[T]he ACE will do the work of about 10,000 computers...Computers will still be employed on small calculations. (The ACE or automatic computing engine was an electronic stored-program computer designed by Turing and built at the National Physical Laboratory, London. A pilot version first ran in 1950 and at the time was the fastest computer in the world.) Turing introduces his “logical computing machines” with the intention of providing an idealized description of a certain human activity, the tedious one of numerical computation, which until the advent of automatic computing machines was the occupation of many thousands of people in commerce, government, and research establishments. These people were referred to as computers. Turing prefaches his first description of a Turing machine with the words: “We may compare a man in the process of computing a...number to a machine.” The Turing ma-

---


machine is a model, idealized in certain respects, of a human computer. Ludwig Wittgenstein\textsuperscript{20} put this point in a striking way: “Turing’s ‘Machines’. These machines are humans who calculate” (ibid., §1096). It is a point that Turing was to emphasize, in various forms, again and again. For example: “A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine.”\textsuperscript{21} The electronic stored-program digital computers for which the universal Turing machine was a blueprint are, each of them, computationally equivalent to a Turing machine with a finite tape, and so they too are, in a sense, models of human beings engaged in computation. Turing chose to emphasize this when explaining the new electronic machines in a manner suitable for an audience of uninitiates: “The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer.”\textsuperscript{22} He makes the point a little more precisely in the technical document containing his preliminary design for the ACE:

> The class of problems capable of solution by the machine can be defined fairly specifically. They are [a subset of] those problems which can be solved by human clerical labour, working to fixed rules, and without understanding.\textsuperscript{23} (Turing went on to characterize the subset in terms of the amount of paper and time available to the human clerk.) It was presumably because he considered the point under discussion to be essential for understanding the nature of the new electronic machines that he chose to begin his \textit{Programmers’ Handbook for Manchester Electronic Computer}\textsuperscript{24} with this explanation:

> Electronic computers are intended to carry out any definite rule of thumb process which could have been done by a human operator working in a disciplined but unintelligent manner (ibid., p. 1).

> It was not some deficiency of imagination that led Turing to model his logical computing machines on what could be achieved by

\textsuperscript{21} “Intelligent Machinery,” p. 9.
\textsuperscript{24} University of Manchester Computing Laboratory, 1950.
a human computer. The purpose for which the Turing machine was invented demanded it. Turing introduced the Turing machine in the course of arguing that the *Entscheidungsproblem*, or decision problem, for the predicate calculus—posed by David Hilbert—is unsolvable. Here is Church's account of the *Entscheidungsproblem*:

> By the Entscheidungsproblem of a system of symbolic logic is here understood the problem to find an effective method by which, given any expression Q in the notation of the system, it can be determined whether or not Q is provable in the system (ibid., p. 41).

‘Effective’ and its synonym ‘mechanical’ are terms of art in mathematical logic. A mathematical method is termed ‘effective’ or ‘mechanical’ if and only if it can be set out in the form of a list of instructions able to be followed by an obedient human clerk—the computer—who works with paper and pencil, reliably but without insight or ingenuity, for as long as is necessary. The truth-table test is such a method for the propositional calculus. Turing showed by means of a two-stage argument that there can be no such method in the case of the predicate calculus. First, he proved formally that there is no Turing machine that can determine, in a finite number of steps, whether or not any given formula Q of the predicate calculus is a theorem of the predicate calculus. Second, he argued informally for the proposition that whenever there is an effective method for performing a mathematical task, then the method can be carried out by a Turing machine in some finite number of steps. These two stages jointly secure the result that there is no effective method for determining whether or not an arbitrary formula Q of the predicate calculus is a theorem of the calculus.

Notice that this result does not entail that there can be no *machine* for determining this (contrary to J. J. C. Smart and numerous other writers). The *Entscheidungsproblem* for the predicate calculus is the problem of finding a humanly executable procedure of a certain sort, and the fact that there is none is entirely consistent with the claim that some machine may nevertheless be able to decide arbitrary formulae of the calculus; all that follows is that such a machine, if it exists, cannot be mimicked by a human computer. Turing’s (and Church’s) discovery was that there are limits to what a human computer can achieve; for all that, their result is often portrayed as a discovery concerning the limitations of mechanisms in general.


The proposition that any effective method can be carried out by a Turing machine is known variously as Turing’s thesis and the Church-Turing thesis. Turing stated his thesis in numerous places, with varying degrees of rigor. The following formulation is one of the most accessible:

LCMs [logical computing machines] can do anything that could be described as “rule of thumb” or “purely mechanical.”

Turing adds

This is sufficiently well established that it is now agreed amongst logicians that “calculable by means of an LCM” is the correct accurate rendering of such phrases (ibid.).

Church proposed the (not quite) equivalent thesis that whenever there is an effective method for calculating the values of a function on the positive integers, then the function is recursive (not quite equivalent because Turing did not restrict attention to functions on the positive integers, mentioning also “computable functions of a real or computable variable, computable predicates, and so forth”). The term ‘Church-Turing thesis’ seems to have been introduced by Stephen Kleene (with a small flourish of bias in favor of Church):

So Turing’s and Church’s theses are equivalent. We shall usually refer to them both as Church’s thesis, or in connection with that one of its... versions which deals with “Turing machines” as the Church-Turing thesis (ibid., p. 232).

