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

Wittgenstein versus Turing on the Nature of Church's Thesis

S. G. SHANKER

1 'Turing's machines are humans who calculate' The title of this paper suggests two highly contentious claims: first, that Wittgenstein was aware of the developments in recursion theory that took place during the 1930s, and second, that he disputed the version of Church's Thesis (hereafter CT) which Turing had presented in 'On Computable Numbers' [36]. It will be best to concede at the outset that both themes represent something of a critical liberty; or rather, a corollary. For the subject of this paper is really Wittgenstein's attack on the mechanist terms in which Turing had interpreted his computability results. But one of the central points that Turing was to make in his 1947 'Lecture to the London Mathematical Society' was that the Mechanist Thesis is not just licensed but is in fact *entailed* by his 1936 development of CT [39]. Wittgenstein's argument thus demands careful scrutiny of both the relation of Turing's argument to CT and the cognitivist implications that have been read into CT as a result of Turing's influence.

Before we consider these matters, however, we must first satisfy ourselves that Wittgenstein was indeed intent on repudiating Turing's computability thesis. For it has long been a source of frustration to Wittgenstein scholars that no overt mention of this issue is found in *Lectures on the Foundations of Mathematics: Cambridge 1939* [48]. Indeed, until recently it might have been thought that the title of this paper makes the still further unwarranted assumption that Wittgenstein was even aware of 'On Computable Numbers'. Any such doubts were laid to rest by the discovery of an off-print of the essay in Wittgenstein's *Nachlass*, and even more important, the following mystifying reference to Turing machines occurs in *Remarks on the Philosophy of Psychology*:

> Turing's 'Machines'. These machines are *humans* who calculate. And one might express what he says also in the form of *games*. And the interesting games would be such as brought one *via* certain rules to nonsensical instructions. I am thinking of games like the "racing game". One has received the

Received May 12, 1987

> order "Go on in the same way" when this makes no sense, say because one has got into a circle. For any order makes sense only in certain positions. ([52], §1096)

The latter half of this passage is clear enough: Wittgenstein is saying that Turing's Halting Problem is no more significant than any other paradox in the philosophy of mathematics.[1] The confusing part is the opening sentence. On first reading this sounds hopelessly obscure: a clear demonstration of Wittgenstein's failure to grasp the significance of Turing's thesis vis-à-vis recursion theory. Yet another way of describing the goal of this paper, therefore, will be to see if any sense can be made of this curious remark.

To see what Wittgenstein was driving at here we have to work our way through a prolonged discussion on the nature of calculation in *Remarks on the Foundations of Mathematics* [49].[2] But before we look at this it will be salutary to fill in some of the background to Wittgenstein's thought. In a widely quoted passage from *The Blue Book* Wittgenstein had told his students:

> the problem here arises which could be expressed by the question: "Is it possible for a machine to think?" (whether the action of this machine can be described and predicted by the laws of physics or, possibly, only by laws of a different kind applying to the behaviour of organisms). And the trouble which is expressed in this question is not really that we don't yet know a machine which could do the job. The question is not analogous to that which someone might have asked a hundred years ago: "Can a machine liquefy a gas?" The trouble is rather that the sentence, "A machine thinks (perceives, wishes)" seems somehow nonsensical. It is as though we had asked "Has the number 3 a colour?" ([44], p. 47)

Turing would probably have read *The Blue and Brown Books*, and he would almost certainly have been aware of Wittgenstein's opposition to the Mechanist Thesis. And even though Wittgenstein presents this as a rhetorical question Turing might have interpreted it as an open invitation, which the Turing Test was intended to answer. Wittgenstein's last point (a veiled allusion to Frege) is, of course, that unlike the empirical question of whether a machine could liquefy a gas the question whether a machine thinks is unanswerable because it is logically absurd in much the same manner as whether 3 is coloured; i.e., both are conceptually ill-formed because they transgress rules of logical grammar. But Turing might well have regarded this as a request for the criteria whereby one would respond to each question. If there are no means to ascertain whether or not 3 has a colour it might well be accepted as absurd (viz., a type-violation such as Turing was later to explore), but the point of the Turing Test is to argue that there is no a priori reason why we should not apply the same criteria to machines as to humans; hence the former can indeed be seen as a *bona fide* empirical question.

The point that tends to get lost in the ensuing debate between Wittgenstein and Turing supporters on the logico-grammatical status of the question 'Can a machine think?' is the most surprising of all: namely the date at which the above passage was written. This was in 1933: nearly ten years before Turing began to think seriously about the Mechanist Thesis. Close to the same time Wittgenstein wrote in *Philosophical Grammar*:

> If one thinks of thought as something specifically human and organic, one is inclined to ask "could there be a prosthetic apparatus for thinking, an inorganic substitute for thought?" But if thinking consists only in writing or speaking, why shouldn't a machine do it? "Yes, but the machine doesn't know anything." Certainly it is senseless to talk of a prosthetic substitute for seeing and hearing. We do talk of artificial feet, but not of artificial pains in the foot.
>
> "But could a machine think?"—Could it be in pain?—Here the important thing is what one means by something *being in pain*. ([47], p. 105)

That is, by parity of reasoning, 'the important thing is to clarify what one means by "thought"'. Here is yet further evidence (already familiar from the writings of Curry and Post) of the extent to which the Mechanist Thesis was in the air at least a decade before Turing began serious work on it. From Wittgenstein's point of view, 'On Computable Numbers' thus took on the aspect of a hybrid paper: an attempt to integrate what should be regarded as independent issues in mathematical logic and the philosophy of mind. And it was precisely Turing's bridging argument which concerned Wittgenstein.

