Notre Dame Journal of Formal Logic
Volume 28, Number 4, October 1987

Church's Thesis and Cognitive Science

R. J. NELSON*

1 Introduction Although Church's Thesis (CT) has been central to the theory of effective decidability for fifty years, the question of its epistemological status is still an open one. My own view, which is prompted by a naturalistic attitude toward such questions in mathematics as elsewhere, is that the thesis is an empirical statement of cognitive science, which is open to confirmation, amendment, or discard, and which, on the current evidence, appears to be true. Naturalism, although not to be identified with any of the classical schools of philosophy of mathematics, including the historicism of Webb [65], is hardly new; and if pushed to its Quinean limits would have to insist that mathematical epistemology is in principle a part of psychology. However in this paper I wish only to advocate the limited metathesis that CT is empirical, yet mathematical. I leave defense of the wider claim to others.

This interpretation of CT is quite naturally suggested by one of the standard arguments for a mechanist theory of the mind, which CT supports. That argument, which I will review in more detail below, is this: Human cognitive processes are effective; by the thesis it follows they are recursive relations. This justifies defining the mind *qua* cognizing as a system of recursive rules, i.e., as a *machine* of some kind. Considerations in defense of mechanism thus tend to support CT much as empirical findings and low level laws in the physical and biological sciences tend to confirm or disconfirm relevant hypothetical generalizations. In my opinion much of this support is likely to come from cognitive science.[1] Likewise, putative refutations of mechanism threaten CT and are likely to stem from empirical findings.

*I have benefited from many discussions of Church's Thesis with William Thomas and Judson Webb. I do not want to suggest, however, that either one would wholly agree with the position expressed in this article. I also wish to thank David Helman and Stewart Shapiro for their helpful comments

Received July 7, 1986; revised August 6, 1986

However, the empirical content of CT can be appreciated independently of any mechanist considerations by logical analysis of the meanings of 'effective', 'computable', etc. When this has been done it is relatively easy to defeat many arguments against CT since they deal with it on too superficial a level. So a secondary thesis of this paper is that failure to analyze CT down to its empirical roots has generated misunderstanding of its full significance.

Prima facie this viewpoint seems wrong. Church himself ([12], p. 100) refers to his thesis as a *definition*: he defines

(1) the notion . . . of an *effectively calculable function* of the positive [or non-negative] integers by identifying it with the notion of a recursive function of positive integers. . . .

Construed thus as a definition, CT presumably has no synthetic content; the notions in it are identical, by convention.[2]

Moreover it evidently expresses a relation between mathematical objects and is not explicitly about mental entities at all, except for the intuitionist. For the latter, mathematical objects are indeed conceptual; but I doubt any intuitionist would grant that cognitive science has the slightest relevance to the mathematical character or significance of certain cognitions (intuitions of one-twoness, construction of choice sequences, e.g.) in which mathematical reasonings and concepts are intuitionistically rooted. At any rate CT is widely recognized as a proposition of mathematics, not of ordinary empirical psychology.

Again, many critics have seriously questioned the truth or adequacy of CT. For instance, fairly good arguments can be set out suggesting that some effectively computable functions are not recursive or conversely, or maintaining that the arguments both a priori and empirical meant to uphold the thesis are without respectable standing measured by the usual canons of mathematical and scientific inquiry (see Thomas [58]). Post [45] and Rogers [49] following him have used CT as an informal rule of inference: "from 'A is a generated set' to infer 'A is recursively enumerable'" or "from 'f is effectively computable' to infer 'f is recursive'". Considered so, CT is neither a true nor false statement, but a kind of directive.

So there is an initial presumption, shared by many, that CT is neither empirical nor true.

Arguing to the contrary, in Section 2 I will defend an empirical interpretation of CT based on an analysis that centers on the meaning of 'effective'. In Section 3 I review a few arguments against CT and show that they fail on points that can be settled only empirically. These include intuitionist arguments, and objections to CT that have been raised by Kalmar, and objections to mechanism of Lucas and Benacerraf based on Gödel theorems that indirectly threaten CT. Section 4 reviews certain Turing machine models of sundry cognitive processes and shows that the models satisfy various empirical adequacy conditions. I like to think of this work as a continuation of Turing's own defense of his version of CT, in which he showed that his machines model the primitive operations performed by an idealized human computist following algorithms ([60], p. 136f). The upshot of Section 4 is that such arguments as well as many developments

in artificial intelligence and cognitive psychology provide strong empirical support for CT.

2 Analysis of Church's thesis The question whether or not CT is empirical (cf. [62], pp. 87–89) resurrects a host of questions that might better remain entombed with logical empiricism. As noted, Church himself refers to CT as a "definition". But this can hardly be meant in the sense of "abbreviational" or "reductive" definition; it is a convention or a proposal, as Rogers ([49], p. 20) terms it. If all definitions are analytic, the thesis is certainly not a definition. For it is not an instance of a tautology—its negation is not self-contradictory; and it is not analytic in the sense that it holds in virtue of the meanings of the terms occurring in it ([64], p. 453).[3]

For familiar reasons it is better to think of CT as an *explication* (Carnap, [6], p. 5ff.). The extension of 'effectively computable' is vague and the explicans 'recursive' sharpens it, which is the main reason CT is important and was introduced by Church in the first place. On the other hand, the concept of analytic statement, if legitimacy still attaches to it outside of pure logic, certainly presupposes precision in the expressions to which it applies. Minimal clarity requires, therefore, making a distinction between definitions that relate two antecedently precise concepts (such as the real number $\frac{1}{3}$ and the set of rationals less than $\frac{1}{3}$) from those that do not (such as effective computability and recursiveness).

Assuming it is agreed that CT is an explication of mathematical terms, how can it be plausibly construed as empirical? In answer, let us take note of five grades of a prioricity of definitions in the domain of mathematics.

First, there are strictly notational definitions such as "let 'R.E. set' abbreviate 'recursively enumerable set'". Such definitions might be the only purely analytic definitions there are. They are a priori by *fiat* (cf. [47], p. 26).

Second, there are nominal definitions such as "'$a \leq b$' for 'there is a c such that $a + c = b$', a, b, c, natural numbers". This is analytic a priori if any mathematical statement is; anyone who understands the meanings of the definiens and definiendum would accept it as necessarily true.

Third, there are reductive definitions such as "'$\frac{1}{3}_{\text{real}}$' for 'the set of fractions less than $\frac{1}{3}_{\text{rational}}$, which are analytic in the sense that the definiens is interchangeable *salva veritate* with the definiendum *mutatis mutandis* in all contexts in real analysis where '$\frac{1}{3}_{\text{real}}$' occurs, and insofar forth is a priori.

Fourth, there are explications such as the definition of 'continuous real function' which analyze the intuitive idea of a continuous (functional) curve. It cannot be said that the explication is a priori in the third sense above inasmuch as 'continuous real function' is not synonymous with 'continuous curve [in the intuitive sense]' throughout the latter's vague extension ([47], p. 25). Such definitions might be said to be empirical in point of the manner of justifying them. They are 'fruitful', 'simple', 'lend coherence to the theory', 'hold in all cases examined', and insofar forth are "true" or mathematically acceptable. They are justified on nondeductive, heuristic, pragmatic, i.e., *dialectical* grounds. In a weak sense they are *empirical* [49], not because the *content* is material but

because they are warranted by reasonings similar to those one would ordinarily associate with choices of definitions and hypotheses in natural science.

Fifth, and finally, there are explications warranted empirically in the following strong sense: their acceptability stems in part from bearing verifiable material content. Explication veers over into explanation. Examples don't come to mind easily, but CT is one. That CT has empirical content is especially clear in Turing's version, as we shall see. Another example is the definition of 'scale' in measurement theory [55] which explicates the notion of scale by way of an empirical assumption of the existence of homomorphisms of natural relational structures, such as the ordered pair consisting of a set of lumps of metal and the relation *heavier than* to a structure of the nonnegative reals and *greater than.*

The point of making these distinctions is epistemological. CT is not true a priori, yet it is mathematical from the point of view of the practicing mathematician, and is a priori in a methodological sense. As every one knows, it is fundamental to decidability theory in formal logic and nearly indispensable for practical reasons (as so elegantly illustrated in Rogers's [49]) in the pursuit of recursion theory. It indeed has the status of an axiom. Nevertheless it is an explication, as it supplies "previously intuitive terms . . . with . . . precise meanings" (p. 20).

Epistemologically it is a posteriori or, in the above classification, a priori in the lowest grade; i.e. it is empirical in both the fourth and fifth senses of a prioricity just delineated, depending on what you are after. If you are interested in abstract functions, the same sort of dialectic you would employ to warrant acceptance of the usual definition of 'continuity' would obtain for CT. In that case a suitable reading of Church's Thesis would be

(2) All effectively computable functions are recursive.[4]

On the other hand, if you are interested in the nature of effectiveness and algorithms, CT is empirical in the stronger sense: you would read it as a statement about computists and algorithms relative to certain classes of functions. A suitable reading from such a standpoint might be

(2′) Any function whose values can be algorithmically generated by a computist is recursive.[5]

(2) is an elliptical reading of (2′); (2′) unpacks the antecedent of (2).

I claim no novelty for the distinction, as it just reflects two rather different classes of argument that have been used in recent years in support of CT. For instance, the argument that no computable function has been found that is nonrecursive directly supports (2), insofar as it has any force. Of course it supports (2′) as well. But the point is, such reasoning does not deliberately heed questions of what computists can do, whether anyone knows an algorithm (even though one exists), and the like. Similarly the argument from the invariance of the known explications of 'effective computability', namely, that recursiveness, λ-definability, program computability, Turing computability, Markov algorithm computability, etc., characterize one and the same set of functions in different ways, applies directly to (2). That argument simply says that if mathematicians strongly believe 'if P then Q_1', 'if P then Q_2',... and believe 'Q_1, Q_2, . . . , are

equivalent', then they are justified in believing the statements 'if P then Q_i'. This is a kind of nondeductive argument and is completely indifferent to the analysis suggested by (2′), although intuitive feelings about computists and the algorithms they might realize would probably have something to do with the strength of belief in the several versions of the conditional.

Moreover, if it appeals to one to adapt a Kuhnian view of science to mathematical logic, it is an impressive fact that the accumulated results of recursion theory, construed precisely as the formal counterpart of informal notions about effective processes, enjoys enormous professional acceptance. The theory is "normal" mathematical logic indeed! Kleene and Vesley ([27], p. 2) mention dozens of contributors to recursion theory with the obvious intent of heralding the professional acceptance of Church's Thesis "whether or not one accepts the thesis [as true]. . . ."

All of these arguments are empirical in exactly the same sense as those that favor other mathematical explications such as that of 'continuity' or of 'area under a curve'. The only difference I can see is that in times past mathematicians have not been as self-critical of their epistemic presuppositions as modern logicians have been. So I count CT, when defended by such standards, as of grade four in my classification and read it in the usual way, (2).