Essentially, then, the Church-Turing thesis says that no human computer, or machine that mimics a human computer, can out-compute a universal Turing machine. A further proposition, very different from this—namely, that a Turing machine can compute whatever can be computed by any machine—is nowadays sometimes referred to as the Church-Turing thesis or as Church’s thesis. For example, Paul Smolensky says:

connectionist models...may possibly even challenge the strong construal of Church's Thesis as the claim that the class of well-defined computations is exhausted by those of Turing machines (ibid., p. 3).

This loosening of established terminology is unfortunate, for neither Church nor Turing endorsed, or even formulated, this further proposition. There are numerous examples of this and other extended usages in the literature. The following are typical:

That there exists a most general formulation of machine and that it leads to a unique set of input-output functions has come to be called *Church's thesis*.32

Church-Turing thesis: If there is a well defined procedure for manipulating symbols, then a Turing machine can be designed to do the procedure.33

[I]t is difficult to see how any language that could actually be run on a physical computer could do more than Fortran can do. The idea that there is no such language is called Church's thesis.34

A typical way of stating Church's thesis is the following: In an ideal world the limit of computation is exactly captured by Turing computability.35

More distant still from anything that Church or Turing actually wrote:

The first aspect that we examine of Church's Thesis...[w]e can formulate, more precisely: The behaviour of any discrete physical system evolving according to local mechanical laws is recursive.36

I can now state the physical version of the Church-Turing principle: Every finitely realizable physical system can be perfectly simulated by

[Turing’s] universal model computing machine...This formulation is both better defined and more physical than Turing’s own way of expressing it.37

It is important to distinguish between Turing’s thesis and the stronger proposition that whatever functions (in the mathematical sense of ‘function’) can be generated by machines can be generated by a universal Turing machine.38 (To say that a function \( f \) can be generated by a machine \( m \) is simply to say that for each of the function’s arguments, \( x \), if \( x \) is presented to \( m \) as input, \( m \) will carry out some finite number of atomic processing steps at the end of which it produces the corresponding value of the function, \( f(x) \).

I shall call this stronger proposition the maximality thesis (thesis M) and shall use expressions such as ‘the Church-Turing thesis properly so called’ for the proposition that Church and Turing themselves endorsed.39

Maximality thesis: all functions that can be generated by machines (working on finite input in accordance with a finite program of instructions) are Turing-machine-computable.

Thesis M itself admits of two interpretations, according to whether the phrase ‘can be generated by a machine’ is taken in the this-worldly sense of ‘can be generated by a machine that conforms to the physical laws (if not to the resource constraints) of the actual world’, or in a sense that abstracts from the issue of whether or not the notional machine in question could exist in the actual world. The former version of thesis M is an empirical proposition whose truth value is unknown. The latter version of thesis M is known to be false. As I explain in the next section, there are notional machines that generate functions that no Turing machine can generate.

As previously remarked, the word ‘mechanical’, in technical usage, is tied to effectiveness, ‘mechanical’ and ‘effective’ being used interchangeably. (Gandy40 has outlined the history of this usage of the word “mechanical.”) Thus, statements like the following are to be found in the technical literature:

38 Robin Gandy is one of the few writers to draw such a distinction—“Church’s Thesis and Principles for Mechanisms,” in Jon Barwise, Jerome H. Keisler, and Kenneth Kunen, eds., The Kleene Symposium (Amsterdam: North-Holland, 1980), pp. 123-26.
39 Gandy (ibid.) uses the label ‘thesis M’ but not the term ‘maximality thesis’ (and his thesis M differs in certain respects from the maximality thesis stated here).
Turing proposed that a certain class of abstract machines could perform any "mechanical" computing procedure.  

Understood correctly, this remark attributes to Turing not thesis M but the Church-Turing thesis properly so called. This usage of 'mechanical' tends to obscure the possibility that there may be machines, or biological organs, that generate (or compute, in a broad sense) functions that cannot be computed by Turing machine. For the question 'Can a machine execute a procedure that is not mechanical?' may appear self-answering, yet this is precisely what is asked if thesis M is questioned.  

In the technical literature, the word 'computable' is sometimes tied by definition to effectiveness: a function is said to be computable if and only if there is an effective procedure for determining its values. The Church-Turing thesis then becomes:

Every computable function can be computed by a Turing machine.

Corollaries such as the following are sometimes offered:

...certain functions are uncomputable in an absolute sense: uncomputable even by [a Turing machine], and, therefore, uncomputable by any past, present, or future real machine.  

Of course, the decision to tie the term 'computable' and its cognates to the concept of effectiveness does not settle the truth value of thesis M; rather, those who abide by this terminological decision are prevented from describing any machine that falsifies thesis M as computing the function that it generates. Yet to a casual reader of the technical literature, statements like the one just quoted may appear to say more than they in fact do.  

Hilary Putnam, himself at one time a narrow mechanist, is one of the few writers on the philosophy of mind to question the maximality thesis:

42 Daniel Dennett appears simply to conflate the proposition that evolution is a "mindless, mechanical process" with the proposition that "evolution is an algorithmic process"—Darwin's Dangerous Idea: Evolution and the Meanings of Life (New York: Simon and Schuster, 1995), pp. 60, 75-76; see also pp. 48-60 passim. (As is customary, Dennett explicates the notion of an algorithm in terms of Turing machine activity: "Consider the set of all Turing machines—in other words, the set of all possible algorithms" (p. 437).) This conflation appears to underlie his view that "algorithmic processes...have created the entire biosphere, ourselves included" (p. 427).  
...materialists are committed to the view that a human being is—at least metaphorically—a machine. It is understandable that the notion of a Turing machine might be seen as just a way of making this materialist idea precise. Understandable, but hardly well thought out. The problem is the following: a "machine" in the sense of a physical system obeying the laws of Newtonian physics need not be a Turing machine (ibid., p. 4).