This might make it seem all that much more surprising that Wittgenstein and Turing never touched on the Mechanist Thesis in *Lectures on the Foundations of Mathematics*. But there are two important points to be drawn from all this; first, that *Lectures on the Foundations of Mathematics* is exactly that and second, that even when he attacked Turing's version of CT Wittgenstein did not tie this in to the Mechanist Thesis. What Wittgenstein addressed in *Remarks on the Foundations of Mathematics* was solely that part of 'On Computable Numbers' which inspired the assumption that the latter is entailed by the former. That is, Wittgenstein's critique was solely concerned with Turing's prose interpretation of his computability thesis: i.e., with the *philosophical* argument presented in 'On Computable Numbers'.

This last point is crucial to understanding Wittgenstein's argument. Turing discusses the nature of Turing machines ('computing machines') at two different places: §§1–2 and §9 of [36]. In the first instance he defines the terms of his argument; in the second he takes up his promise to defend these definitions.[3] Thus in his justification Turing naturally shifts from mathematics to philosophy as he seeks to elucidate the epistemological basis of his argument; and in so doing he introduces an empirical theme which constitutes the focus of Wittgenstein's remarks. In essence, Wittgenstein's objection was that the mathematical and philosophical strands in 'On Computable Numbers' are not just independent but, more importantly, that the latter misrepresents the mathematical element. The remark that 'Turing's machines are *humans* who calculate' is only concerned with the *prose* presented at §9, but its corollary is that we must go back and reinterpret Turing's mathematical achievement in such a way as to avoid Turing's philosophical confusion.

Turing introduces Turing machines in §§1 and 2 with the following argument:

> We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions $q_1, q_2, \ldots, q_R$ which will be called "*m*-configurations". The machine is supplied with a

> "tape" . . . running through it, and divided into sections . . . each capable of bearing a "symbol". At any moment there is just one square, say the *r*-th, bearing the symbol G(r) which is "in the machine". We may call this square the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". [36], p. 117

This reiterated warning of the anomaly—'so to speak' and inverted commas around 'directly aware'—is the premise which Turing felt constrained to justify.[4] When read as a strictly mathematical affair, §§1 and 2 operate as an abstract outline for how to construct a machine that could be used to execute calculations, with no reason to present the argument in terms of the various cognitive terms with their attendant qualifications. The thesis reads as follows: suppose it were possible to transform a recursive function into, e.g., binary terms. It would then be possible to construct a machine that could be used to compute analogues of those functions if it used some system that could encode those '0s' and '1s'. This still leaves out considerable detail about how the machine works, but that itself is part of the significance of the paper. On this picture both the function (the table of instructions) and the argument (the tape input) must first be encoded in binary terms and then converted into some mechanical analogue of a binary system. Turing speaks of the machine scanning a symbol but that is entirely irrelevant to the argument; how the binary input is actually structured and how the program/tape interact is not at issue. For Turing was presenting here, not the mechanical blueprints for a primitive computer but rather, a *logical design* which only five years later he sought to concretize using electrical signals to represent the binary code. And he did this with a version of CT which demonstrated that:

> All effective number-theoretic functions (viz. algorithms) can be encoded in binary terms, and these binary-encoded functions are Turing machine computable.

The argument which Turing defends at §9 is not *this*, however, but rather the original premise that 'We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions q_1, $q_2, \ldots, q_R$ which will be called "*m*-configurations".' He begins by describing some of the conditions which govern human computing. The crucial part of his argument is that:

> The behaviour of the computer at any moment is determined by the symbols which he is observing, and his "state of mind" at that moment. . . . Let us imagine the operations performed by the computer to be split up into "simple operations" which are so elementary that it is not easy to imagine them further divided. Every such operation consists of some change of the physical system consisting of the computer and his tape. We know the state of the system if we know the sequence of symbols on the tape, which of these are observed by the computer . . . and the state of mind of the computer. . . . The simple operations must therefore include:
>
> (a) Changes of the symbol on one of the observed squares.
> (b) Changes of one of the squares observed to another square within L squares of one of the previously observed squares. . . .

> The operation actually performed is determined . . . by the state of mind of the computer and the observed symbols. In particular, they determine the state of mind of the computer after the operation is carried out.
>
> We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an "m-configuration" of the machine. . . . ([36], pp. 136–137)

This argument raises what is perhaps the most intriguing question à propos Wittgenstein's objection that 'Turing's machines are really *humans* who calculate'; for this looks like the exact opposite: viz., that Turing has actually defined human calculation in mechanical terms so as to license the application of quasi-cognitive terms to the operation of his machines. Why, then, did Wittgenstein not make the converse point that 'Turing's humans are really machines that calculate'? The answer to this lies in an investigation into the nature of *calculation* in book V of *Remarks on the Foundations of Mathematics.*

It is important to bear in mind when reading this material that at the time when this was written (1942–1943) 'computer' signified 'calculating machine'.[5] Thus when he referred to a 'calculating machine' what Wittgenstein had in mind was most likely a Turing machine, and the following passage can be seen as a direct response to Turing's argument at §9. Wittgenstein asks:

> Does a calculating machine *calculate*?
>
> Imagine that a calculating machine had come into existence by accident; now someone accidentally presses its knobs (or an animal walks over it) and it calculates the product 25×20.
>
> I want to say: it is essential to mathematics that its signs are also employed in *mufti.*
>
> It is the use outside mathematics, and so the *meaning* of the signs, that makes the sign-game into mathematics.
>
> Just as it is not logical inference either, for me to make a change from one formation to another (say from one arrangement of chairs to another) if these arrangements have not a linguistic function apart from this transformation. ([49] V, §2)

Perhaps the most difficult aspect of this section is simply knowing how the various points are supposed to link up with one another. Certainly one can appreciate how, if read out of context, this passage would appear to be hopelessly out of step with recursion theory, for Wittgenstein seems to be denying the one part of Turing's argument that no one has ever questioned: the idea that recursive functions are *mechanically calculable.*

The point that Wittgenstein was driving at here is entirely concerned with the logical grammar of calculation vis-à-vis that of Turing machines. What he was trying to bring out is that the mathematical concept of calculation—as opposed to the empirical concept of counting—cannot be separated from its essential normativity. This becomes clear two passages later when he asks us to

> Imagine that calculating machines occurred in nature, but that people could not pierce their cases. And now suppose that these people use these appliances, say as we use calculation, though of that they know nothing. Thus e.g. they make predictions with the aid of calculating machines, but for them manipulating these queer objects is experimenting.