But the so-called 'direct' arguments, such as Turing's ([60] and Post's [44]) are obviously aimed at a reading such as (2′), which is of the fifth grade of aprioricity (i.e., are a posteriori). Turing argues that in getting values of functions algorithmically, computists display certain abilities and powers that are totally captured by Turing machine rules. Similarly, to establish the "full generality [of normal systems], a complete analysis would have to be made of all the possible ways in which the human mind could set up finite processes for generating sequences" ([44], p. 408). For both Turing and Post, support of a thesis equivalent to CT is a matter of showing, in principle, that certain cognitive abilities are matched by idealized systems of computational rules. The principle being defended is an empirical one best expressed in something like (2′) where, in strictness, 'recursive' should be replaced by 'Turing computable', or 'Post generable', as the case may be.[6]

The vagueness of (2) is perhaps traceable to 'effectively'. Quite clearly computability is a property of functions, but effectiveness seems to have a rather mixed reference. Church's discussion ([12], p. 89), which is in terms of several mathematical examples, scarcely gives a clue, although elsewhere he writes of effectiveness of formulas ([13], p. 50). In (2) 'effectively' is an adverb that modifies 'computable'. It restricts the abstract set of tuples denoted by 'computable function'. But in (2′) the concept is replaced by a notion of a computist producing function values via algorithm. (2′) already suggests that 'effectively' does not apply directly to computable functions as abstract mathematical objects, but to descriptions of functions and to symbol manipulations by agents. Indeed, 'effective', turns out to be adequately interpreted only as a predicate of expressions or other concrete objects.

Shapiro [52] seems to be the first person to have deliberately applied the distinctions most careful philosophers would use in studying linguistic reference to this area of philosophy of mathematics. Functions, he rightly insists, must be distinguished from presentations of functions. A function is an abstract set

of ordered tuples. A presentation of a function, on the other hand, is a linguistic expression that describes the function. For instance '$\lambda x[x + 1]$' is a presentation of the set of ordered pairs $(x, x + 1)$, for natural numbers, x. Similarly, 'the least integer greater than the number of hickory railway ties laid in region r in the United States between January 1, 1887 and January 1, 1900 divided by Euler's constant' is a presentation of a finite function of r, r ranging over labels $0, 1, 2, \ldots, m$, of disjoint geographical regions, linearly ordered in some way. Presentations are not limited to effectively computable functions, as one example is definitely of that class while the other is not. Of course the concept applies to recursive (Turing computable, etc.) functions; for instance primitive recursive functions are defined by the familiar equation-pair presentations.

This distinction opens the way for an appropriate explication of 'effective' as an attribute of presentations, not functions. We shall see that there are *two* concepts of presentation we must heed, one pertaining to computists *following* algorithms and the other to computists *embodying* algorithms, a distinction that is roughly analogous to the familiar one between a programmable computer, which follows algorithms, and a computer circuit such as a parallel adder, which incorporates an algorithm in its very structure. We shall see that this distinction leads to a further split of (2′) into two readings, one of them expressing precisely Turing's version of CT.

As to the first, keeping in mind the received sense of 'algorithm', a *(unary) computable function* f of natural numbers is one such that there is an algorithm for determining $f(x)$ from x for any x in the domain. As such it is an abstract set of ordered pairs; and computability is a property of functions. A *presentation* of f is "an interpreted linguistic expression that denotes f" (Shapiro [52], p. 210). A presentation of f is *effective* if it suggests (to a computist) an algorithm for determining $f(x)$ from x (p. 211); effectiveness is a property of presentations. The first of the above examples is a computable function, and its presentation is surely effective. The second is also computable since it is finite; however its presentation is not effective.[7]

We now say that an *effectively computable* function f is a computable function having an effective presentation, whence CT in version (2′) may be alternatively read:

(3) Any computable function f having an effective presentation is recursive.

The quantifier implicit in 'having an' should be read classically.

The somewhat indefinite idea in (2′) expressed by 'can be algorithmically generated' has now been replaced by that of a presentation of a function that suggests an algorithm to a computist.[8]

Next I will introduce a notion related to presentation that calls not for denotation of a function, but rather for denotation of a *processor* of a function. Quite roughly, the idea is this. One can think of an algorithm as a list of instructions a computist follows; or alternatively, as a structure such as a special purpose Turing machine or a human agent, the computist, which embodies instructions. For comparison, a presentation of the successor function, as above, might be '$\lambda x[x + 1]$'. And this presentation suggests an algorithm that could be written into a computer program. On the other hand a *processor* structure for that function might be the sequential machine denoted by

$$y(0) = 0$$
$$y(t+1) = -x(t) \vee y(t)$$
$$z(t) = -(x(t) \vee y(t)) \vee (x(t) \wedge y(t)).$$

This pair of recursion equations denotes (in fact, algorithmically translates into) a binary circuit, provided that we understand x, y, z (input, next state and output, respectively) as variable propositional functions of time and the $\vee$ and $\wedge$ operations as the usual logic connectives. It is easy to verify that the circuit, which is equivalent to a one-way tape Turing machine, computes the successor function. It does not "suggest" an algorithm to be followed by a processor, but indeed *is* a processor and *embodies* an algorithm.

This second kind of presentation is to special Turing machines roughly as Shapiro's original concept of presentation is to universal machines or programmable computers. The distinction apparently is not very clear to many writers who talk of finite automata or Turing machine "programs", as if an automaton or Turing table (set of quadruples or quintuples) were such. Of course a universal Turing machine follows a program consisting of encoded quadruples of a simulated machine in virtue of itself having a certain processor structure; i.e., of embodying (being) a set of quadruples that enables it to follow the problem. But following and embodying are significantly different in import, as we shall see.

An example from psychology is language translation. One who knows but little Russian might be able to translate sentences into English using a list of grammatical rules, assuming a complete, effective set were known for Russian. From the mathematical point of view, the translator would use presentations to get an algorithm for computing a map from Russian to English. But a fluent speaker of both Russian and English would effect the translation directly, making no explicit appeal to rules. The required instructions will have been internalized or "built in" to his cognitive repertoire.

The example of the successor processor is already, as I said, essentially a Turing machine. However it must be emphasized that function processors as well as function presentations are not to be limited to effective functions. 'Processor structure' should be taken in a suitably vague way to include beating hearts, cracking plants, and robot controls just as long as they are 'discrete state' and their inputs construable as numerals or strings on finite sets of objects of some sort. In other words, processors are computer-like, but need not be effective. We shall need to say more about processors in Section 4.

More precisely, let x, y, and z be strings (see Note 7), and let us take the idea of structure or "black box" with inputs and outputs as primitive. A *processor* is a structure that produces output y from input x; if it produces z from x, then $y = z$. In other words a processor is functional. By specifying that a processor "produce" an output we allow the possibility that a processor be nondeterministic; it might have more than one way of producing a function value. The *function f computed by a processor* is the set of ordered pairs of strings (x, y) such that x produces y; i.e., $f(x) = y$. A *computable function* is the function computed by a processor. An *effective processor* is one that embodies an algorithm for determining $f(x)$ from x where f is the function computed. 'Embodies' is perhaps not crystal clear, but examples should suffice to make the meaning

plain. A Turing machine embodies an algorithm, as does a finite automaton, a McCulloch-Pitts ([33] or Kleene [25]) nerve network, a brain neural network (on the mechanist hypothesis, at any rate), an automatic dishwasher control, and a Coke machine that makes change. An *effectively computable function* is the function computed by an effective processor.

(2′) might now be read using this second meaning of 'presentation', *viz.* 'processor':

(4) Any function computed by an effective processor is recursive.

There is a good reason, inessential for logic or recursion theory but important for the broad defense of CT, for replacing 'recursive' in (4) by 'Turing computable':

(4′) Any function computed by an effective processor is Turing computable.

Definitions of 'recursive function' refer to sequences of functions to which certain schemes may be applied ([26], p. 275). Clearly recursive functions are thus presented by interpreted expressions in a kind of formalism that indeed suggests a computational procedure. That a function is Turing computable, however, means there is a machine that computes it, not by following a routine suggested by a verbal presentation (except for universal or programmable machines) but by tacitly following built-in instructions that inform its structure. If we are to think of recursive processes as *models* of effective processes, which seems to be demanded by mechanism, then (4′) is the more natural rendering of CT when read as an assertion about processors rather than presentations.

Although the distinction between (3) and (4) is of no theoretical moment from the point of view of a worker in recursion theory, it does highlight for us the role of computists and the distinction between expressly following a prescribed routine and having one built in. It is worth stressing that 'presentation' in my sense of 'processor' is psychologically the more fundamental notion. To formulate an algorithm as suggested by a presentation and to follow it in an actual computation clearly itself requires built-in, tacit, largely unconscious, cognitive processes which are themselves, by the mechanist hypothesis, operations for underlying computable functions. For instance, recognition of a symbol as a token of an input type (a power uncritically assumed for Turing machines) seems to demand computation of a characteristic function of a symbol type. Indeed, all computations from presentations necessarily depend on the presence of subprocesses of some sort (more on this in later sections).

To summarize the results of this analysis, CT is an explication of 'effectively computable function', is methodologically a priori for the working mathematical logician, and yet is empirical in the sense that (a) it is supported by nonmathematical dialectical arguments much as those that support explications of 'continuity', when it is read as relating abstract entities (2), and (b) it asserts that actual procedures of following algorithms, (3), or embodying algorithms, (4) and (4′), can be modeled by recursive functions or automata, when it is read as an assertion about computists.

Despite what I believe to be the compelling evidence, mainly to follow, that CT is empirical, there is no lack of arguments that it is not. I will consider three. The first shows the epistemic importance of our distinction between (3) and (4).

Thomas ([57], pp. 21–22) has argued against the empirical interpretation on the grounds that we could not falsify, by any conceivable experiment, the proposition that a given physical object processes a partial recursive function only. We could never carry out the required infinite number of input experiments to check whether a computable function is recursive.

Perhaps we could follow him and make out a case against CT as an empirical statement along these lines if we persisted in reading it in style (3). For suppose it is known that a physical object effectively processes input that by coding is homomorphic to a natural number input or a string of a free semigroup. Then the problem is whether or not the process is recursive. Unless the object in question were a human computist or a digital computer (which of course would give the whole game away!) it simply would not suffice to display a presentation of a nonrecursive function and claim the object computed from *that*. Some kind of verificational experiment would have to be performed. However, the function would have an infinite domain; and it would seem "that at no time t would we be able to say 'the machine M has computed the nonrecursive function . . . '" (p. 22). For, assuming each step of computation would take a finite time, however small, one could never establish computability over anything except a finite domain in which case we could conclude only that the function is recursive.

However, even if we accept *this* argument, it overlooks the possibility of establishing the nonrecursiveness of a function computed by a machine or other object via examination of machine structures, what we are terming "processors". There is no known reason for believing that a processor for computing nonrecursive functions, which is at least a logical possibility, would consist of the embodiment of anything but a *finite* set of relations. For its operations would have to be effective or neither the thesis nor its denial would be relevant. As Wang has suggested, anything a "physical object can do reliably and systematically would seem to be effective" ([63], p. 87). So sticking to Turing machine embodiments, it would seem to be always possible in principle to see whether the process is governed by a finite set of quadruple rules or the equivalent, or not. I conclude that CT read fully as in (4) is falsifiable in principle.

A similar argument has been advanced that 'effective computability' as analyzed by Turing is not 'human computability' on grounds that Turing wants to allow computations of all finite length as effective ([57], p. 65). Of course this judgment would also apply to digital computers or other material objects in this finite universe. A related point ([53], p. 362) is that the converse of CT is hardly "trivially" true since recursiveness is unlike *actual computability*, which is limited to bounded input. Recursive functions are not actually computable, as the arguments from the relevant domains are coded in arbitrarily long strings. Again, many authors express the idea that parts of recursion theory "idealize" human memory capacities by imagining infinite resources. Benacerraf ([1], p. 19) writes of "the simplifying assumption that humans internalize certain (infinite) recursive devices", as if the finitude of the brain meant that modeling by Turing machines would be problematic. The upshot of these observations is that CT has no relevance to empirical objects such as humans.