III. TURING’S OTHER MACHINES

In his doctoral thesis (which was supervised by Church), Turing introduced the idea of machines able to solve mathematical problems that cannot be solved by the “logical computing machines” of his 1936 paper. He described these as “a new kind of machine” and called them “O-machines” (ibid., p. 173).

An O-machine is in essence an ordinary Turing machine augmented with a black box that generates some function that cannot be generated by a Turing machine. Turing refers to the black box as an “oracle” (ibid.). As in the case of an ordinary Turing machine, the behavior of an O-machine is determined by a table of instructions (or program). The table provides an exhaustive specification of which fundamental processes the machine is to perform when it is in such-and-such state and has such-and-such symbol in its scanner. The tables of the two sorts of machine differ only in the following respect: an O-machine table may contain instructions of the form ‘TRANSFORM #*’. ‘#*’ refers to some particular string of symbols on the machine’s tape, the beginning of the string being marked by the presence on the tape of a reserved symbol ‘#’ and the end of the string being marked by a reserved symbol ‘*’. The instruction causes the portion of tape so marked to be presented to the black box. The symbols on this portion of tape constitute a specification of an argument of whatever function it is that the box generates (or of a series of n arguments in case the function is n-ary). The box replaces the symbols on the tape with a specification of the corresponding value of the function.

One way of conceptualizing an oracle—which need not reflect the box’s actual manner of functioning—is as a device accessing an infinite internal tape upon which there have been inscribed, in order, all the infinitely many arguments and values of whatever function it

46 A fuller account of O-machines may be found in my “Turing’s O-machines, Penrose, Searle, and the Brain.”
47 For ease of exposition, the present account departs from Turing’s own in various matters of detail.
is that the oracle generates. This device can produce any of the function’s values after only a finite search along the tape.

The transform operation performed by the oracle is, in Turing’s expression, one of the “fundamental processes” of the machine. He gave no indication of how this process might conceivably be carried out, saying only that an oracle works by “unspecified means” and that “we shall not go any further into the nature of [an] oracle” (ibid., pp. 172-73). In fact, notional machinery that discharges the task of an O-machine’s black box is not hard to concoct. Suppose for the sake of illustration that the function generated by the box is the Halting function. Turing introduced the Halting function in his 1936 paper and proved that no Turing machine generates it. The function is easily explained. Assume the Turing machines to be ordered in some way, so that we may speak of the first Turing machine in the ordering, the second, and so on (there are various standard ways of accomplishing this ordering). The arguments of the Halting function are simply 1, 2, 3,... The value of the function for any argument n is 1 if and only if the n\textsuperscript{th} Turing machine eventually comes to the end of its computation and halts, and is 0 if and only if the n\textsuperscript{th} machine runs forever (as would a Turing machine reprogrammed to produce in succession the digits of the decimal representation of π, for instance).

It is convenient to write ‘\(h_n\)’ to represent the value of the Halting function for argument n. \(h_n\) is always 0 or 1. Consider the following decimal specification of a number: 0\(\cdot\)h\(_1\)h\(_2\)h\(_3\)\(\ldots\); call this number ‘\(\tau\)’ (for Turing). (The first few digits of \(\tau\) might be 0\(\cdot\)000000011\ldots) Like π, \(\tau\) is a definite—irrational—number. The magnitude of some physical quantity might conceivably be exactly \(\tau\) units. Suppose that some mechanism A does store exactly \(\tau\) units of such a physical quantity, which for the sake of vividness one might call ‘charge’. Suppose further that a mechanism B can measure the quantity of ‘charge’ stored in A to any specified number of significant figures. A and B jointly discharge the task of an oracle that generates the Halting function. B determines \(h_n\) by measuring A’s

---

48 “Systems of Logic Based on Ordinals,” p. 173. In Turing’s original exposition, these new fundamental processes produce the values only of \(\pi_2\) functions. In the subsequent technical literature, the notion of an O-machine has been widened to include fundamental processes that produce values of any function on the integers that is not Turing-machine-computable. I employ this extended notion here.

49 Gregory Chaitin has defined a number \(\Omega\) that is analogous to, but not the same as, \(\tau\)—see, for example, his “Randomness in Arithmetic,” Scientific American, CCLIX (July 1988): 80-85.

50 Pace the intuitionists. Turing assumes a classical framework.
charge to an appropriate number of significant figures and outputting the \( n \)th digit of the result.

An O-machine consisting of a Turing machine and the oracle just described is a machine in the sense that its behavior is the product of the nature and arrangement of its material parts. The core claim of historical mechanism—namely, that the mind is wholly explicable within the resources of some monistic, materialist theory in a manner analogous to that in which the behavior of artifacts is accounted for in terms of the organization and functions of their parts—is evidently consistent with the hypothesis that the mind is an O-machine.

### IV. ARGUMENTS THAT NARROW MECHANISM EXHAUSTS MECHANISM

I have elsewhere\(^5\) coined the term *Church-Turing fallacy* for a persistent error that is to be encountered in modern writing concerning mechanism. This fallacy takes a number of distinct but closely related forms, several of which are described below. Far from being confined to writers of one particular philosophical stripe, the Church-Turing fallacy is to be found in the work of thinkers of very different persuasions. I shall illustrate the widespread nature of the fallacy by considering arguments put forward by a diverse field of modern thinkers (including Paul Churchland, Patricia Churchland, Daniel Dennett, Hubert Dreyfus, Jerry Fodor, Allen Newell, and John Searle). It is important that the fallacy be exposed and eradicated. Anyone in its grip will think that narrow mechanism exhausts mechanism: to them, conceptual space will seem to contain no room for mechanical models of the mind that are not equivalent, in the appropriate sense, to one or another class of Turing machines. Propagation of the fallacy by leading theoreticians has assisted in blotting from view a potentially rich field of possible models of human cognition.