> These people lack concepts which we have; but what takes their place? Think of the mechanism whose movement we saw as a geometrical (kinematic) proof: clearly it would not normally be said of someone turning the wheel that he was proving something. Isn't it the same with someone who makes and changes arrangements of signs as [an experiment]; even when what he produces could be seen as a proof? ([49] V, §4)

The normativity of mathematics could justly be said to be the main subject of *Lectures on the Foundations of Mathematics* [48] and *Remarks on the Foundations of Mathematics* [49]. This theme is best summed up by Wittgenstein's insistence that 'A proof leads me to say: this *must* be like this' ([48] III, §30). The point of a mathematical proof is that it carves out *rules of mathematical grammar*. 'Let us remember that in mathematics we are convinced of *grammatical* propositions; so the expression, the result, of our being convinced is that we *accept a rule*' ([49] III, §27). That is, the role of a proof is to forge the internal relations which constitute the use of a mathematical proposition as a rule of grammar: 'For the proof is part of the grammar of the proposition!' ([47], p. 370). 'The proof changes the grammar of our language, changes our concepts. It makes new connexions and it creates the concept of these connexions' ([48] III, §31, and see [33], Chap. III).

In Book V of *Remarks on the Foundations of Mathematics* Wittgenstein turned this argument directly onto Turing machines. In what follows Wittgenstein draws attention to how the essential normativity of mathematics underpins the notion of calculation. In the long discussion on rule-following in Book VI Wittgenstein continually reverts to the example of calculation in order to ask the question: under what circumstances can I say of myself, an isolated individual, another culture, etc. that I/they are calculating/following a rule? Wittgenstein's response is to take us through a careful examination of the network of concepts which constitute the normativity of rule-following. The arguments are too lengthy to go into here (see [2]); for our purposes the following passages suffice to give some idea of the core of his thought:

> There might be a cave-man who produced *regular* sequences of marks for himself. He amused himself, e.g., by drawing on the wall of the cave:
>
> -.--.--.--.
>
> or
>
> -.--..-...-....-
>
> But he is not following the general expression of a rule. And when we say that he acts in a regular way that is not because we can form such an expression. ([49] VI, §41)

That is, the fact that we could construct a rule to describe the regularity of his behaviour does not entail that he was following that rule. The question is: under what circumstances could we say that *he* was following a rule?

> Let us consider very simple rules. Let the expression be a figure, say this one:
>
> |--|
>
> and one follows the rule by drawing a straight sequence of such figures (perhaps as an ornament).
>
> |--||--||--||--||--|

> Under what circumstances should we say: someone gives a rule by writing down such a figure? Under what circumstances: someone is following this rule when he draws that sequence? It is difficult to describe this.

The crucial point in what follows is that the answer to this question has nothing whatsoever to do with any putative 'mental events' which might accompany such regular behaviour.

> If one of a pair of chimpanzees once scratched the figure |--| in the earth and thereupon the other the series |--||--| etc., the first would not have given a rule nor would the other be following it, whatever else went on at the same time in the mind of the two of them.
>
> If however there were observed, e.g., the phenomenon of a kind of instruction, of shewing how and of imitation, of lucky and misfiring attempts, or reward and punishment and the like; if at length the one who had been so trained put figures which he had never seen before one after another in sequence as in the first example, then we should probably say that the one chimpanzee was writing rules down, and the other was following them. ([49] VI, §42)

In other words, while the concept of rule-following is internally related to that of regularity it does not reduce to mere regularity. In order to constitute rule-following the behaviour must be seen to be normative. To say of an agent that he/she/it is following a rule, then he/she/it must exhibit the ability to, e.g., instruct, explain, correct, or justify his/her/its behaviour by reference to the expression of the rule.

The gravamen of Wittgenstein's remarks on Turing's thesis remains exactly the same: viz., that calculation is embedded in this cluster of normative concepts, which do not obtain in the case of Turing machines. The point is that *only* coming up with the 'correct results' is not a sufficient condition to say of someone/thing that he/she/it is calculating. But at this stage the obvious objection to Wittgenstein's argument is that this is precisely the point which is meant to be covered by the Turing test. On Turing's account all that matters is that the monkeys satisfy the complex behavioural criteria which govern our use of the concept of calculation vis-à-vis human computers. What it is that causes them (or for that matter, ourselves) to accomplish this is not the issue that matters; only that the resulting behaviour satisfies the same criteria we demand of ourselves. Hence, by parity of reasoning, provided a machine could be made to satisfy those criteria it would be no less irrelevant how *it* achieved this. We need not know whether or not it is empirically possible to build a machine that could satisfy such criteria; all that matters is that there is no reason why such a contingency should be ruled a priori impossible.