Arguments of this kind are mistaken. A function f is computable if there is an algorithm for computing $f(x)$ from x for any input x. The algorithm must be *uniform*: i.e., independent of any input and in particular of its length,

bounded or not. The definition does not say that for each input one can find some procedure or other. . . . If so, it might be the case that the length would have something to do with computability. But it does not.

It is true that the length of tape has a subtle relationship to the class of functions computable by the machines of a certain family. For some classes of automata, tape serves both for input and auxiliary memory.[9] For instance, the linear bounded automata—Turing machines whose maximal tape length in a computation is a linear function of the input—cannot compute all of the primitive recursive functions (Myhill [36]). But what limits these automata is the rule structure (automaton table), which includes special symbols that block unbounded tape use. As is true of all Turing machines, their tables are finite collections of rules.[10]

It is entirely possible that human beings as well as other empirical objects could compute any effectively computable function in the (correct) sense that they could master a presentation suggesting an algorithm or otherwise embody a processor structure for doing the same. Running out of "tape", if and when they do, has nothing whatsoever to do with what they can do.

In answer to Benacerraf, what we suppose humans internalize, if anything, are finite systems of rules. The only infinity assumption needed is that humans be given unbounded time and limitless stacks of pencil and paper.

Still another way of answering these objections as to the empirical standing of CT is to insist that computing power is *dispositional.* In all aspects of the theory of computability, what interests the investigator is what a computist could do if it were provided with arbitrary input. The concern here is with competence, not performance.[11] As Boolos and Jeffrey remark ([2], p. 19), one of the essential requirements of the theory is to be able to "use this notion [computability] to prove that certain functions are not computable . . . even if physical limitations on time, speed, and amount of material could somehow be overcome".

An entirely different line of argument stems from intuitionism. It is an interesting possibility that CT is an a priori mathematical proposition after all, even though it does make essential reference to computists and their mental powers, somewhat as in (4). Intuitionism already takes all mathematical objects as creations of the mind. It is grounded essentially in a conceptualist psychology. The very idea of a proof or computation entails that of a prover or computist. For instance 'there is an x' intuitionistically means 'any competent mathematician can find or show he could find an x, in accordance with intuitionistic principles of constructive reasoning'. Similarly, the idea of an intuitionistically definable function of natural numbers already ascribes certain constructivist powers to the mind. So perhaps there is a suitable formulation of CT as a principle of intuitionistic mathematics. One might interpret our notion of processor intuitionistically: the structure or presentation underlying computational thought is whatever a full-blown intuitionistic theory of mathematical mentality says it is; and this structure could be identified in CT with Turing rules or recursive equations.

In fact in [28] (p. 145) Georg Kreisel attempts to express CT as a formula of intuitionistic first-order arithmetic with added function variables as follows:

(5) $\forall f \exists x \forall y [\exists u T(x,y,u) \ \& \ \{x\}(y) = f(y)]$

where 'f' ranges over certain constructive functions of natural numbers, 'T' is the Kleene T-predicate and $\{x\}$ is the indexed recursive function whose value for y is $\{x\}(y)$.

This appears to succeed in capturing the Thesis in a formalized part of intuitionistic mathematics in something like the way we have suggested. In particular, the f's are constructive functions, a concept that already comprehends the notion of a mathematical mind endowed with certain illative powers. Kreisel [29] has even gone further and suggested that by taking CT as an axiom for an intuitionistic system one might be able to develop a precise theory of constructivity and related concepts.

But if 'f' is meant to range over intuitionistic entities, it cannot also be consistently intended to range over effectively computable functions, for reasons discussed in the next section (I shall be arguing that the effective functions and the constructive ones in the sense of intuitionism are different classes or at best have irreducibly different presentations). Moreover, if these functions are constructive, expression (5) involves a circularity ([63], p. 96): the quantifiers x (ranging over indexes of systems of equations or Turing machines) and u (over derivations or Turing computations) must be constructive in some appropriate intuitive sense that seemingly involves CT itself. On the other hand, if (5) is interpreted classically it is false.

3 Arguments against Church's Thesis This is no place to canvass all of the arguments that have been mustered against CT during the past fifty years. Except for the indirect arguments that hinge on the use of Gödel's theorems, most of them have turned out to be unconvincing to most logicians, if not to their authors. I do, however, want to examine a small selection from among them for the additional light that might be shed on the empirical character of CT. The discussion falls into three parts: (a) arguments against the positive arguments for CT; (b) arguments to the effect that CT is false; (c) indirect arguments against mechanism, which presupposes CT.

(a) William Thomas [58] maintains that any of the arguments meant to support Church's Thesis as a *mathematical thesis* carry very little conviction, even that they are not arguments that mathematicians, who are otherwise the exemplars of good reasoning, should be proud of. Every argument including that from the equivalence of recursiveness, lambda-definability, etc., alluded to above, ultimately reduces to showing that all of the effectively calculable functions so far encountered turn out to be recursive. This is a kind of empirical induction of the weakest sort. Strictly speaking he is right; such arguments have little standing as mathematics. But neither do many other arguments which are meant to support mathematical explications. Nondeductive, roughly empirical, dialectical arguments are all that are available for *any* mathematical concepts in what we have called the fourth and fifth grades of aprioricity, in particular for explications.

(b) An obvious test of the Thesis would be to search for an effectively computable function that is not recursive. To my knowledge all such attempts have failed, and this fact in itself has been held to be a kind of confirmation of CT as just urged. However, let us consider two of them to see what goes wrong.

There are two intuitionistic arguments I know of, one of which leads to a denial of (5). Since I already have doubts about (5) as an expression of CT, I don't have anything to add to the rather extensive commentary on this assault on CT. Another argument seeks to establish that the class of intuitionistically computable functions properly includes the recursive functions. If this is intended as grounds for a refutation of CT it must establish some kind of connection between the vague notion of an effectively computable function and the equally vague notion of a constructive function.

From a certain intuitionistic choice principle ([27], p. 14) one can show that any subset of the set of natural numbers you choose is intuitionistically enumerable. This principle, expressed in an appropriately intuitionistic language including function variables, is as follows:

(6) $\forall x \exists f A(x,f) \rightarrow \exists f \forall x A[x, \lambda y f(\langle x,y \rangle)]$

where $A(x,f)$ is any formula in which x is free for f. Now let A be any subset of the set of natural numbers. Then one can show constructively that

$$\forall x \exists f[(x \in A) \leftrightarrow \exists y(f(y) \neq 0)]$$

holds, from Kripke's principle [30], (Troelstra, [59]). From the choice principle, it follows that

$$\exists f \forall x[(x \in A) \leftrightarrow \exists y(f(\langle x,y \rangle) \neq 0)].$$

Thus f enumerates A: i.e., $A = \{x \mid \exists y f(\langle x,y \rangle) \neq 0\rangle$. So A is intuitionistically enumerable.[12] But it is not true that any subset of the natural numbers is recursively enumerable. Hence it is said that the general recursive functions are a proper subset of the intuitionistically computable functions.[13] So there is a non-recursive effectively computable function.

But to count this as a refutation of Church's Thesis one would have to assume that 'effectively computable function' is interpretable in a suitably intuitionistic way. However, according to Kreisel this is not quite straightforward as the axioms of an intuitionistic logic suitable for the formalization of Church's Thesis are not "especially evident" (recall (5) and attending discussion) unless 'effectively computable' is interpreted as 'constructive' in a "more general sense of *constructive definability*" ([28], pp. 144–145).

If you were to replace the antecedent of the Thesis with this more general concept and the consequent with that of intuitionistic computability you would no longer have Church's Thesis but rather a new thesis expressing the relation between a novel informal notion of effective computability and intuitionistic computability. An argument for or against *that* thesis might be of some slight interest, but cannot be our topic here. Both propositions could be true, one of the classicist's (Church's) preanalytic concept of computability, and the other of the intuitionist's.

Of course it might be argued that the relatively vague idea of constructivity is more general than that of the equally vague idea of effectiveness in the sense that the former *comprehends* the latter. Assuming that it makes any sense at all to talk of one indefinite class comprehending another, perhaps this is what Kreisel means when he suggests 'constructive' in a "more general sense of *con-*

structive definability". But in my opinion this possibility is precluded following close scrutiny of the underlying sense of the respective concepts. *Constructivity*, if that concept is to have significance for intuitionism, connotes a mathematical method each step of which is validated by primitive intuitions including those that freely construct or exhibit mathematical objects. *Effective computability*, on the other hand, connotes presentations that suggest or embody algorithms that prescribe processes fixed in advance of calculation, requiring no mathematical insight, no validation of a mental act save that it follow a prescribed rule; it connotes clerical process.

This point can be made more concisely using the idea of presentation. Let a 'constructive presentation' be an interpreted linguistic expression that denotes a function and suggests an intuitionistically constructive method of computing that function. Then although it might be shown that computable functions (in extension) are constructive, it is not obviously the case that effective presentations are constructive presentations, nor perhaps even true. If there were to be any hope of showing overlap or inclusion of such concepts, nothing short of deep empirical inquiry into the psychological nature of the processes would be necessary. A subthesis of mechanism might be that informal intuitionist reasoning is a form of recursive computation—but no one is in sight of showing any such thing. Nor the opposite. Intuitionistic reasoning is conscious and deliberate of necessity; but checking through an algorithm need not be—witness machine computation or subconscious cognitive process such as recognition of the grammaticalness of sentences in a natural language. It seems to me that any proposal to formalize the notions of computability, constructivity, intuitionistic definability, etc., within some version of intuitionistic logic—especially if it is to take care in observing the distinction between functions and presentations—in order to understand the interrelationships (Kreisel [29]) is totally out of the question—short of obtaining more knowledge of cognitive processes than we have available today.

I conclude that the argument from the intuitionist choice principle is largely irrelevant to the standing of Church's Thesis, given our present state of ignorance about cognitive processes of reasoning and computation.

A widely discussed argument of Kalmar's [23] purports to show that CT (3) is "unplausible". He presents a nonrecursive function g, and argues it is reasonable to believe that g is effectively computable.

Given a certain general recursive function f of two variables, define

$$g(x) = \begin{cases} \text{the least } y \text{ such that } f(x,y) = 0, \text{ if there is a } y, \\ 0, \text{ if there is no } y. \end{cases}$$

g is nonrecursive (Kleene, [26], p. 324). By CT (3) it follows that g is not effectively computable.

To the contrary, Kalmar argues, there is an algorithm for computing $g(a)$ for any argument a as follows: Compute $f(a,0)$, $f(a,1)$, etc., in turn and at the same time try to prove "not in the frame of some fixed postulate system but by means of arbitrary—of course, correct—arguments that no b exists such that $f(a,b) = 0$". Since f is recursive we can either execute the computation for each natural number a in turn seeking a b such that $f(a,b) = 0$, or otherwise by the

method of proof indicated determine in a finite number of steps that there is no such number b. Now if g were not effectively computable we should have to infer the existence of a natural number a for which there is no b with $f(a,b) = 0$; and yet this fact could "not be proved by any correct means". So it is implausible that g is not effectively computable.