In essence, to commit the Church-Turing fallacy is to believe that the Church-Turing thesis, or some formal or semiformal result established by Turing or Church, secures the proposition that, if mechanism is true, the functions generated by Turing machines provide sufficient mathematical resources for a complete account of human cognition. One form of the fallacy concerns specifically the notion of a Turing machine simulating, or mimicking, the human cognitive apparatus.\(^5\) Someone commits this form of the fal-

---

\(^{5}\) "Turing’s O-machines, Penrose, Searle, and the Brain."

\(^{5}\) I take the claim that some entity \( e \) can be simulated by Turing machine to mean that some Turing machine can pair any given descriptions of the stimuli impinging on \( e \) with either exact descriptions of \( e \)’s consequent behavior or descriptions that are accurate to any prespecified number of significant figures.
lacy (the simulation fallacy) by believing that the Church-Turing thesis—or, again, some formal or semiformal result established by Turing or Church—entails that, if mechanism is true, then a universal Turing machine can simulate the mind. The equivalence fallacy involves mistaking the strong evidence for the Church-Turing thesis properly so called for evidence supporting thesis M, and so passing from various logico-mathematical considerations to the view that narrow mechanism exhausts mechanism.

The Church-Turing fallacy has led to some remarkable claims in the foundations of psychology. For example, one frequently encounters the view that psychology must be capable of being expressed in computational terms and so ultimately in terms of the Turing machine. Yet it is certainly possible that psychology will find need to employ mathematical functions that cannot be generated by Turing machine.

Fodor\(^{53}\) writes:

> Although the elementary operations of the Turing machine are restricted, iterations of the operations enable the machine to carry out any well-defined computation on discrete symbols.... If a mental process can be functionally defined as an operation on symbols, there is a Turing machine capable of carrying out the computation.... The “black boxes” that are common in flow charts drawn by psychologists often serve to indicate postulated mental processes for which Turing reductions are wanting. Even so, the possibility in principle of such reductions serves as a methodological constraint on psychological theorizing by determining what functional definitions are to be allowed (ibid., p. 130).

The claim made in the second sentence of this quotation is false.\(^{54}\) Each O-machine carries out some well-defined operation on discrete symbols. As in the case of an ordinary Turing machine, an O-machine generates discrete symbolic output from discrete symbolic input, possibly via intermediate structures of discrete symbols, by means of the step-by-step procedure specified in its machine table. Fodor’s overarching view that mental processes consist of operations on discrete symbols does not entail the narrow mechanist view of psychology which he advocates.


\(^{54}\) As is Dreyfus’s similar claim that “any process which can be formalized so that it can be represented as a series of instructions for the manipulation of discrete elements can, at least in principle, be reproduced by [a universal Turing machine]”—*What Computers Still Can’t Do: A Critique of Artificial Reason* (Cambridge: MIT, 1992), p. 72.
The Churchlands\textsuperscript{55} hold that Turing’s results entail something remarkable, namely that a standard digital computer, given only the right program, a large enough memory and sufficient time, can compute any rule-governed input-output function. That is, it can display any systematic pattern of responses to the environment whatsoever \textit{(ibid.}, p. 26).

If this were true, then the view that psychology must be capable of being expressed in standard computational terms would be secure. But Turing had no result entailing this. What he did have was a result entailing the exact opposite. His theorem that no Turing machine can generate the Halting function entails that there are possible patterns of responses to the environment, perfectly systematic patterns, which no Turing machine can display. The Halting function is a mathematical characterization of just such a pattern.

Searle argues for a narrow mechanist account of mind directly from Church’s thesis:

Can the operations of the brain be simulated on a digital computer?... [G]iven Church’s thesis that anything that can be given a precise enough characterization as a set of steps can be simulated on a digital computer, it follows trivially that the question has an affirmative answer.\textsuperscript{56}

If the question [‘Is consciousness computable?’] asks ‘Is there some level of description at which conscious processes and their correlated brain processes can be simulated [by a Turing machine]?’ the answer is trivially yes. Anything that can be described as a precise series of steps can be simulated [by a Turing machine].\textsuperscript{57}

Of course, Church’s thesis properly so called does not say that anything that can be described as a precise series of steps can be simulated by Turing machine. The behavior of an O-machine as it follows the instructions in its table can certainly be characterized as consisting of a set of steps, each step consisting of the execution of one of the machine’s fundamental processes; yet if the brain is an O-machine, it is false that its operations can (in their entirety) be simulated by a Turing machine.

As I have already said, the error that Searle commits here—that of holding that Church’s and Turing’s results somehow entail that the


brain can be simulated by a Turing machine—is a common one. The entry on Turing in Samuel Guttenplan’s recent *A Companion to the Philosophy of Mind* contains the following claims: “we can depend on there being a Turing machine that captures the functional relations of the brain,” for so long as “these relations between input and output are functionally well-behaved enough to be describable by... mathematical relationships...we know that some specific version of a Turing machine will be able to mimic them” (*ibid.*, p. 595). Even Dreyfus, in the course of criticizing the view that “man is a Turing machine,” succumbs to the belief that it is a “fundamental truth that every form of ‘information processing’ (even those which in practice can only be carried out on an ‘analogue computer’) must in principle be simulable on a [Turing machine]” (*op. cit.*, p. 195). Similarly, Philip Johnson-Laird and the Churchlands argue:

If you assume that [consciousness] is scientifically explicable...[and] granted that the [Church-Turing] thesis is correct, then the final dichotomy rests on Craik’s functionalism. If you believe [functionalism] to be false...then presumably you hold that consciousness could be modelled in a computer program in the same way that, say, the weather can be modelled... If you accept functionalism, however, then you should believe that consciousness is a computational process.