Whatever one might feel about this argument in light of some of the more exotic 'machines' proposed by philosophers to substantiate it, the crucial issue as far as Wittgenstein's attack on Turing's thesis is concerned is that it overlooks a fundamental shift from the monkey to the Turing machine example. And it does so because it arises from a picture which distinguishes between observed behaviour (giving correct answers) and the unobserved 'events' which cause the former to occur (the 'mental states' of human or animal calculation and the physical states of the Turing machine). The argument presented at §9

assumes that the answer to the question 'How did *x* arrive at the correct answer?' consists in a specification of these 'mental/physical states'. The problem with this premise which Wittgenstein seized on has nothing to do with the crudity of the actual theory of either state which Turing developed. Rather, it is the point emphasized in Book VII, §42 of *Remarks on the Foundations of Mathematics* (quoted above) that whether or not the chimpanzees are following a rule has nothing to do with what went on in their minds. Hence the answer to the question 'How did *x* arrive at the answer?' is a question about the rules that were used. It is an agent's ability to manifest or explain the rules he/she/it is following which govern our description of him/her/it as *calculating*. But the answer to the question 'How did the Turing machine arrive at the result?' is indeed satisfied by Turing's account of the mechanics of the system in §§1–2 (together with an account of some particular program in §§3f). This is by no means the crux of Wittgenstein's objection, however; for Turing's answer to this point was that it is precisely because of the machine's ability to follow the subrules of the program that he was able to build up his picture of mechanical-calculation. And if the monkeys' ability to calculate depends on their ability to follow simple rules why should the picture be any different for the case of Turing machines?

To this idea Wittgenstein had objected (in a passage which clearly harks back to Lecture XX of *Lectures on the Foundations of Mathematics*):

> "This calculus is purely mechanical; a machine could carry it out." What sort of machine? One constructed of the usual materials—or a super-machine? Are you not confusing the hardness of a rule with the hardness of a material? ([49] III, §87)

But it is not at all clear how such a charge could be levelled at Turing. For Turing's point was that to say 'This calculus is purely mechanical' means that the rules of calculation have been broken into a series of meaningless subrules each of which is devoid of cognitive content and for that reason are such that 'a machine could carry it out'. Here, finally, is the level of the argument at which we can begin to see why Wittgenstein had insisted that 'Turing's machines are really *humans* who calculate': a theme which significantly is picked up—purged of any explicit reference to Turing—at [49] IV, §20: 'If calculating looks to us like the action of a machine, it is the *human being* doing the calculation that is the machine'.

From the foregoing we know that Wittgenstein was not sanctioning here the picture of 'human calculation' Turing offered in 'On Computable Numbers'. But to see what he was after we have to consider the notion of algorithms—of effective calculation—which Turing had inherited from Hilbert, and which was indirectly responsible for the arguments contained not just in §9, but in the post-1941 mechanist development of Turing's thought on the basis of that Thesis. For Turing felt, not just that the Mechanist Thesis was entailed by his analysis of computability, but that the latter was itself the inexorable consequence of Hilbert's approach to the *Entscheidungsproblem* [39,40]. The whole force of 'On Computable Numbers' derived from the fact that it was seen as the culmination of those very themes which had originally sparked off the development of recursion theory. Whether or not Turing had succeeded in clarifying the

nature of effective procedures, this much at least was agreed: that Turing had succeeded in explicating the notion of computability in terms of mechanical procedures. It is to the framework of Turing's thought that we must look, therefore, if we are to grasp the thrust of Wittgenstein's remarks.

2 Church's convention It was stressed in the preceding section that Wittgenstein viewed the epistemological element suffusing 'On Computable Numbers' as not just independent from but, more importantly, as a distortion of its mathematical content. Indeed, it was imperative that Wittgenstein see Turing's argument in these terms; otherwise he could not—in light of his insistence that the philosopher has no business meddling in the internal affairs of the mathematician—have justified his own involvement with Turing's Thesis. The most basic principles of his approach to the philosophy of mathematics demanded that the epistemological thread in Turing's argument be severed from the mathematical (see [33], Chap. I and V). To sustain this position demands, however, a far deeper involvement in the conceptual foundations of Turing's thought than has so far been explored, for the key to the almost immediate success of Turing's thesis lay in the themes which preceded it. Hilbert's original attempt to reduce transfinitary mathematical truths to finitary was but a reflection of his basic epistemological premise that the human mind is bound by its finite limitations: an idea which naturally governed his approach to the *Entscheidungsproblem*. This is manifested in his twin demands that the processes employed in any computation must be fixed in advance and that every computation terminate in a finite number of steps (see [43], pp. 75–76, 176). How, then, can one ignore the epistemological basis of subsequent efforts to provide a rigorous analysis of effective procedures?

In a sense, however, this would appear to be precisely what Church attempted. Insofar as one can judge from the terms in which he formally presented his thesis,[6] Church's interests were exclusively function-theoretic. From Kleene and Rosser's histories of the development of Church's thought (see [21] and [29]) we know that in his early work on the lambda-calculus (around 1930) Church's goal was to construct analogues of the integers and the algorithms that can be performed on them using lambda-functions. By 1932 he had clarified that lambda-definable functions must be effectively (i.e., algorithmically) calculable.[71] In 1933 he began to speculate that the opposite might also hold: viz., that the lambda-definable functions are *all* the effectively calculable functions. In his first announcement of the thesis (in 1935) he suggested that every effectively calculable function is lambda-definable. Following Kleene's demonstration of the equivalence of lambda-definability and recursiveness Church shifted the emphasis onto the latter. Most significantly of all, he presented his 1936 version of the argument as a definition, not a conjecture:

> We now define the notion . . . of an *effectively calculable* function of positive integers by identifying it with the notion of a recursive function of positive integers (or of a lambda-definable function of positive integers). This definition is thought to be justified by the considerations which follow, so far as positive justification can ever be obtained for the selection of a formal definition to correspond to an intuitive notion. ([6], p. 100)