But there is no end of trouble here, some of it concerning what it is that Kalmar has really proved, and some concerning the algorithmic character, or lack of it, of "arbitrary but correct methods of proof". For Kalmar to draw an implausibility conclusion he must derive, if not an outright contradiction, something more than a vague intimation that his proof method involves a kind of computation. Keeping (3) or (4) in mind, it is quite evident that Kalmar's discussion does not suggest an algorithm. The finiteness of the method is not enough, for there are many kinds of cognition that consist of disciplined discrete steps (or that so might be construed) such as the yogi's steps to *samadhi.* Other conditions must be satisfied. There is no algorithm specified since Kalmar calls for a correct (perhaps different) proof procedure for each value of x, rather than the other way around (cf. Moschovakis [35]); as previously remarked, to be algorithmic a proof method must be uniform, at the very least. Mendelson has pointed out that Kalmar's method assumes there is a recursive enumeration of classes of "correct proofs" (Mendelson [34]).[14] But even so, the proofs included in a class of an enumeration need not suggest algorithms (i.e., be effective) even if correct. The fact that there are proof procedures for first-order logic provides us with a standard for the algorithmiticity of a proof. But Kalmar does not suggest a proof procedure since his "correct means" is not in the "frame of a fixed postulate system". Hence it is hard to see that a *presentation* of his method would suggest an algorithm of any sort. So the result is not implausible. There could perfectly well be a proof by some correct means of nonexistence of a least b, while at the same time be no effective presentation of any kind. Kalmar's methodological viewpoint is essentially that the possibility of a vague, "premathematical" notion of proof is sufficient to show the implausibility of a necessarily vague principle, which CT is. But 'effectively computable' is not *that* vague, not so much so that the concepts of algorithm and effectiveness are totally without structure.

(c) Arguments against mechanism that use Gödel's Theorems apply indirectly to Church's Thesis. For the sake of argument, let us assume all human cognitive functions are executed by underlying effective processes. Then by CT – (4) or (4′) – they are recursive or Turing computable, or other: the mind is a machine. Now if mechanism is false, but there is still evidence that the hypothesis of underlying effective processes is true, then CT must be false. Arguments of this kind bring issues to mind that have been pretty thoroughly discussed over the past thirty-five years, and I am not going to trace them out again except to bring front and center the principled view that they are guilty of assuming far more about the capacities of the mind and less about the resources of recursion and machine theory than is available, and ultimately reduce to empirical questions about cognitive powers, minds, and machines.

Suppose you are a Platonist (or conceptualist), and one of your favorite themes is that the mind is able to apprehend universals directly. If this theory is true then mechanism must be false. For, the true Platonist would argue, a

machine might be able to prove the existence of a universal (say a property), but it could never get a direct grasp of it except in the trivial sense of replacing a quantified variable with a name. Hence if a direct grasp is *effective*, Church's Thesis must be false. Could intuitive "grasps" be effective?

Here is an argument to the effect that grasping universals is effective in precisely the sense relevant to the antecedent of CT (4′). Suppose you are to follow an algorithm which has the present instruction: 'read the symbol next to the mark # and determine whether it is an instance of the type x; if it is, do so and so; else do something else'. Interpretation and execution of this instruction depend on an underlying cognitive process, as observed earlier. Reading 'x' and determining it to be an instance of a type is a computation of a function, and it is effective. The function here is a characteristic function for the symbol type x (a universal); the determination is a computation, for it tacitly follows an algorithm embodied in the mind that matches the instance with a universal. So grasping universals is an effectively computable procedure.

On the other hand, it is claimed, the function is not Turing computable, for instantiations of Turing machines manipulate concrete symbols, not types; and they don't grasp types to match against tokens. To the objection that computers recognize symbols every day, the conceptualist would reply that computers do not *recognize* anything; they react causally to physical input. The user *interprets* them as recognizing symbols. I will answer this argument in Section 4.

Of more direct relevance to Gödelian arguments, the Platonist (or conceptualist) might argue that the mind can intuitively grasp *true* interpretations of formal arithmetics, but no machine could. Again, inasmuch as a machine is a syntactical system it could not really be minded (Searle [51]): it could not have beliefs or grasp meanings.

The first two arguments attribute capacities to humans they are not known to have or even suspected to have on any naturalistic grounds I know of; and the third assumes that grasp of meanings requires mental abilities which are inexplicable in purely mechanistic terms. Assuming anyone defending such a view has a respectable theory about grasping meanings, this is not necessarily true even in the face of the fact that semantics in formal metamathematics cannot be reduced to syntax.

I wish to comment below on the second negative argument, which Lucas [31] among others has advanced. Then in Section 4 I will consider the argument about symbol types as an exhibit of an empirical question that favors CT, which is of course my ultimate concern. I assume the reader is familiar with Lucas-type arguments to the effect that we can see-to-be-true certain Gödelian formulas that are not provable in an adequate arithmetical logic.

Now an objection to Lucas's line is that such a formula S *can* be proved in another formal system. But then, says Lucas, still another formula S' of the new system will be seen to be true by the mind, but is not provable in the new system, and so on. The mind, he says, can always "go one better" and "always has the last word". At any stage it can perceive the truth of a formula the mechanical system cannot prove. So the mind is not a machine.

There is a direct answer to this argument based on the idea of a universal Turing machine. Suppose the formulas in question are S, S', S'', S''', etc. Suppose that W is the set of theorems provable by the formal system M, W' by M',

W'' by M'', W''' by M''', etc. These formal systems can all be considered to be Turing machines. Now it is agreed that although $S \notin W$, $S \in W'$; moreover $S' \in W''$, $S'' \in W'''$, etc. Since each W is a recursively enumerable set it can be generated by one and the same machine, i.e., a universal Turing machine. So it is false that the mind can always "go one better" (cf. Dennett [16]).

Now it might be objected the machine has to be programmed in order to simulate any other given machine. But programs can be stored. There is no evidence that Lucas's seeing-to-be-true is anything other than "pulling out" a program from the store. In fact this is close to the idea of mechanical mind he seeks to lay low.

Another possible rejoinder is that the infinite union of the sets W computable by the universal machine can be imagined to be the output of some sort of logic having a Gödel sentence S which the machine cannot prove. But to say one could "produce as true" such a sentence is nonsense.

Putnam pointed out long ago [46] that arguments of Lucas's sort are based on a misunderstanding of Gödel's theorem. Given a machine M, all Lucas or anyone else can prove is that for an undecidable formula S, if M is consistent, then S is true. So even in my response above there is a hidden assumption that the systems in question are consistent, unless all of us engaging in the argument have some transcendent gift of seeing the truth of S absolutely. So let us imagine that Lucas means to "produce as true" the statement that M is consistent. Then if one can indeed produce the required consistency, S can be seen to be true even though M can't prove it.

Chihara [7] has commented that Lucas's reasons for convincing himself of the consistency of M amount to little more than a disinclination to live with inconsistency. Again, it is questionable whether a *mathematician* could come up with a respectable consistency proof method for arbitrary adequate formal arithmetics. So Lucas-type arguments amount to attribution of somewhat doubtful intellectual powers to persons and then denying them of machines (Nelson [39]). At any rate, the issues don't boil down to questions that can be settled a priori by appeal to any philosopher's intuitions.

There remains the possibility of reformulating Lucas's argument in careful enough manner to withstand the sort of objections just reviewed. What I have in mind is Benacerraf's [1] interesting reconstruction of Lucas's argument. Although it flushes the vagueness out of the argument it still places mechanism, and along with it CT, in an unfavorable light as it casts a shadow of doubt on the possibility of cognitive psychology as a science, if mechanism is true. But this argument, too, boils things down to an empirical residue. What at first appears to be a compelling a priori argument about the very possibility of psychology turns out at best to indicate possible limitations on psychology, to its impossibility—a result which I regard to be the most telling of plausible consequences of Gödelian arguments against mechanism, if we accept Benacerraf's premises.

Benacerraf's treatment depends on some fairly plausible assumptions needed to make the argument from Gödel technically unobjectionable. These are about relationships between what a person can prove and what machines can prove and are as follows:

Let S be the set of statements B (Benacerraf or you or me) *correctly* proves;

let S^* be the deductive closure of S under first-order logic with identity. Let Q be the theorems of an adequate arithmetical logic; and let W_x be the recursively enumerable set of theorems proved by the xth Turing machine M_x, where there is an effective map from the set of indexed machines to machine "programs", i.e. tables. Now suppose there is a W_j such that

(a) '$Q \subseteq W_j$' $\in S^*$
(b) '$W_j \subseteq S^*$' $\in S^*$
(c) $S^* \subseteq W_j$.

These assumptions are meant to capture the essence of Lucas's attack on mechanism, *viz*, that B can prove ('B can prove p' means that p is in S^*) that the machine M_j can prove the theorems of Q; that B can prove that B can prove the machine output; and that B's output is included in the machine's: (c) says that B is (deductively) a machine.

From these assumptions, using Gödel's theorems, Benacerraf derives a contradiction. Assuming that S^* actually follows by first-order logic with identity from S (what it means to follow by first-order logic from what B can correctly prove is far from clear (Chihara [7])), this means it is false there is a W_j with properties (a)–(c). But, Benacerraf argues, this by no means defeats mechanism, which is expressed by (c). All we can conclude is that for any W_j, at least one of (a)–(c) is false. Certainly (a) is plausible; and if (c) is assumed to be true, we are left with the conclusion that although W_j might indeed be included in S^*, B can't prove it. Although by (c) he is a machine, *he might not be able to ascertain which one.* He might not know the index j by which to recover the machine table, nor have other means. One can even make the stronger statement that [if knowing the machine table is a necessary and sufficient condition for proving '$W_j \subseteq S^*$'], B categorically cannot know W_j's table, and by (c) *cannot know his own.* Indeed, "Psychology as we know it might be impossible". "I might be barred by my very nature from obeying Socrates' profound philosophic injunction: Know thyself" (p. 30). If this is correct there is an a priori block resting on Gödel's theorems to knowing oneself. As many writers have pointed out this does not rule out the possibility of others knowing B by empirical means (unless, of course, W_j is the same for all persons![15]).

There is a way of rescuing mechanism from this impasse based on recursive function theory itself. Assume that B realizes a universal machine U (cf. [65], pp. 222–229). This requires no more idealization than assuming B is any other kind of Turing machine.[16] Let us also continue to suppose there is a W_j, the output of a Turing machine M_j that U could simulate, which satisfies (a)–(c). I will argue that: (i) $U(= B)$ can know itself; (ii) Benacerraf's contradiction derived from (a)–(c) and Gödel's theorems still obtains and is consistent with U's self knowledge; (iii) assuming that (a) and (c) hold, it remains false that '$W_j \subseteq S^*$' $\in S^*$, although it is quite possible that '$W_j \subseteq S^*$' be a part of the output of U.

(i) Now as to self-description, it was shown by Lee [31] and Thatcher [56] that there is a universal self-describing machine U^s whose output $O(U^s)$ includes a coded representation of itself printed on tape. Briefly (Rogers [49], p. 189), let f be an injective map from Turing machines to strings on machine

alphabets, the value of f for M_a being M_a's table in an admissible code. Let the expression '$M(x,y)$' mean that machine M with input x halts with output y on its tape, where x and y are strings on M's alphabet. Now let h be the total recursive function chosen so that $M_{h(n)}$ is the machine that prints $f(M_n)$ symbol-by-symbol from the empty tape, Λ; i.e., $M_{h(n)}(\Lambda, f(M_n))$. Let ϕ_n be the partial function computed by M_n. By the recursion theorem (Kleene, [26]; Rogers, [48], p. 280), $\phi_n = \phi_{g(n)}$ for any recursive function g, in particular for h just defined. Consequently, $M_n = M_{h(n)}$. So $M_n(\Lambda, f(M_n))$, which means that M_n is able to print out its own table in code computing from empty input.