Church’s Thesis says that whatever is computable is Turing computable. Assuming, with some safety, that what the mind-brain does is computable, then it can in principle be simulated by a computer.

As previously mentioned, the Churchlands believe, erroneously, that Turing’s “results entail...that a standard digital computer...can...display any systematic pattern of responses to the environment whatsoever.” This no doubt explains why they think they can assume “with some safety” that what the mind-brain does is computable, for on their understanding of matters, this is to assume only that the mind-brain exhibits a systematic pattern of responses, or is characterized by a “rule-governed” input-output function.

---

Each of the authors quoted appears to be assuming the truth of a close cousin of thesis M, which I shall call:

Thesis S: any process that can be given a systematic mathematical description (or a “precise enough characterization as a set of steps,” or that is scientifically describable or scientifically explicable) can be simulated by a Turing machine.

As with thesis M, neither the Church-Turing thesis properly so called nor any result of Turing or Church entails thesis S. This is so even when thesis S is taken as concerning only processes that conform to the physics of the real world. Taken in a broader sense that abstracts from the issue of whether or not the processes in question could exist in the actual world, thesis S is known to make a false claim (the processing carried out by an O-machine suffices to counterexample it). The view that the mind is scientifically explicable in no way entails narrow mechanism. For all we presently know, a completed neuroscience may present the mind-brain as a machine that generates functions which no Turing machine can generate.

Paramount among the evidence for the Church-Turing thesis properly so called is the fact that all attempts to give an exact analysis of the intuitive notion of an effective mathematical method have turned out to be equivalent in extension. Because of the prima facie diversity of the various analyses, their equivalence is generally considered extremely strong evidence for the Church-Turing thesis properly so called.62 (Apart from Turing’s analysis, and Church’s analyses in terms of lambda-definability and recursiveness, there are analyses, for example, in terms of register machines, Post’s canonical and normal systems, combinatory definability, Markov algorithms, and Gödel’s notion of reckonability.) In the narrow mechanist literature, the equivalence of these diverse analyses is commonly taken to be evidence for thesis M. This is nothing more than a confusion—the equivalence fallacy. The analyses under discussion are of the notion of an effective method, not of the notion of a machine-generable function; the equivalence of the analyses bears only on the issue of the extent of the former notion and indicates nothing concerning the extent of the latter.

Newell and Simon63 encapsulate a narrow mechanist view of mind in their famous physical symbol system hypothesis:

---

62 See, for example, Kleene, *Introduction to Metamathematics* (Amsterdam: North-Holland, 1952), chapters 12, 13.

A physical symbol system has the necessary and sufficient means for general intelligent action.... Any physical symbol system of sufficient size can be organized further to exhibit general intelligence (*ibid.*, p. 116).

By the phrases ‘general intelligent action’ and ‘general intelligence’, Newell and Simon “wish to indicate the same scope of intelligence as we see in human action” (*ibid.*). A physical symbol system is a universal Turing machine, or any equivalent system, situated in the physical (as opposed to the conceptual) world. (The tape of the machine is accordingly finite; Newell specifies that the storage capacity of the tape, or equivalent, be unlimited in the practical sense of finite yet not small enough to “force concern.”) The physical symbol system hypothesis is the foundation stone of the particular brand of narrow mechanism that has been the dominant research paradigm in cognitive science since mid-twentieth century.

Newell thinks it is easily established that a physical symbol system can be organized to exhibit general intelligence:

A [physical symbol] system always contains the potential for being any other system if so instructed. Thus, a [physical symbol] system can become a generally intelligent system (*ibid.*, p. 170).

Is the premise true? A physical symbol system, being a *universal* Turing machine situated in the real world, can, if suitably instructed, simulate (or, metaphorically, become) any other physical symbol system (modulo some fine print concerning storage capacity). If this is what the premise means, then it is true; but if taken literally, the premise is false, for systems can be specified which no physical symbol system can simulate (for example, an O-machine). If the premise is interpreted in the former manner, however, the argument is simply a non sequitur. Only to one who believes, as Newell does, that “the notion of machine or determinate physical mechanism” is “formalized” by the notion of a Turing machine (*ibid.*) will the argument appear deductively valid. His defense of his view that the universal Turing machine exhausts the possibilities of mechanism involves an example of the equivalence fallacy:

...an important chapter in the theory of computing...has shown that all attempts to...formulate...general notions of mechanism...lead to classes of machines that are equivalent in that they encompass in toto exactly the same set of input-output functions. In effect, there is a single large frog pond of functions no matter what species of frogs (types of ma-

---

64 “Physical Symbol Systems,” p. 161. (See also Turing’s “Intelligent Machinery,” p. 15.)
chines) is used.... A large zoo of different formulations of maximal classes of machines is known by now—Turing machines, recursive functions, Post canonical systems, Markov algorithms... (op. cit., p. 150).