What follows is somewhat confusing, however, for his justification is that there is no known recursive function on the positive integers for which there is not an algorithm to calculate its values, and conversely, that there is no known example of a function for which we possess an algorithm that is not recursive. The problem with this argument is that, strictly speaking, these are reasons for regarding CT as a valid conjecture: not a constructive definition. The kind of justification appropriate for the latter would be one which drew attention to the consequences of adopting such a definition; not reasons for accepting its 'truth'. Indeed, Church's format forces us to conclude, not that it is unlikely, but rather, that it is logically impossible to discover a function on the primitive integers for which there is an algorithm but which is not recursive; any suggestion to the contrary must on these terms be regarded, not as (contingently) false but rather, as *meaningless*. But if this was indeed his intention then it was misleading to speak of the thesis as a *definition*: for this suggests an element of synonymity which is foreign to Church's design.[8] Rather than pursuing an axiomatic definition of effective calculability Church had, in fact, presented CT as a basic axiom of recursion theory. For as has frequently been observed, Church's argument can best be seen in conventionalist terms, where the inference from 'ϕ is effectively calculable' to 'ϕ is recursive' is stipulated, and Church's justification can best be understood as a partial attempt to defend this piece of conceptual legislation on the (negative) grounds that the possibility of encountering anomalies seems remote.[9]

The important point to bear in mind in all this is that Church's proposed 'definition' was of 'effectively calculable function on the positive integers'. Far from representing an attempt to analyse a problematic epistemological concept Church's sole intention was to delimit the range of number-theoretic functions for which there are algorithms (cf. [21], p. 48). That is, Church sought to employ the notion of recursiveness as the criterion for what should henceforward be deemed 'effectively calculable functions'; should a novel method lead outside the class of general recursive functions 'the new function obtained cannot be considered as effectively defined ([19], p. 320). And the crux of Wittgenstein's remarks on Turing's thesis turns on a clear understanding of the logical status of this *cannot*. To paraphrase what Wittgenstein says in a related context: 'Here the word "cannot" means *logical impossibility* whose expression is not a proposition but a rule of syntax. (A rule delimits the form of description)' ([50], p. 241). For

> So much is clear: when someone says: "If you follow the *rule*, it *must* be like this", he has not any *clear* concept of what experience would correspond to the opposite.
>
> Or again: he has not any clear concept of what it would be like for it to be otherwise. And this is very important. . . . For the word "must" surely expresses our inability to depart from *this* concept. (Or ought I to say "refusal"?) ([49] IV, §30)

For this reason, the certainty that there could not on Church's version of CT be a nonrecursive effectively calculable function is logico-grammatical, not inductive. It is *use* which renders this version of CT 'unshakably certain': not 'the manner in which its truth is known'. Even to speak of the 'truth' of CT

(without drawing attention to the normative basis in which the term should here be understood) is potentially misleading, insofar as it invites the confusion that CT is a hypothesis. But whatever arguments Church might have employed to defend the adoption of his convention would have had no bearing on the logical status of CT itself; for, unlike a hypothesis, it is impossible to doubt the 'truth' of Church's rule of inference: not because it is *irrefutable*, but rather, because the possibility of doubt has been *logically excluded*.[10] Hence Wittgenstein would have agreed with the letter if not the spirit of Kleene's claim that CT 'excludes doubt that one could describe an effective process for determining the values of a function which could not be transformed by these methods into a general recursive definition of the function' ([19], p. 319).

In which case it is misleading even to refer to CT as a *thesis*: for all this represents the fruits of logico-grammatical clarification, not epistemological justification. When Kleene first baptized CT as such it was on the grounds that 'such functions as have been recognized as being effectively calculable (effectively decidable), and for which the question has been investigated, have turned out always to be general recursive' ([18], p. 274). But, as Kleene recognized, Church's conventionalist outlook renders such an argument strictly heuristic. Hence Kleene immediately shifted from mathematical to pragmatic considerations in his subsequent appeal that 'If we consider the thesis and its converse as definition, then the hypothesis is an hypothesis about the application of the mathematical theory developed from the definition' ([18], p. 274). Kleene's overriding concern was, of course, that CT 'cannot conflict with the intuitive notion which it is supposed to complete'. But it is not at all clear, given the above arguments, how it could. If *ex hypothesi* there was no pre-existing mathematical notion of an effectively calculable function then there was nothing with which Church's new rule, and the totality of functions thereby defined, *could* conflict. But even had this not been the case, Church's argument would simply have constituted an attempt to persuade mathematicians to *redefine* their terms. On neither account would CT assume a (quasi-) empirical aspect, for it was its designated role *qua* rule of inference which determined its logical status, not the arguments summoned in support of that mandate. The problem with Church's approach, however, was that he offered no explanation for the fact that effectively calculable functions are *effective procedures* in the sense envisaged by Hilbert; on the contrary Church had merely recapitulated Hilbert's original premise that the latter are algorithms. What was missing was some sort of elucidation of this prior assumption; and it was precisely this lacuna which Turing filled in 'On Computable Numbers'.