Finally it is a straightforward exercise from this result to show there is a universal self-describing Turing machine U^s. Starting from an encodement on tape of an arbitrary machine and an input x, it prints out symbol-by-symbol the value of the function computed by that machine and its own, that is U's, code script. Let $U = U^s$.

(ii) There is no change in Benacerraf's result: a contradiction derives from (a)–(c) taken together with Gödel's theorems. It is conceivable that by taking W_j to be $O(U^s)$ this result would not obtain. But then (b) would be false a priori and we should not succeed in reproducing Lucas. We wish to find the *worst* that would follow from such arguments. Here's the reason $O(U^s)$ would be the wrong set to take:

S is B's deductive output and S^* its first-order closure. The output $O(U^s)$ of U^s is everything B could prove given an eternal life, interaction with an environment, unlimited stacks of paper and piles of pencils. $O(U^s)$ includes all recursively enumerable sets, results of heuristic procedures which might include nonrecursively enumerable sets ([40], p. 104), the outputs of semi-algorithms, its self-description and more (cf. [7], p. 513; [24], pp. 439f). No one knows the details, but at any rate $O(U) \not\subseteq S^*$. Consequently since B's output S is *correct*, so is S^*, and (c) must be false. Therefore we take W_j, which is some recursively enumerable set generated by a nonuniversal Turing machine M_j.

(iii) However, since the contradiction goes through, it is false that '$W_j \subseteq S^*$' is an element of S^*, assuming that (a) and (c) are true. But what this means is just that the identity of the *program* for W_j is unknowable by B with deductive powers limited to S^*. It is quite likely that B *qua* realizing the universal machine U^s could prove '$W_j \subseteq S^*$', but we'll leave the question open here.

The proper conclusion to be drawn is that Lucas's argument as regimented by Benacerraf does not threaten mechanism, hence does not threaten CT, and does not threaten the possibility of cognitive psychology. It does suggest that if humans realize universal machines—which in the end is an empirical question that cannot be settled a priori—there might be mental processes that cannot be fully understood; in our logico-mathematical jargon, we might not be able to prove that certain programs have outputs in a certain recursively enumerable class. But this, again, is not to be settled by intuitions as to what one can "produce as true", or by a priori impossibility arguments. Prima facie, man can know himself in the relevant sense; witness contemporary genetics, the logic of which, as Burks [5] and Stahl [52], have discussed, is underwritten by the very theory we have sketched above. A human also would appear to be universal, as given the time, patience, and enough paper, etc., he could compute any partial recursive function.

4 Mechanism Returning to positive arguments for CT, let us pick up the empirical line initiated by Post and Turing, which was represented in our analysis by the full readings (3) and (4). Our idea is to focus on specific cognitive functions and show they are recursive by empirical methods, essentially by modeling. The procedure I have in mind can be understood in two different ways, which come down to about the same thing. To fix ideas, consider the example of symbol recognition already alluded to twice in this paper. Let us continue to assume that recognizing a symbol (or any other sensible property type) as a token of a type is an effective process. The first procedure says to apply CT and thereby conclude that the process is recursive, contrary to our Platonistic protagonist of the previous section. This method treats CT as an empirical hypothesis. By experimental means or other considerations of cognitive theory, it is then disconfirmed or perhaps in the long run confirmed, that recognition is indeed Turing (or whatever kind of model we choose) computable. Such an application of CT supports mechanism and adds to the accumulated evidence for CT itself as well by much the same rationale as other empirical hypotheses are sustained. In a way the procedure is reminiscent of Rogers's use of CT since it applies the thesis as an inferential instrument. And just as Rogers's claim that a function is recursive on CT grounds is verifiable by an appropriate recursive derivation, so the claim that recognition is recursive is subject to other, in this case psychological or neurophysiological, tests.

The second procedure accepts the Platonist's assurances that recognition is an effective process and establishes a recognition model, either a program or a processor—a Turing machine perhaps—which it advances as evidence that recognition is a recursive process. This claim is then supported by demonstrating that the model satisfies empirical adequacy conditions. In our example, these conditions must include a universality requirement: that an adequate model must be able to recognize all tokens of a type that a human could, and must be able to discriminate among a large number of types (tell an occurrence of an *a* from a *b*, etc.). If the adequacy conditions are satisfied the method then concludes, for each cognitive trait it takes up, to a specialization of CT, in this case to recognition. Of course the approach also supports mechanism insofar as it succeeds in establishing specific cognitive functions to be recursive, one by one.[17] Ultimately, adequacy of the models (or equivalent ones) must be validated by scientific practice in order to attain long standing. The arguments are at best "plausibility" arguments.

In the remainder of this section I will follow the second procedure and exhibit a model for recognition or, more in keeping with terminology in the literature, for *type-acceptance*.[18] I will then indicate a half dozen adequacy conditions the model should satisfy and sketch an adequacy demonstration. The exercise will be at once an argument for a specialization of CT: all acceptance computations are Turing computable; and an argument for mechanism: mind *qua* type acceptor (intuiter of universals) is a system of recursive rules.

Basically there are two modeling options: write programs that simulate type acceptance, or construct processor models. A computer running under a program, given some widely accepted idealizations, is essentially a universal Turing machine and thus, other things being equal, might demonstrate Turing computability. Processor models could be special Turing machines, sequential

circuits, finite automata, systems of production rules or the like. The construction of any model of this class is a demonstration that the phenomenon at hand is recursive. Satisfaction of the adequacy conditions tends to show the model is relevant, that it captures crucial properties of the subject.[19]

These options are of course suggested by the distinction between presentations that suggest algorithms to be followed and processors that embody algorithms, a distinction we proposed in the analysis of CT and are reflected in (4) and (4′). For reasons taken up in the Postscript, I opt for automaton modeling. Not to put the reader off completely, the main reason for the choice (there are others) is that automaton construction is far more amenable to demonstration of adequacy conditions than programs are (Nelson [41]). Another is that if the mind is a recursive processor in any sense, i.e., if mechanism is true, it must embody, not only follow, algorithms at some point on pain of courting infinite regress otherwise. For if there are program-like cognitive functions that are actualized, there must be an underlying process that executes the instructions. Execution, in turn, must either occur in virtue of another algorithm-following program (as in microprogramming) or of an algorithm-embodying structure (a CPU). Ultimately there must be an embodying structure or nothing will ever get done. So in a sense processors, not programs, are basic.[20]

It is now time to rewrite CT once again as an assertion about processors and automata. Recalling the definitions in Section 2, a processor is a structure that produces input from output. The function computed by a processor is a set of ordered pairs of the usual kind. An effective processor is one that embodies an algorithm for determining $f(x)$ from x. And an effectively computable function is the function computed by an effective processor. A *Turing computable function* is the function computed by a Turing machine; i.e., a processor structure defined by a quadruple table (Davis [15]). In the following I will use 'automaton' as a generic expression to include Turing machines, finite automata, serial and parallel compositions (i.e. direct products) of automata, pushdown automata, universal machines, self-reproducing automata, and so forth. Under this convention, a function is Turing computable if it is the function computed by an automaton. Processor input is to include any finite, nonempty sets of discriminable individuals including multidimensional arrays ([25], p. 372) and parallel inputs. The intention is to include stimulus inputs of all kinds (under suitable codings), so long as they are discrete, to the cognitive machinery.

Consider

(6) Every effective processor is an automaton.

This is just another way of expressing version (4′) of CT; for, by the foregoing definitions, if f is effectively computable, then it is equal to the function computed by an effective processor, which by (6) is an automaton; and the function computed by an automaton is Turing computable.

Proceeding with the example, I take it as an established working hypothesis of cognitive theory that human intelligent functions are effectively computable, i.e., are string functions ("processes") computed by effective processors[21] in the brain or realized by the brain. So recognition is effective, as our Platonist agrees. We have to show that a sufficiently elaborate automaton or composi-

tion of automata is adequate to the task. First, I will indicate adequacy conditions and second present the relevant automaton constructions in enough detail to convey the sense of the procedure, assuming the reader is acquainted with the basics. The reader who is not familiar with automata and Turing machine theory will probably get most benefit by skipping over the technical details (they are sketchy at best, anyway) and turning to the commentary that follows. A detailed exposition can be found in Nelson [38], [40]. If the reader wants them, details about automata structures can be found in Hartmanis and Stearns [19] and Nelson [37].

It should be clear from my preliminary discussion of the example in Section 3 that the central question is whether or not an automaton *qua* system of computational rules can possibly exhibit *intentionality*, or only have intentionality *ascribed* to it by a human being. The Platonist (he's not the only one!) maintains that computers do not type-accept, but that we ascribe acceptance to them only.[22] My reply is, it is possible to capture what we mean by 'intentionality' in the aforesaid adequacy conditions. If there are automata that satisfy the conditions, we shall have shown, I claim, that intentionality can be a feature of a very complex automaton structure. Of course there are deep philosophical questions about the completeness of any conditions one might be moved to write down, and even about the possibility of expressing linguistically what it means for a cognitive attitude to be intentional. I will comment further on these questions after describing the models.

The objects-to-be-type-accepted in this exercise include twenty-six Roman capital letters of some font. Tokens of a single type vary quite greatly in size and level of degradedness and might be presented to the acceptor upside down, rotated any number of degrees, and other nonstandard positions. Standard printing is in black ink, but other colors might occur. There are also other objects that appear to observation as identical individuals but are tokens of other than the Roman types (see conditions II and IV).

Here are the conditions:

(I) *Universality and discrimination.* The model must be able to accept particulars from an indefinitely large class as instances of universal types. It must be able to discriminate among at least the twenty-six types and maybe more.

(II) *Recognition of many types in one set of tokens.* The model must be able to assign more than one distinct type to a single set of tokens. Examples are gestalt patterns. Tokens of one and the same set of 'A's' might be accepted as instances of a picture of an A frame house or of the letter type A. The example is a bit strained. Better, but more complex, are the Necker Cube and Duck/Rabbit phenomena. In music, an example is the first three notes of 'the First Noel' which could be heard as the beginning of 'Three Blind Mice'. This phenomenon is analogous to that of terms in a language which have a common extension but different intensions.

(III) *Recognition of one type in qualitatively disjoint sets of tokens.* It must be able to accept individuals having deviant properties as being of a normal type. It should accept a blue 'B' as a B even though the standard tokens are black. A better example is from music: to recognize a Bach B-flat Chorale played by a brass band in A-flat.

(IV) *Recognition of degraded tokens.* It must be able on some occasions depending on context and expectations to recognize ill-defined, degraded characters, for instance a degraded B as an instance of the symbol type B.

(V) *De dicto acceptance.* It must be possible for the model on occasion to accept the null object as of some type it *expects.* A human might expect to see an 'A' on a paper ground where there is nothing but a blank and still see an 'A'. (It must be possible for it to hallucinate or engage in wishful perception.)

(VI) *Nonveridicality of acceptance.* It must be possible for the model to accept a degraded input x as a type p when it is false that x is p.

(VII) *Failure of substitutivity.* It must be possible for the model to accept a degraded x as a p and not accept y as p, although x and y be qualitatively identical strings.

The first four conditions are jointly *gestalt* conditions which observation shows are satisfied by most humans. The last three conditions are *intentionality* conditions proper, and are suggested by the familiar properties of intentional sentences in natural languages (Chisholm [8], p. 170; Nelson [40], pp. 254f).