Dennett\(^{65}\) has fielded the following argument from “Church’s Thesis,” which he states in the form “anything computable is Turing-machine computable”:

...a non-question-begging psychology will be a psychology that makes no ultimate appeals to unexplained intelligence, and that condition can be reformulated as the condition that whatever functional parts a psychology breaks its subjects into, the smallest, or most fundamental, or least sophisticated parts must not be supposed to perform tasks or follow procedures requiring intelligence. That condition in turn is surely strong enough to ensure that any procedure admissible as an “ultimate” procedure in a psychological theory falls well within the intuitive boundaries of the “computable” or “effective” as these terms are...used in Church’s Thesis...[A]ny psychology that stipulated atomic tasks that were “too difficult” to fall under Church’s Thesis would be a theory with undischarged homunculi (ibid., p. 83).

The conclusion Dennett draws from the argument is that “the supposition that there might be a non-question-begging non-mechanistic psychology gets you nothing” (ibid., and see also p. 112); and, clearly, if the argument worked, it would also show that there cannot be a non-question-begging mechanistic psychology postulating atomic processes that are not Turing-machine computable. The transform operation discussed in section \(\text{III}\) serves to highlight the error in the argument: this operation is an example of an atomic task “too difficult to fall under Church’s thesis,” yet the account of its implementation in terms of mechanisms \(A\) and \(B\) is entirely mechanical and makes no “appeal to unexplained intelligence.” At bottom, what has led Dennett astray is his belief that Church’s thesis tells us that every

...task for which there is a clear recipe composed of simple steps can be performed by a very simple computer, a universal Turing machine, the universal recipe-follower (ibid., p. xviii).

It is worth remarking that a number of well-known arguments against computational functionalism cannot be brought to bear when it is wide rather than narrow mechanism which informs the functionalist’s ac-

count. Examples are Searle’s Chinese room argument and Ned Block’s homunculus-head argument. Block too eagerly offers the latter as an embarrassment “for all versions of functionalism” (ibid., p. 277). His brain-of-slaves scenario is well-known: a billion Chinese clerks working effectively are brought into functional equivalence with your mind, each clerk implementing a single line of an “adequate machine table that describes you” (ibid., p. 278). Here, Block tacitly assumes narrow mechanism. His argument is powerless against the richer functionalism countenanced by wide mechanists, which allows that the machine table describing your mind may be of such a nature that—like the table of an O-machine—it cannot be implemented by human clerks working effectively (see section II).

V. SOME POTENTIALLY MISLEADING FEATURES OF TURING’S PRESENTATION

More than anyone else, Turing is to be thanked for uniting historical mechanism with modern mathematics. He enriched mechanism with an abstract theory of (information-processing) machines, presenting us with an indefinitely ascending hierarchy of possible machines, of which the Turing machines form the lowest level. His work posed a new question: If the mind is a machine, where in the hierarchy does it lie? Yet Turing has been widely misinterpreted. He is popularly believed to have had some limitative result concerning the extent of the class of possible machines; and, as we have seen, expressions of the view that mechanism entails narrow mechanism are generally accompanied by a nod toward Turing (or Church).

Precisely how these misunderstandings of Turing’s work arose is a matter of little consequence. Part of the explanation, perhaps, is the presence of various minor features of Turing’s mode of presentation which can easily mislead. One of these has already been mentioned: in Turing’s early papers, the words ‘computer’, ‘computable’, and ‘computation’ are employed not in their modern sense as pertaining to machines but as pertaining to human calculators. So, for example, when Turing maintains that every number or function “which would naturally be regarded as computable” can be generated by a Turing machine, he is advancing only the Church-Turing thesis properly so called, and not a version of thesis M. Similarly, Tur-

66 I discuss Searle’s argument in “Turing’s O-machines, Penrose, Searle, and the Brain.”
ing’s use of the phrase ‘universal computing machine’ implies nothing more than that the machine so denoted can carry out the work of any human computer.

When Turing uses the word ‘machine’, he often means not machine-in-general but logical computing machine or, simply, effective method. At one point, he explicitly draws attention to this usage:

The expression “machine process” of course means one which could be carried out by the type of machine I was considering [in the 1936 paper].

Thus when, a few pages later, he asserts that “machine processes and rule of thumb processes are synonymous” (ibid., p. 112), he is to be understood not as advocating narrow mechanism but as advancing the Church-Turing thesis properly so called (and its converse). Likewise, when he says that an oracle “cannot be a machine,” he means only that an oracle cannot be a Turing machine (as he himself had proved); he remarks in the very next sentence that an O-machine is “a new kind of machine.” Especially liable to mislead are statements like the following, which a casual reader might easily mistake for the claim that the universal Turing machine provides a “general notion of mechanism” (Newell’s phrase):

The importance of the universal machine is clear. We do not need to have an infinity of different machines doing different jobs. A single one will suffice. The engineering problem of producing various machines for various jobs is replaced by the office work of “programming” the universal machine to do these jobs.

In context, it is clear that these remarks of Turing’s concern machines equivalent to logical computing machines.

Turing introduces the term ‘discrete-state machine’ for those machines whose possible states (configurations) form a discrete set: these machines “move by sudden jumps or clicks from one quite definite state to another.” (Each Turing machine is a discrete-state machine, of course.) He opposes the discrete-state machines to “continuous machinery,” the states of which “form a continuous manifold, and the behavior of the machine is described by a curve

---

70 “Systems of Logic Based on Ordinals,” p. 173.
71 “Intelligent Machinery,” p. 7.
It is sometimes said that any discrete-state machine can be simulated by a universal Turing machine. Turing himself may appear to be endorsing this claim (which is a restricted form of thesis M). He says: a “digital computer could mimic the behavior of any discrete-state machine.” The surrounding discussion (ibid., pp. 440-41) makes it clear that he intends this statement to apply only in the case of those discrete-state machines which have “a finite number of possible states” (that is, a finite number of possible configurations) (ibid., p. 440). He points out that when this condition is satisfied, the behavior of the machine can be described exhaustively by a finite table of the sort nowadays commonly called a look-up table (ibid.); it is on the basis of being “[g]iven the table corresponding to a discrete-state machine” that a digital computer could mimic the latter (ibid., p. 441).