The key to the success of Turing's version of CT was that: (i) he offered an 'analysis' of algorithmically calculable functions which (ii) emerged rather than departed from Hilbert's framework in such a way that (iii) the notion of effective procedures assumed a wider epistemological import.[11] It was on the basis of this last point that Turing was to insist in his 'Lecture to the London Mathematical Society' that just as his computability thesis had evolved from Hilbert's formal systems so too the Mechanist Thesis is entailed by the premise that mechanically calculable functions are Turing machine computable. That does not mean that Turing intended his version of CT to be read in mechanist terms, however; only that one is forced *nolens volens* into such a thesis by the

demonstration that algorithms are mechanically calculable. For as has been emphasized by those who wish to treat CT as a foundation for Cognitive Science, there is no suggestion in 'On Computable Numbers' that Turing machines demonstrate or possess cognitive abilities; on the contrary, Turing was to stress that 'machine intelligence' only emerges in the shift from 'brute force' to 'learning' programs. The crucial point here is that each of the 'instructions' of the latter demands the same noncognitive execution as the former; in Dennett's words, the 'atomic tasks' of the program are putatively such that they 'presuppose *no* intelligence' ([10], p. 83). It is rather the overall complexity of the program built up out of these 'mechanical subrules' which, as Turing saw it, forces one 'to admit that [since] the progress of the machine had not been foreseen when its original instructions were put [i]t would be like a pupil who had learnt much from his master, but had added much more by his own work. When this happens I feel that one is obliged to regard the machine as showing intelligence' ([3], pp. 122–123) (*infra*).

The transition in Turing's thought was closely tied to the gradual shift in his interests from recursion theory to (what McCarthy was shortly to call) Artificial Intelligence. Given the guiding spirit behind the development of the former—to show that all number-theoretic functions are recursively calculable—Turing felt obligated to show that all (effective) number-theoretic functions are mechanically calculable. Hence he set out to prove that his mechanical computer was every bit as powerful as a mathematical (i.e., human) 'computer'. This reflects the significance of his emphasis on the term *computer*: of the fact that his attention in 'On Computable Numbers' was very much confined to computation. Even in his major post-war work (the 1946 'Proposal for Development in the Mathematics Division of an Automatic Computing Engine' and the 1947 London lecture) he was still primarily concerned with the mechanization of mathematical procedures: the problem of determining the 'scope of the machine', which he naturally approached within function-theoretic parameters ([38], pp. 38f). Indeed not just Turing's but virtually all early interest in computers was focussed on the development of machines to facilitate brute force approaches to complex computational problems. In 1937 Turing set out to construct a machine to help him solve Riemann's Hypothesis by calculating values of the zeta-function until an exception had been found.[12] In America Wiener and von Neumann were similarly interested in using a machine to calculate the value of the Reynolds number. And, of course, the war accelerated interest in such machines in those areas which demanded enormous computations; not just the obvious example of cryptanalysis, but also artillery (the development of ENIAC), radar, meteorology, and towards the end of the war the problems of implosion and development of the hydrogen bomb.

From numerous memoirs on the origins of digital computers we know that the general awareness that computers could be employed for more than just number-crunching—i.e., that they could be used for general symbol-manipulation—occurred some time during 1952–1954.[13] But at least ten years before this Turing had begun to speculate on a problem from which Artificial Intelligence was rapidly to emerge. From various sources that he has pieced together Andrew Hodges has demonstrated that by 1944 Turing was primarily interested in 'the construction of a universal computer and [in] the service such a machine might

render to psychology in the study of the human brain' ([41], p. 294). Apart from such instrumental factors as his meetings with Shannon[14] the problem itself which led Turing into these domains was a paradigm of formalist thought. According to Hodges, Jack Good and Turing began discussing (in 1941) the

> question of whether there was a 'definite method' for playing chess—a machine method, in fact, although this would not necessarily mean the construction of a physical machine, but only a book of rules that could be followed by a mindless player—like the 'instruction note' formulation of the concept of computability. ([16], p. 211)[15]

By 1943 Turing was seriously involved in the construction of a 'paper machine' (Turing's 'slave player') which was unlike the earlier Turing machines insofar as chess, according to Turing, 'requires some intelligence' to be played at all adequately. In 1936 Turing had insisted that the essence of Turing machines is that they require no intelligence to follow their rules. By 1946 Turing was exploring 'a very large class of non-numerical problems that can be treated by the calculator. Some of these have great military importance, and others are of immense interest to mathematicians' ([3], p. 41). Chief amongst the latter was chess:

> Given a position in chess the machine could be made to list all the 'winning combinations' to a depth of about three moves on either side. This . . . raises the question 'Can the machine play chess?' It could fairly easily be made to play a rather bad game. It would be bad because chess requires intelligence. We stated at the beginning of this section that the machine should be treated as entirely without intelligence. There are indications however that it is possible to make the machine display intelligence at the risk of its making occasional serious mistakes. By following up this aspect the machine could probably be made to play very good chess. ([3], p. 41)

Turing found in chess both the essence of the formalist infrastructure underlying 'On Computable Numbers' and the materials for the mechanist superstructure which he sought to erect on this foundation. It represented the categorial leap from brute force to self-modifying algorithms, paving the way for a shift from the crude picture of 'On Computable Numbers' where no intelligence was required of the Turing machines to one in which the machine would have to be able to *learn* from previous positions in order to improve its quality of play. It was the latter argument which led him to introduce the notion of machine intelligence the following year in his London lecture and then promote a full-blown Mechanist Thesis in 1950 in 'Computing Machinery and Intelligence'. 'On Computable Numbers' thus stands out as a turning-point, not just in Turing's thought, but through his influence, on the mathematical passage from recursion theory into computer science and the philosophical transition into Artificial Intelligence.