It happens that the model for condition (IV) includes a definition of 'taking' and 'expectation' ([8], pp. 182f). This model, which is not fully described here, thus also satisfies conditions on *taking* objects to be such and such, and *fulfillment* and *disruption* of expectations, all of which are intentionality concepts.

As to the model(s), basically there must be twenty-six automata, one for each Roman letter type. We assume there is a common automaton vocabulary, ($s \in$) S, and an infinite set of strings ($x \in$) S^* , on S including the null string, Λ. The automaton structures (next state and output maps) are at most pushdown automata, as some of the procedures require telling in advance whether certain computations will end up in accepting states. This is equivalent to a solution of the halting problem, which is solvable up through the subrecursive hierarchy only as far as pushdown automaton. We think of the elements of the vocabulary as basic *code* components and of each of the twenty-six types as instantiated by an enumerable set of distinct patterns of the components. S^* is the union of all these sets plus more. Thus in simple cases think of the set S^* as partitioned twenty-seven-fold. This includes the case of strings accepted by no automaton.

Each of the automata computes a string function, which is the *characteristic function*, of a single type. The accepting automaton adequate for condition (I) is the direct product of all twenty-six. Suppose for the sake of a reasonably straightforward exposition that all automata are finite state. Let $\langle S, Q_i, q_{i0}, M_i, K_i \rangle$ be the ith automaton where S is the vocabulary common to all twenty-six automata, Q_i is the set of states, q_{i0} is the initial state, M_i the transition function and K_i the accepting states. The direct product is $\langle S, Q, q_0, M, K \rangle$, where Q is the 26-fold cartesian product of states of the component automata, q_0 is likewise a product of the initial states, K of the accepting states, and M is given by

$$M(q,s) = (M_1(q_1,s), \ldots, M_{26}(q_{26},s)).$$

The function M is then extended to $M'(q,x)$ by extending the component functions M_i' in the usual way. Thus we have for $x \in S^*$

$$M'(q,x) = (M_1'(q_1,x), \ldots, M_{26}'(q_{26},x)).$$

This simply says in effect that an input string is processed simultaneously by all automata. In general the set of strings accepted by the ith automaton is infinite, and acceptance of any string is in fact identification of that string as of the ith Roman symbol type. Since S^* is partitioned, any string will be accepted as of at most one type. So the device discriminates. The partitioning convention is lifted in IV. Note that this is a mechanist attempt at explaining how a mind could abstract universals from particulars and discriminate among diverse universals.

As to (II) (keeping in mind an arbitrary one of the acceptors) we assume that different partial recursive functions on one and the same domain correspond to different types, as follows. Suppose f and g are any two partial string functions on domain $D \subseteq S^*$. We assume that the computation of two different functions on the same domain models differential responses to the same set of stimuli—in the example given in the statement of Condition (II), f, say, is the function corresponding to letter type A and g the function corresponding to A-frame house depictions. Now we convert the Turing machines that compute f and g to acceptors by suppressing output and defining halting states as final states.[23] This procedure defines two "characteristic functions"[24]—the functions computed by the modified Turing machines—that are extensionally identical but intensionally other. The processor for each is an acceptor, one that embodies an algorithm derived from the Turing machine for computing f, and the other that embodies an algorithm for computing g. I claim this construction captures the gestalt phenomenon of "seeing" two distinct patterns in a single physical complex, for the phenomenal experiences themselves must depend on dissimilar underling neural processes. So this shows that acceptance of a set of tokens of more than one type is computationally possible. Of course if such scheme were to be realized, technical means would have to be provided to enable just one of the automata on a recognition occasion.

For (III), assume for simplicity (but with no loss of generality up to push-down automata) that all automata are finite state acceptors. Choose a set of generators S' (a vocabulary) disjoint from S but of the same cardinality and define a bijective map ϕ from S to S'. Extend ϕ to an isomorphism from S^* to $(S')^*$ by $\phi(xs) = \phi(x)\phi(s)$. Let $I = \langle S, Q, q_0, M, K\rangle$ be an automaton for an arbitrary one of our types and let $I' = \langle S', Q', q_0', M', K'\rangle$ be an isomorphic automaton; specifically let θ be a bijective map from Q to Q' that satisfies $\theta M(q,s) = M'(\theta(q),\phi(s))$ and such that $q \in K$ iff $q' \in K'$. Then keeping in mind that $\phi(xs) = \phi(x)\phi(s)$, it is easy to show that $\theta(M(q,xs) = M'(\theta(q),\phi(xs))$, where the latter expression entails that I accepts a string $xs = y$ if and only if I' accepts $\phi(xs) = y'$. Inasmuch as the strings correspond under an isomorphism it is plausible to assert that I and I' accept the same types in the sense relevant to (III). They follow the same accepting algorithm applied to disparate primitives, just as one can see the same face in a painting and in a photograph.

(IV) Calls for a complex construction employing the notion of a self-describing automaton. The basic idea we wish to emulate is that of a human being who might under certain circumstances *take* an object that vaguely resembles an object of type p he *expects* to see (or hear, etc.) to be of type p. The basic ingredients of the model in addition to standard automata are three: (a) a finite set B of elements disjoint from S; these are undefined for all twenty-six automata and model the idea of vagueness or degradedness. In the example think

of them as partially broken tokens of the twenty-six types, tokens of a different font, and so forth; (b) the concept of a *winner* or *expecting* state: a state q of an automaton is a winner if there is a string $x \in S^*$ such that $M(q,x) \in K$. Note that if an automaton is in an expecting state it "expects" input that would take it to a recognizing state. However this condition does not imply that the actual input during a computation will drive it to a recognizing state or will not drive it to a recognizing state. (If experience leads you to expect an event of a certain kind to occur there is no guarantee such an event will occur, nor that it will not occur.) In general, if W is the set of winners, $K \subset W \subset Q$; (c) an algorithm—embodied in a sufficiently powerful Turing machine—for deciding whether an arbitrary state of an automaton is a winner (this is the point at which it is required that our basic accepting automata be no more than pushdown automata as the winner decision problem is recursively unsolvable for Turing machines in general); (d) the table of each accepting automaton stored in coded form in a superautomaton that includes provision for comparing states.

Such a system operates as follows, supposing now that all strings up for acceptance are elements of $(S \cup B)^*$, which models corrupt input (a). Whenever an automaton receives standard input it transits to the next state defined by its table. However if it gets input from B (as part of a string x), the transition is undefined. In this eventuality the superautomaton checks whether the current state of the subject acceptor is a winner (b,c). If not, the input is rejected. If it is a winner, the superautomaton finds the current state in the coded table (d) and sets the subject acceptor to any one of the winning states, which according to the coded table the current state would transit to if it had legal input. (Ties can be broken in many computable ways.) The system then continues its computation and will accept or reject the string according as its tail from the undefined symbol on to the end of the string (which of course might included other undefined elements) drives the machine to a recognizing or nonrecognizing state.

This construction indicates that a strictly Turing-computational device can account for type-acceptance of degraded or ill-defined strings containing elements that are not in the machine vocabulary.[25] Since this is a feature of gestalt perception animals are capable of, it is a reasonable hypothesis that they, too, realize self-describing structures—a favorite theme of mechanism (Hofstadter [20]; Johnson-Laird [22]; Webb, [65]) but not heretofore applied outside of genetics, so far as I know.

Owing to the capacity of the model to take *input* in the manner described, it is easy to see that the intentionality conditions (V)–(VII) are satisfied. I omit specific discussion of these conditions (cf. Nelson, [40], pp. 254f), except for the following remark. It might conceivably be objected that the model satisfies (V)–(VIII) simply because it is a machine that makes mistakes; and this is hardly what anyone means by having intentions. However *taking something to be* in virtue of being in a winning or an expecting state (even if my definition of a 'winner' does not really explicate 'expect' in an entirely satisfactory way) certainly must be differentiated from error. Construing phenomena in such a way as to satisfy expectations may be epistemically nonveridical but hardly erroneous. Moreover the system described for (IV), when it takes, is not failing—i.e. not outputing either intermittent or catastrophic error.

I claim it is a reasonable hypothesis that the system of models indicated adequately captures the phenomenon of gestalt "recognition", and also the quality of intentionality in type-acceptance.

It seems to me that our simple example of recognition of a particular as an instance of universal type highlights most of the key problems confronting mechanism. It is paradigmatic. Grasping universals in the sense of subsumming a particular under a universal or abstracting a universal from a particular *is* effective. If not, it is hard to see how any algorithm could possibly be executed since a central ingredient is the correct reading and manipulation of symbol tokens that instantiate types. The problem is to *explain in naturalistic terms* how this might be done. CT suggests how: the act of recognizing a particular as an instance of universal can be accounted for in Turing machine terms. If it can't be done mechanically, CT is false. Turing's own argument [60] for the suitability of his machines is precisely one to the effect that atomic acts such as observing symbols are reproducible in his computational models.

Incidentally, the example brings to light the fallacy committed by philosophies that oppose mechanism on grounds that physical devices that embody or follow algorithms are strictly syntactical and hence could not possibly explain cognitive *cum* intentional *cum* semantical phenomena. Proponents of this view, such as Searle [51], claim too much. Even a "syntactical device" (if it were ever explained what *that* is) must be able to recognize tokens as types. And this is already intentional as our analysis shows even if it is too idealized to be true. So granting computational processors are syntactical in no way concedes a fundamental limitation.

One concluding point. There are still those who will insist that intentional phenomena cannot be accounted for naturalistically; by that I mean by some kind of reduction to psychological, neurological, or, as in our case, to essentially mathematical, terms. Of the many reasons for this conservative view the prominent ones go back to what I have called 'Quine's version of Brentano's thesis': One cannot break out of the circle of intentional terms (Quine [48], p. 220). There are many confusing inequivalent versions of this doctrine, one that says definitions of terms such as 'expects' and 'belief' are implicit and in terms of each other; and another that says that psychological laws tend to explain the intentions denoted by these expressions in terms of one another. Of course *if these persons are right and if all cognition is effective, as seems empirically to be the case, CT is false.* There are ingredients in effective processes that simply cannot be reduced to concepts of recursion and machine theory. This is the Platonist's point, and it applies to Turing's own version of CT.

However I claim the conservatives *could* be right only if they could show that adequacy lists such as the above are incomplete*able*, not merely incomplete, or that the model constructions do not actually do what they are supposed to do, for example that recognition or type-acceptance is a phenomenon not captured by our definitions or any other naturalistic one. This is a big order. I don't think anything but detailed, informed counterarguments are worthy of any attention. Arguments from unanalyzable intuitions won't do; and neither will those that associate mechanism with some vaguely conceived programming theory of mind. Strong intutions must be respected, especially in philosophy. But they should lead, not block.

Postscript An alternative to automaton models in verifying CT in specific cases is programming models or simulation. Under our blanket assumption that cognitive processes are effectively computable, I suppose any computer program that runs successfully duplicates some human task we would usually say requires intelligence, and to that extent verifies CT. Of course such programs range from spread-sheet software produced for the market that claim no merit as models of human cognition to, say, theorem provers or natural language processors, which do. I suspect that 'computationalism' (which is popularly preferred to 'mechanism') means to most philosophers and logicians some sort of program theory of mind. Discounting those who appreciate the distinction between programs executed by a stored program computer and automaton tables (including those implicit in the design of hardware), most persons would probably opt for programming models of the mind as the most appropriate tools for philosophical support of mechanism. Webb ([65], p. 235) for instance maintains that the fruitfulness of the mechanist theory of mind "can only be established empirically by programming".