An example of a discrete-state machine whose behavior cannot be calculated by a universal Turing machine is a digital computer with an infinite-capacity store and what Turing calls a “random element” (ibid., pp. 438-39). He refers to computing machines with a random element as “partially random machines.”

VI. TURING’S VIEW: THE MIND AS PARTIALLY RANDOM MACHINE

A device that outputs a genuinely random and unboundedly long sequence of integers is a form of oracle (section III). As suggested previously, the device may be conceptualized as one accessing a tape upon which an infinite random sequence of integers has been inscribed. Turing explains that a discrete-state machine to which

---

73 “Intelligent Machinery,” p. 5. Turing concludes on the basis of neurophysiological evidence that the mind, if mechanical, is “not...a discrete-state machine” but a continuous machine—“Computing Machinery and Intelligence,” pp. 451, 455. He sees no theoretical significance in this, however: “brains very nearly fall into this class [discrete-state machines], and there seems every reason to believe that they could have been made to fall genuinely into it, without any change in their essential properties”—“Intelligent Machinery,” p. 6.

74 Hodges, for example, asserts this, attributing the view to Turing—Alan Turing: The Enigma, p. xvii, and Turing (London: Phoenix, 1997), pp. 34-36, 39.

75 “Computing Machinery and Intelligence,” p. 441.

76 Hodges (Turing, p. 34) quotes extensively from the relevant pages of “Computing Machinery and Intelligence,” but fails to include the crucial words “discrete state machines...can be described by such tables provided they have only a finite number of possible states”—“Computing Machinery and Intelligence,” p. 440 (my italics).

77 “Intelligent Machinery,” p. 9.

78 The arguments of the function generated by such an oracle are 1st call, 2nd call, 3rd call,...(or simply 1, 2, 3...) and the first value of the function is the number that the oracle produces in response to the first call, and so on.
such a device is attached may be set up so as to choose between two paths of action by calling to the device for a number and following one path if, say, the number is even and the other if it is odd (ibid.). Except in the case where the number of possible configurations of the machine is finite, a partially random discrete-state machine cannot be simulated by a Turing machine, for as Church pointed out in 1939, if a sequence of integers $a_1, a_2, ..., a_n, ...$ is random, then there is no function $f(n) = a_n$ that is calculable by Turing machine.79

Turing often mentions this idea of partial randomness. For example, in a paper on machine intelligence he wrote:

[O]ne feature that I would like to suggest should be incorporated in the machines...is a “random element”. This would result in the behavior of the machine not being by any means completely determined by the experiences to which it was subjected.80

Much interested in the issue of free will, Turing seems to have believed that the mind is a partially random machine. We have the word of one of Turing’s closest associates, mathematician Max Newman, that Turing “had a deep-seated conviction that the real brain has a ‘roulette wheel’ somewhere in it.”81 So far as is known, Turing’s only surviving discussion of these matters occurs in the type-

---

80 “Intelligent Machinery, A Heretical Theory,” in my “A Lecture and Two Radio Broadcasts on Machine Intelligence by Alan Turing,” in K. Furukawa, D. Michie, and S. Muggleton, eds., Machine Intelligence 15 (New York: Oxford, 1999): 445-76, here p. 461. (The date of writing of this paper is not known with certainty. It was presented in Manchester to a meeting of the “51 Society,” the name being taken from the year.)
81 Newman in interview with Christopher Evans—“The Pioneers of Computing: An Oral History of Computing” (London: Science Museum, 1976). Newman played an important part in Turing’s intellectual life over many years. It was Newman who, in a lecture in Cambridge in 1935, introduced Turing to the concept that led directly to the Turing machine: Newman defined a constructive process as one that a machine can carry out (ibid.). During the war, Newman and Turing both worked at the Government Code and Cypher School, Bletchley Park, where the two cooperated closely. It was Newman who initiated the electronic decryption project that culminated in the construction of Colossus, the first large-scale electronic digital computing machine (designed by the engineer Thomas H. Flowers). At the end of the war, Newman established the Royal Society Computing Machine Laboratory at the University of Manchester, where he introduced the engineers F.C. Williams and T. Kilburn to Turing’s idea of a universal computing machine; and, under Newman’s guidance, Williams and Kilburn built the first stored-program electronic digital computer (see the introduction to “A Lecture and Two Radio Broadcasts on Machine Intelligence by Alan Turing,” pp. 455-57). In 1948, Newman appointed Turing as Deputy Director of the Computing Machine Laboratory (there being no director), and Turing remained at Manchester until his death in 1954.
script of a lecture that he gave in 1951 on BBC radio, entitled “Can Digital Computers Think?” In the course of his discussion, Turing considers the claim that if “some particular machine can be described as a brain we have only to programme our digital computer to imitate it and it will also be a brain.” He remarks that this “can quite reasonably be challenged,” pointing out that there is a difficulty if the behavior of the machine is not “predictable by calculation,” and he draws attention to Eddington’s view that “no such prediction is even theoretically possible” on account of “the indeterminacy principle in quantum mechanics.”