There are numerous examples in the history of mathematics of impossibility proofs which in the process of closing off one branch of inquiry have served to open another. Rarely, however, has the transition effected led outside of mathematics proper into the realm of epistemology. But Turing could draw on no less a precedent than Gödel's second incompleteness theorem to support such a reading of his computability thesis.[16] The immediate problem his interpretation raises, therefore, is whether 'On Computable Numbers' does indeed operate as a transitional impossibility proof which as such provides a cognitive

foundation for the Mechanist Thesis. Nowhere could the underlying tension in Turing's extended thesis be more clear than in Gödel's response to 'On Computable Numbers'. As Davis has documented, Gödel 'insisted (as Church later reported to Kleene) that it was "thoroughly unsatisfactory" to *define* the effectively calculable functions to be some particular class without first showing that "the generally accepted properties" of the notion of effective calculability necessarily lead to this class. . . . [I]t was not until Turing's work became known that Gödel was willing to concede that this difficulty had been overcome' ([9], pp. 12–13). But why should Gödel have resisted an argument which, as Turing himself was to remark, bore strong resemblance to Gödel's own earlier approach? (see [37], p. 160 and cf. [12]). Even more to the point, why should Gödel have endorsed Turing's as opposed to Church's version of CT in the first place, and why should he have continued to support a thesis whose consequences he so strongly deprecated?[17]

Gödel's argument holds a special fascination for Wittgensteinians interested in Artificial Intelligence (cf. [26]), not only for the obvious reason that his attitude to mechanism seems to run parallel to Wittgenstein's—albeit for reasons which Wittgenstein was loathe to accept (see [34])—but even more intriguingly, because these sentiments were to lead Gödel remarkably close to endorsing Hilbert's faith that 'in mathematics there is *ignorabimus*': a theme which finds a striking echo in Wittgenstein's approach to the Decision Problem (see [33], Chap. III). But how important is the above caveat? In his 'Remarks Before the Princeton Bicentennial Conference' Gödel credited Turing with having succeeded 'in giving an absolute definition of an interesting epistemological notion' ([13], p. 84). As should now be clear this suggests a line of thought which departs significantly from Wittgenstein's. For this shifts the focus from Church's interest in the mathematical characterization of a class of functions to an epistemological problem about—in Turing's terms—the properties displayed by the computists engaged in the calculation of those functions. Hilbert's influence on Gödel is obviously evident here. In simplest terms an effective procedure would appear to be one that does not transcend our computational abilities. The fact that this carries with it a host of familiar sceptical problems (e.g., the questions of relativity, reliability, variability, etc.) would no doubt have struck Gödel as the inevitable concomitant of a profound epistemological issue ([33], Chap. IV). The real problem which all this raises, however, is whether, having taken this initial step, Gödel could have avoided the quandary in which he subsequently found himself vis-à-vis Turing's Mechanist Thesis (cf. [28], p. 408).

Interestingly, unlike the 'Note added 28 August 1963' to 'On Formally Undecidable Propositions' ([11], p. 195), in the 1967 postscript to 'On Undecidable Propositions of Formal Mathematical Systems', Gödel was careful to exclude the mechanist consequences that Turing had drawn from his version of CT. In the latter he began by crediting Turing with having realized an illuminating account of the notion of formal systems in terms of mechanical calculation:

> In consequence of later advances, in particular of . . . A.M. Turing's work, a precise and unquestionably adequate definition of the general concept of formal system can now be given. . . . Turing's work gives an analysis of the concept of "mechanical procedure" (alias "algorithm" or "computation procedure" or "finite combinatorial procedure"). This concept is shown

> to be equivalent with that of a "Turing machine". A formal system can simply be defined to be any mechanical procedure for producing formulas. ([12], pp. 369–370)

But he concluded with the warning: ('Note that the question of whether there exist finite *non-mechanical* procedures, not equivalent with any algorithm, has nothing whatsoever to do with the adequacy of the definition of "formal system" and of "mechanical procedure"' (p. 370)). Wang's record of Gödel's *obiter dicta* has enabled us to piece together the reasoning which underlay this parenthetical remark. Wang first tells us that

> The concept of a mechanical procedure is involved in the characterization of formal systems. It seems natural to ask what mechanically solvable problems or computable functions are. This is related to the rather popular question: can machines think? can machines imitate the human mind? ([42], p. 83)

This, of course, is precisely what Turing had asked in 'Computing Machinery and Intelligence'; overlooking the possibility that it is the assumption on which the question rests that should most concern the antimechanist, Wang continues:

> One often hears that in mathematical logic a sharp concept has been developed which corresponds exactly to our vague intuitive notion of computability. But how could a sharp concept correspond exactly to a vague notion? A closer look reveals that the sharp notion, often referred to as recursiveness or Turing computability, is actually not as sharp as it appears at first sight.

All this is offered by way of circumscribing the bounds of Turing's 'analysis' in order to thwart his mechanist ambitions. And this argument, Wang explains, grew out of Gödel's reaction to Turing's post-1941 interests:

> Gödel points out that the precise notion of mechanical procedures is brought out clearly by Turing machines producing partial rather than general recursive functions. In other words, the intuitive notion does not require that a mechanical procedure should always terminate or succeed. A sometimes unsuccessful procedure, if sharply defined, still is a procedure, i.e. a well determined manner of proceeding. Hence we have an excellent example here of a concept which did not appear sharp to us but has become so as a result of a careful reflection. The resulting definition of the concept of mechanical by the sharp concept of 'performable by a Turing machine' is both correct and unique.

At this point we are clearly still operating within the parameters of recursion theory. The crux of this issue is that 'A formal system is nothing but a mechanical procedure for producing theorems. The concept of formal system requires that reasoning be completely replaced by "mechanical operations" on formulas in just the sense made clear by Turing machines' ([42], p. 84). So much makes clear both why Gödel was persuaded to accept CT on the basis of Turing's thesis and also why he repudiated the consequences which Turing was to draw. The former question is explained by Gödel's recognition of the epistemological aetiology of Turing's argument; the latter by his objection to Turing's philosophical

embellishments. But the question is whether, having agreed with (i) and (ii), Gödel was entitled to reject Turing's reading of (iii).