It seems to me this attitude is mistaken, even given the most optimistic predictions (which I share) of the fruitfulness of artificial intelligence programming for understanding the human mind. For there is a wide difference between assuming mechanism as a working framework hypothesis for artifical intelligence and cognitive science, and philosophical arguments to establish that hypothesis. In principle, to show that a specific skill is computable an adequate program will do the job; but in general it is far more troublesome to show a program will do what it is supposed to do than to construct and verify an automaton structure. The usual order of events is to show by abstract mathematical means, including automata theory, that a putative algorithm works and then write a program that realizes it. This has been true in the practice of artificial intelligence itself for years.

Programs realize algorithms that are *followed* by processors. Other algorithms are *embodied* in processor structures. If human cognition is essentially program-running, which seems to me the central assumption of "Strong AI" (Searle [51]), there is no reason that the underlying processor must be a system of recursive rules only. It is conceivable that a program be interpreted by a Cartesian mind that interacts with the brain in program execution. Or a conceptualist's soul. There is nothing in the concept of effective processor that implies *material* or *mechanistic* mind. The point of the symbol-token recognition example was of course to show that the *processor* that interprets and executes an algorithm is mechanical. It seems to me that approaches to cognitive science that build on analogies to digital computer and program paradigms (executive systems, input-output routines, slow, fast, and cache memories, etc.) are philosophically naive though perhaps suggestive for artifical intelligence. Ultimately the processor itself, the brain, must be shown to embody recursive relations if CT is to be verified in the general case.

The "followed-embodied" distinction is already presupposed by many computationalist proposals for managing the problem of intentionality, notably Dennett's [17]. We are justified in ascribing beliefs, hopes, expectations, and actions to programs (hence, by Strong AI, to minds) if program execution reduces down via machine languages to the actions of physical, electronic, or chemical

switches—in our reckoning to instantiated automata. Thus even those forms of mechanism that do not insist as mine does on explications of intentional vocabulary in purely mathematical terms, already tacitly assign what turns out to be an essentially ontological priority to embodied algorithms.

Program-theoretic versions of computationalism are prone, it seems to me, to confusion of problems of program design with adequacy conditions. They are certainly not the same. In pattern recognition theory there is a standing practice, based on an assumption that recognition is some kind of map from tokens to types (cf. Nilsson [43], of working out both matching and signature identifications schemes of increasingly elaborate kinds. All of them present formidable technical problems in algorithm design. Almost none of them, to my knowledge, pay attention to the fact that recognition is many-many, not functional (Condition II above), and to that extent do not heed an adequacy condition. Of course in principle any automaton model can be translated to a program which, in turn, can in principle be shown (or not shown, as the case may be) to satisfy appropriate adequacy conditions. But the demands of writing useful or theoretically interesting programs are certain to be diversionary, as experience abundantly shows, although such demands do not preclude attention to fundamental theoretical questions of the philosophy of mechanism.

NOTES

1. Many neuroscientists also have adopted a mechanist point of view. I do not mean to rule out the significance of neuroscience for the cognitive sciences, and hence for the Thesis.

2. 'Calculable' has generally been supplanted by 'computable'. 'Calculator' connotes 'fixed program' as associated with hand and desk calculators of the 1960s and before. In Turing's version of CT, 'Turing computable' or 'program computable', relative to some fixed, precise programming language, then is to be distinguished from 'effectively computable'.

3. I will continue to write of definitions and explications as "analytic", "a priori", etc., although it is the statements that result from substitutions of definienda for definientia (or vice versa) that are among the analytic, or nonanalytic, etc., properly speaking.

   Wang, in [64], attacks the Carnap–Quine doctrine that analyticity is truth by convention (p. 451) and advocates, following Gödel [18] that it is either a tautology or truth grasped conceptually in terms of some kind of coherence of meanings. For logical empiricism CT would be analytic, and for Quine certainly conventional but would escape the typical analytical–synthetical distinctions of past years, as would many mathematical statements outside of pure logic. Although I follow Quine for the most part I do not intend to revive old business here beyond these remarks.

4. I count the converse thesis as problematic. There are primitive recursive functions that lack effective presentations in the sense defined below (cf. Rogers ([49], pp. 9f) and Note 7 below). However this paper is devoted mainly to the direct Thesis.

5. Reading CT as a proposition about computations of algorithmic functions, we must suppose that the notion of computable function has been extended to fields of

notations, i.e., numerals and strings of identifiable objects of a finite set. Strictly speaking, versions (3) and (4) of CT to follow, in which there is explicit reference to algorithmic procedures, must be read as functions of strings or numerals, since algorithms apply to concrete objects only. For a precise treatment of string theory see [14]. For a wide variety of different types of string functions see [37]. I rely on the reader to interpret functions as relations of numbers, appropriate notations, or strings, depending on context.

6. Shapiro [53] has drawn a distinction along essentially the same lines. He suggests one can read CT either as proposing the substitution of a precise notion of recursiveness for the vague one of effective computability, or as identifying a "fixed, preformal structure underlying mathematical thought" with recursiveness. He notes that the first of these two interpretations is essentially that of Rogers; i.e., of explication in the sense of Carnap, while the second is essentially that of Turing and Post.

7. The interplay of the concept of computability as a property of functions with that of effectiveness as a property of representations requires more study, it seems to me. Some authors (Moschovakis [35], p. 472; Webb [65], p. 54) think an effective function is one having an algorithm. Surely this cannot be right, for then the railway-tie function would be effective. On the other hand Shapiro argues that functions are computable if and only if they have effective presentations. It seems to me that my railway-tie function is again a counterexample. As it is finite, it is computable, by the converse of CT; and trivially there must be an algorithm, according to both Shapiro's and my construal of 'computable'. But the presentation does not suggest an algorithm, so it is not effective. But there are difficulties here even without using CT. If I understand Shapiro correctly, a presentation is *quasi-effective* "if there is an algorithm P such that (in principle) one can establish that 'P realizes the function' [described by that presentation]" ([52], p. 211). There is an algorithm for the railway tie function one could in principle establish given detailed railway histories, 19th century land surveys, etc. So the railway-tie function seems to be quasi-effective, hence computable (p. 215). But by the equivalence of computability and effectiveness of presentations, it is not computable.

   Shapiro's principles about effectiveness, if they are right after all, help defuse certain arguments against CT. One of Bowie's [3] criticisms is a case in point. Bowie means to show that recursiveness and computability are not coextensive whereas he only shows that recursiveness and quasi-effectiveness are not coextensive ([52], p. 226).

   The doubts about the converse of CT raised in Note 4 get reinforced by examples. There are recursive functions that are not effective inasmuch as the expressions that define them do not provide a suggestion of an algorithmic procedure. For example, the familiar "Fermat function" is recursive, assuming the law of the excluded middle. But its received presentation is not effective. If a presentation existed that were effective; i.e., suggested an algorithm for determining which constant function obtained, it would necessarily be true if and only if Fermat's Last Theorem were true.

8. The verb 'suggests' in the definition of 'effective' presentation is not entirely clear, but I shall take it for granted it is clear enough for purposes of this paper. Perhaps the vagueness of 'effective' (and hence 'effectively computable function') can be traced in part to the vagueness of 'suggests', and in part to that of 'algorithm' (cf. Rogers [49], pp. 1–5, for a discussion of the latter).

9. A finite state automaton (acceptor or transducer, cf. Nelson [37]) can be thought of as a box supplied with a one-way input tape that introduces symbols into the box one-by-one. Moreover the input of a finite state automaton is unbounded—it can receive any finite length string of defined symbols. A Turing machine can also be thought of as a box with an input tape. However in that case the box also contains a working tape that is loaded by the input, after which the input is inactive or thrown away. What makes a Turing machine "infinite" is the unboundedness of its working tape, not the cardinality of the quadruple structure, which is finite. A finite automaton (e.g., a digital computer) might also have a working tape; but it will be bounded. Thus the differences in computational power can be traced to working tapes, not to input length or cardinality of the quadruple structures. See Kirk [24] and Nelson [42] for more discussion.

10. Kirk ([24], pp. 442–445) critically discusses many of the so-called "limitations" on the computational powers of finite state automata.

11. I have in mind essentially Chomsky's [10] competence–performance distinction. Also see Nelson [39].

12. A similar view about the effectiveness of arbitrary sets of natural numbers, stemming from vaguely constructive if not intuitionistic impulses, was once suggested by Church himself [11].

13. This view is Dirk van Dalen's, conveyed to me in a private communication. A contrary view is Kleene's: given his concept of recursively realizability, he is convinced that "only number-theoretic functions which are general recursive can be proved to exist intuitionistically" ([26], p. 509).

14. Having made this correction, Mendelson claims Kalmar actually does show that CT is contradictory, not merely implausible. I think this is wrong for reasons given in the immediately following text above.

15. I owe this observation to Mortimer Kadish.

16. We are again assuming something like the Chomsky competence–performance distinction in the idealization that B is a universal machine. The universal machine represents B's competence. If a human being is only a finite automaton, she nevertheless might realize the table of a universal machine, and also a two-way tape, provided that "tape" additions are external—in the environment (see Note 9).

17. Besides recognition (perception), Nelson ([40], Chapters VI–X) attempts constructions for taking, expectation, belief, desire, reference, meaning and a theory of truth for a primitive language based on acceptance rather than on satisfaction.

18. See Hopcroft and Ullman [21]. 'Recognition' seems to imply veridicality. Acceptance yields recognition (perception plus identification—see Sayre [50]) when in accord with objective facts. The interrelationships among concepts here are complex, and there is little general philosophical agreement on what a full theory should be.

19. The forerunner of all such methods is Turing's [60]. His version of CT is established precisely by showing that Turing machine quadruples are adequate to the conditions posed by effective computist operations.

    Beyond Turing, the method of constructing models consisting of Turing or Post production type rules and insisting that they have "explanatory adequacy" in the realm of cognitive science was pioneered by Chomsky [9], [10]. McCulloch and

Pitts [33] present a model of nervous activity and are in the same spirit, as it too uses essentially finite state automata (which are weakly equivalent to finite state grammars and are expressible Turing or Post rules). However the authors are not as expressly bent on meeting adequacy conditions as Chomsky. Indeed their model, although a landmark in the field and deserving of great respect, falls far short of empirical adequacy. None of these thinkers appear to be aware of the involvement of CT in version (6)—to appear shortly—in their investigations.

20. Dennett [17] insists that an adequate cognitive science must not assume "unanalyzed bits of intelligence". Intelligent computer programs tend to show how cognitive features can be reduced to machine operations, which satisfies the demand. Our observation about the primary character of embodied algorithms really says the same thing; for digital machine operations do embody algorithms.

21. Burks [4] argues that all cognitive (in fact, all "natural human") functions can be performed by a finite automaton or digital computer. The part that is relevant to us tries to establish that such functions are all effectively computable, i.e., functions computed by effective processors. Sensory inputs (stimuli and sequences of stimuli), even if they include "every qualitative and quantitative detail that might make a difference in the output" (p. 56) must be finite. So are internal states: "There is a minimal size physiological part that matters in human information processing. . . . Since the human body occupies a finite volume, there is only a finite number of these parts. Also there is a maximum speed at which these parts can operate. It follows that man's transformation of input(s) to outputs is mediated by a finite number of internal states (p. 57). Similarly outputs are finite and the transformations are deterministic, i.e., functional. The concept of just noticeable difference which is involved in this account is well established experimentally, widely accepted, and indeed a cornerstone of cognitive science.