Turing’s overarching aim in the lecture is to answer the question posed by his title, and his strategy is to argue for the proposition that “[i]f any machine can appropriately be described as a brain, then any digital computer can be so described.” This proposition is consistent, he explains, with the possibility that the brain is the seat of free will:

To behave like a brain seems to involve free will, but the behaviour of a digital computer, when it has been programmed, is completely determined.... It is certain that a machine which is to imitate a brain must appear to behave as if it had free will, and it may well be asked how this is to be achieved. One possibility is to make its behaviour depend on something like a roulette wheel or a supply of radium.... It is, however, not really even necessary to do this. It is not difficult to design machines whose behaviour appears quite random to anyone who does not know the details of their construction (op. cit., p. 464).

He calls machines of the latter sort “apparently partially random”; an example is a Turing machine in which “the digits of the number $\pi$ [are] used to determine the choices.” Apparently partially random machines imitate partially random machines. If the brain is a partially random machine, an appropriately programed digital computer may nevertheless give a convincing imitation of a brain. The appearance that this deterministic machine gives of possessing free will is

---

82 Hitherto unpublished, the text is now available in my “A Lecture and Two Radio Broadcasts on Machine Intelligence by Alan Turing.” In an early essay entitled “Nature of Spirit,” possibly dating from Turing’s undergraduate days, he wrote: “the theory which held that as eclipses etc. are predestined so were all our actions breaks down...We have a will which is able to determine the action of the atoms probably in a small portion of the brain, or possibly all over it.”

83 “Intelligent Machinery,” p. 9. Turing devised a program which caused the Manchester computer to behave in an apparently partially random manner. When given a number the program would respond with another. Turing says “I would defy anyone to learn from these replies sufficient about the programme to be able to predict any replies to untried values”—“Computing Machinery and Intelligence,” p. 453.
“mere sham”; but free will aside, it is “not altogether unreasonable” to describe a machine that “imitate[s] a brain” as itself being a brain. (As is well known, Turing advocates imitation as the basis of a test that “[y]ou might call...a test to see whether the machine thinks.”

In the course of the past four decades, there have been a number of detailed suggestions for notional machines which, although completely deterministic, generate functions that cannot be generated by a universal Turing machine. These suggestions are of considerable interest to wide mechanists. They abundantly falsify the more expansive version of thesis M which abstracts from the issue of existence in the actual world. It remains an open empirical question whether or not the this-worldly version of thesis M is likewise false, and in particular whether the thesis is falsified by any deterministic mechanism. It is uncertain what Turing himself might have thought about the latter issue; if he ever discussed it, nothing appears to have survived.

The proposition, so important to Turing, that, “[i]f any machine can appropriately be described as a brain, then any digital computer can be so described” is consistent with the view that the brain is computationally equivalent to, say, an O-machine whose oracle produces the values of the Halting function. For the foregoing argument of Turing’s can readily be modified to cover this case: an appropriately programed Turing machine will appear to an observer “who does not know the details of [its] construction” to behave in a perfectly brain-like fashion, and a machine that successfully imitates a brain can reasonably be said to be a brain. As Turing remarks elsewhere, the Turing machine will produce “an occasional wrong result,” but this will hardly mark out the Turing machine from the brain.

---

84 The quotation is from “Can Automatic Calculating Machines Be Said To Think?” in my “A Lecture and Two Radio Broadcasts on Machine Intelligence by Alan Turing,” p. 466.

85 My and Sylvan’s “Beyond the Universal Turing Machine” is a survey; see also the references in my “Turing’s O-machines, Penrose, Searle, and the Brain.”

86 Turing does endorse the thesis that results when the words ‘any machine’ in the statement of thesis M are replaced by ‘any calculating machine’, saying “[a] digital computer is a universal machine in the sense that it can be made to replace... any rival design of calculating machine”—“Can Digital Computers Think?” p. 462. If he were pressed to make it clear exactly what is meant by ‘calculating machine’, he would perhaps offer paradigm examples, as on pp. 5-6 of his earlier “Intelligent Machinery”: the Brunsviga and the NCR (popular desk calculating machines), the ENIAC (the electronic numerical integrator and computer), and so on. Or perhaps he would say, with greater generality, that a calculating machine is any machine that apes a human mathematician working with pencil and paper in accordance with a “rule of thumb” procedure. As previously remarked, it was in that manner that he explained the idea of an electronic computing machine in the opening paragraph of his Programmers’ Handbook.

Finally, what of the point that the behavior of any discrete-state machine with only a finite number of possible configurations can be simulated by a universal Turing machine? Does this undermine wide mechanism or provide a reason for saying that the historical mechanism of Descartes, Hobbes, La Mettrie, et alia carried an implicit commitment to narrow mechanism? Not at all. For one thing, the mind may be some form of continuous machine not simulable by Turing machine. But let us suppose for argument’s sake that the mind is a discrete-state machine, and that, being situated in a world of bounded resources (time, energy, memory, and so on), the number of possible configurations that this machine can adopt is finite. In this case, each mind is simulable by a Turing machine equipped with a suitable look-up table, even if the table can be constructed only post hoc; but this provides no support for narrow mechanism. The crucial issue here is whether our cognitive architecture, abstracted from resource constraints, is best understood as being a generator of (one or more) Turing-machine-uncomputable functions, and the fact that the mind is simulable by Turing machine when certain resource constraints are operative says nothing either way. The wide mechanist stands firm on the claims that the empirical issue of how best to model the central mechanisms underlying cognitive performance is still pretty much completely open, and that there are no compelling reasons to believe that the model ultimately adopted will be selected from the narrow mechanist’s artificially constrained space of models.

B. JACk COPELAND

University of Canterbury