22. Note that acceptance is intentional: in ordinary philosophical terms it is a mental act that need have no *de re* object, but yet need *mean* or *refer*. One might take an object to be of a type in order to satisfy expectations, while the object is really of another (or null) type. Moreover, acceptance entails knowledge of universals, which is not a physical act according to the Platonist or conceptualist. I mean to capture all such aspects in the list of adequacy conditions to follow in the text.

23. The construction indicated essentially copies Davis's construction of a semi-Thue combinatorial system from a Turing machine ([15], pp. 88–91).

24. The resulting automaton computes the characteristic function if and only if the completion of the original partial function is computable (see Davis [15], p. 16). This is guaranteed by the limitation of all functions to those computable by push-down machines.

25. The construction suggests a concept of *virtual* class membership for ill-defined strings. Let R be a recursive resemblance relation, and A a recursive set. (Roughly, R is a resemblance relation if it is the taking relation described informally in the text.) We say that x is a virtual member of A if and only if there is a y such that $R(x,y)$ and $y \in A$. Since A is recursive there is a machine that computes its characteristic function; i.e., accepts y. So we say the machine *virtually accepts* x. The complex machine of our construction virtually accepts a string x since taking is recursive (it is the result of a computation of a composition of automata), and the set determined by a type is recursive as its characteristic function is Turing computable. It has seemed to me that this concept might be a useful alternative to that of fuzzy set in cognitive studies.

REFERENCES

[1] Benacerraf, Paul, "God, the devil, and Gödel," *Monist*, vol. 51 (1967), pp. 9–32.

[2] Boolos, George S. and Richard C. Jeffrey, *Computability and Logic*, Cambridge University Press, Cambridge, 1980.

[3] Bowie, G. Lee, "An argument against Church's Thesis," *The Journal of Philosophy*, vol. LXX (1973), pp. 57–76.

[4] Burks, A. W., "Logic, computers, and men," *Proceedings and Addresses of the American Philosophical Association*, vol. XLVI (1972), pp. 39–57.

[5] Burks, A. W., "Theory of self-reproducing automata and biology," *Proceedings of Conference on Biologically Motivated Automata Theory*, 1974.

[6] Carnap, Rudolf, *Logical Foundations of Probability*, University of Chicago Press, Chicago, 1950.

[7] Chihara, Charles S., "On alleged refutations of mechanism using Gödel's incompleteness results," *The Journal of Philosophy*, vol. LXIX (1972), pp. 507–526.

[8] Chisholm, Roderick, *Perceiving: A Philosophical Study*, Cornell University Press, Ithaca, New York, 1957.

[9] Chomsky, Noam, *Syntactic Structures*, Mouton & Co., The Hague, 1957.

[10] Chomsky, Noam, *Aspects of the Theory of Syntax*, MIT Press, Cambridge, Massachusetts, 1965.

[11] Church, Alonzo, "The Richard paradox," *The American Mathematical Monthly*, vol. 41 (1934), pp. 356–361.

[12] Church, Alonzo, "An unsolvable problem of elementary number theory," reprinted as pp. 89–109 in *The Undecidable*, ed., Martin Davis, Raven Press, Hewlett, New York, 1936.

[13] Church, Alonzo, *Introduction to Mathematical Logic, Volume I*, Princeton University Press, Princeton, New Jersey, 1956.

[14] Corcoran, John, William Frank, and Michael Maloney, "String theory," *The Journal of Symbolic Logic*, vol. 39 (1974), pp. 625–637.

[15] Davis, Martin, *Computability and Unsolvability*, McGraw-Hill Book Co., New York, 1957.

[16] Dennett, Daniel C., "Review of J. R. Lucas, *The Freedom of the Will*," *The Journal of Philosophy*, vol. LXIX (1972), pp. 527–531.

[17] Dennett, Daniel C., "Why the law of effect will not go away" (1975), reprinted as pp. 71–89 in *Brainstorms*, Bradford Books, Montgomery, Vermont, 1978.

[18] Gödel, Kurt, "Russell's mathematical logic," in *The Philosophy of Bertrand Russell*, ed., Paul Schilpp, Harper and Row, New York, 1944.

[19] Hartmanis, J. and R. E. Stearns, *Algebraic Structure of Sequential Machines*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1966.

[20] Hofstadter, Douglas R., *Gödel, Escher, Bach: An Eternal Golden Braid*, Basic Books, New York, 1979.

[21] Hopcroft, John E. and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Co., Reading, Massachusetts, 1979.

[22] Johnson-Laird, P. N., "A computational analysis of consciousness," *Cognition and Brain Theory*, vol. 6 (1983), pp. 499–508.

[23] Kalmar, Laszlo, "An argument against the plausibility of Church's Thesis," in *Constructivity in Mathematics*, ed., A. Heyting, North Holland Publishing Co., Amsterdam, 1959.

[24] Kirk, Robert, "Mental machinery and Gödel," *Synthese*, vol. 66 (1986), pp. 437–452.

[25] Kleene, S. C., "Representation of events in nerve nets and finite automata" (1951), reprinted in *Automata Studies*, ed. C. E. Shannon and J. McCarthy, Princeton University Press, Princeton, New Jersey, 1956.

[26] Kleene, S. C., *Introduction to Metamathematics*, D. Van Nostrand Co. Inc., New York, 1952.

[27] Kleene, S. C. and R. E. Vesley, *The Foundations of Intuitionistic Mathematics*, North Holland Publishing Co., Amsterdam, 1965.

[28] Kreisel, Georg, "Mathematical logic," *Lectures in Modern Mathematics*, ed. T. L. Saaty, John Wiley and Sons, Inc., New York, 1965.

[29] Kreisel, Georg, "Church's Thesis: A kind of reducibility thesis for constructive mathematics," *Intuitionism and Proof Theory: Proceedings of a Summer Conference at Buffalo, N.Y.*, ed. A. Kino, J. Myhill, and R. E. Vesley, North Holland Publishing Co., Amsterdam, 1968.

[30] Kripke, Saul, Lecture Notes in [59].

[31] Lee, C. Y., "A Turing machine that prints its own code script," *Proceedings of the Symposium on Mathematical Theory of Automata*, Polytechnic Press, Brooklyn, New York, 1963.

[32] Lucas, J. R., "Minds, machines, and Gödel" (1961), printed in *Minds and Machines*, ed. Alan Ross Anderson, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1964.

[33] McCulloch, Warren S. and Walter H. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5 (1943), pp. 115–133.

[34] Mendelson, E., "On some recent criticisms of Church's Thesis," *Notre Dame Journal of Formal Logic*, vol. IV (1963), pp. 201–205.

[35] Moschovakis, Yiannis N., "Review of Laszlo Kalmar (1959)," *The Journal of Symbolic Logic*, vol. 33 (1968), pp. 451–472.

[36] Myhill, J., "Linear bounded automata," WADD Technical Note, 60–165 (1960).

[37] Nelson, R. J., *Introduction to Automata*, John Wiley and Sons, Inc., New York, 1968.

[38] Nelson, R. J., "On mechanical recognition," *Philosophy of Science*, vol. 43 (1976), pp. 24–52.

[40] Nelson, R. J., *The Logic of Mind*, D. Reidel Publishing Co., Dordrecht, Holland, 1982.

[41] Nelson, R. J., "Philosophy, artificial intelligence, and existence proofs" in *Machine Intelligence 10*, ed. J. E. Hayes, Donald Michie, and Y-H Pao, Ellis Norwood, Chichester, Great Britain, 1984.

[42] Nelson, R. J., "Models for cognitive science," *Philosophy of Science* (in press).

[43] Nilsson, Nils J., *Learning Machines*, McGraw-Hill Book Co., New York, 1965.

[44] Post, Emil, "Absolutely unsolvable problems and relatively undecidable propositions: Account of an anticipation" (1941), printed in *The Undecidable* (ed. Martin Davis), Raven Press, Hewlett, New York, 1965.

[45] Post, Emil, "Recursively enumerable sets of positive integers and their decision problems," *Bulletin of the American Mathematical Society*, vol. 50 (1944), pp. 284–316.

[46] Putnam, Hilary, "Minds and machines," in *Dimensions of Mind*, ed. Sidney Hook, New York University Press, New York, 1960.

[47] Quine, W. V., "Two dogmas of empiricism," pp. 20–46 in *From a Logical Point of View*, Harvard University Press, Cambridge, Massachusetts, 1953.

[48] Quine, W. V., *Word and Object*, John Wiley and Sons, Inc., New York, 1960.

[49] Rogers, Hartley, Jr., *Theory of Recursive Functions and Effective Computability*, McGraw-Hill Book Co., New York, 1967.

[50] Sayre, Kenneth M., *Recognition: A Study in the Philosophy of Artificial Intelligence*, University of Notre Dame Press, Notre Dame, Indiana, 1965.

[51] Searle, John R., "Minds, brains, and programs," *The Behavioral and Brain Sciences*, vol. 3 (1980), pp. 417–424.

[52] Shapiro, Stewart, "On the notion of effectiveness," *History and Philosophy of Logic*, vol. 1 (1980), pp. 209–230.

[53] Shapiro, Stewart, "Understanding Church's Thesis," *Journal of Philosophical Logic*, vol. 10 (1981), pp. 353–365.

[54] Stahl, Walter R., "Self-reproducing automata," *Perspectives in Biology and Medicine*, vol. VIII (1965), pp. 373–393.

[55] Suppes, Patrick and Joseph L. Zinnes, "Basic measurement theory" in *Handbook of Mathematical Psychology, Vol. I*, ed. R. Duncan Luce, Robert R. Buck, and Eugene Galanter, John Wiley and Sons, Inc., New York, 1963.

[56] Thatcher, James W., "The construction of a self-describing Turing machine" in *Mathematical Theory of Automata*, Polytechnic Press, Brooklyn, New York, 1963.

[57] Thomas, William John, *Church's Thesis and Philosophy*, Ph.D. Dissertation, Case Western Reserve University, Cleveland (microfilm), 1972.

[58] Thomas, William John, "Doubts about some standard arguments for Church's Thesis," *Papers of the Fourth International Congress for Logic, Methodology, and Philosophy of Science, Bucharest, 1971*, D. Reidel Publishing Co., Holland, 1973.

[39] Nelson, R. J., "The competence-performance distinction in mental philosophy," *Synthese*, vol. 39 (1978), pp. 337–381.

[59] Troelstra, A. S., *Principles of Intuitionism: Lectures Presented at the Summer Conference on Intuitionism and Proof Theory at SUNY Buffalo, N.Y.*, Springer, Berlin, 1969.

[60] Turing, A. M., "On computable numbers with applications to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. 42 (1936), pp. 230–265.

[61] Turing, A. M., "Computing machinery and intelligence," *Mind*, vol. LIX (1950), pp. 433–460.

[62] Wang, Hao, *Survey of Mathematical Logic* (1959), republished North Holland Publishing Co., Amsterdam, 1964.

[63] Wang, Hao, *From Mathematics to Philosophy*, Humanities Press, New York, 1974.

[64] Wang, Hao, "Two commandments of analytic empiricism," *The Journal of Philosophy*, vol. LXXXII (1985), pp. 449–462.

[65] Webb, Judson C., *Mechanism, Mentalism, and Metamathematics*, D. Reidel Publishing Co., Dordrecht, Holland, 1980.

Department of Philosophy
Case Western Reserve University
Cleveland, Ohio 44106