As far as Gödel was concerned Turing had only succeeded in analysing the concept of formal systems as mechanical procedures. After all, in his 1947 lecture Turing vaguely claimed that he had succeeded in giving a precise definition of a 'rule of thumb' procedure in terms of mechanical calculability. The issue has thus devolved onto the debate whether Gödel's objection that a distinction must be drawn between formal systems or algorithms and effective procedures is warranted. Wang insists:

> What is adequately explicated [by Turing] is the intuitive concept of mechanical procedures or algorithms or computation procedures of finite combinatorial procedures. The related concept of effective procedures or constructive procedures, meaning procedures which can in the most general sense be carried out, suggest somewhat different elements which are related to the difference between mental and mechanical procedures. ([42], p. 89)

In other words, Gödel identified mechanically with humanly effective procedures so far as finite combinatorial procedures are concerned. The former are those cases where the computation terminates (i.e., partial recursive functions). And, of course, the Halting Problem had established the existence of a class of functions where we do not know whether or not the program will terminate (Gödel's general recursive functions). But humanly effective procedures, according to Gödel, must always be capable of termination. It is not enough, however, to say that his reason for this relied on the assumption that humanly effective procedures must be finitely bound. Rather, Gödel seemed to share Hilbert's belief that to grasp a problem is to know in advance that there must be an answer; something which, according to the Halting Problem, cannot be guaranteed to Turing machines (*infra*).

Wang depicts Gödel's position in terms of a *categorial* difference between mechanically and humanly effective procedures. Yet he offers no grounds for such a demarcation, other than the puzzling remark that

> The related concept of effective procedures or constructive procedures, meaning procedures which can in the most general sense be carried out, suggests somewhat different elements which are related to the difference between mental and mechanical procedures and the question as to the method by which a Turing machine or a set of equations is seen to be one which defines a Turing computable or general recursive function. ([42], p. 89)

As it stands the argument is hardly convincing. For one thing it is presented as a dogmatic thesis, with no reason offered why effective procedures should be characterized by the criterion that we must know in advance whether or not they terminate. But in the Gibbs lecture (as quoted by Wang) Gödel appears to be interested in the slightly different objection that what is involved here is not so much a categorial as a qualitative difference: viz. humanly effective procedures *transcend* the capabilities of mechanical. Turing's argument, according to Gödel, presents the 'following disjunction: Either the human mind surpasses all machines (to be more precise: it can decide more number theoretical questions than any machine) or else there exist number theoretical questions undecidable for the human mind.' Gödel's response was to regard 'Hilbert [as] right in reject-

ing the second alternative. If it were true it would mean that human reason is utterly irrational by asking questions it cannot answer, while asserting emphatically that only reason can answer them' ([42], pp. 324–325).

To this argument Judson Webb responds that it is simply irrelevant, insofar as the parallels between effective and mechanical procedures are concerned, whether or not we know in advance that they will terminate. For

> whether a procedure literally 'can in the most general sense be carried out' does not depend on whether or not its termination can be constructively guaranteed in advance, *but only on the execution of its atomic tasks.* If I can actually follow some procedure, however general, step by step, this would surely not be changed by forgetting that termination had been constructively proved. . . . Why should mental procedures require any more assurance of their termination than mechanical ones? ([43], pp. 224–225)

Indeed, Webb argues that, far from being a liability, the Halting Problem is the saving feature of the Mechanist Thesis. Turing's undecidable sentences he describes as the 'guardian angels of computability theory' ([43], pp. 202, 208). Without them the mechanist would be confronted with a strongly determinist account of thought; but the upshot of the Halting Problem is that to describe a Turing machine as rule-governed is not at all to say that we can predict for all Turing machines how they will terminate. In fact, as Turing was to emphasize in his account of mechanical chess, the whole point of learning programs is that it is often impossible to foresee how they will evolve (see [3], pp. 122ff). One may thus sympathize with the spirit of Gödel's stand, which harks back to the inspiring theme of Hilbert's Paris lecture that there is no problem in mathematics, however intractable it might at first appear, which cannot be resolved if only the right point of view from which to attack it is discovered. The question is whether Gödel provided any reason to deny a priori the possibility of a self-modifying system's realizing a similar ability.

Whatever one might feel about the strongly mechanist conclusions which Webb draws from this objection, it must be conceded that he has successfully clarified what is perhaps the crucial feature of Gödel's argument. According to Gödel *thought* and *mechanical calculation* are partially co-extensive (viz. for the case of partial recursive functions); they diverge because of the epistemological constraints imposed on the latter (i.e., the premise that algorithmic calculability must be finitely bound) and the unbounded possibilities open to the former. But even before we can address the subtleties of the conflict created by a constructivist picture of mechanical computation versus a platonist conception of mathematical thought, we are left with the problem contained in Gödel's initial premise: does it even make sense to say that thought and mechanical calculation could be *partially co-extensive*? In conceding this had Gödel already admitted the very premise which landed him in the dilemma from which he could find no other alternative but to look for an escape route via epistemological obscurity? Did the very reason Gödel embraced Turing's version of CT induce him to distort the real—mathematical—significance of the latter? These are very much the questions which form the starting-point for Wittgenstein's approach to Turing's thesis. For the real challenge which Turing posed lay in the framework which he augmented; and it was precisely this element which, unlike Gödel,

Wittgenstein tackled head-on. His immediate objectives were, as we shall now see, to clarify whether epistemology has anything to do with the difference between *mechanical* and *effective* procedures, and whether Turing's post-1941 development of the Mechanist Thesis represents a misinterpretation of his 1936 results, the seeds of which were indeed sown in this 1936 presentation of his mechanical version of CT.

3 The foundations of Turing's thesis In the cosmology of Artificial Intelligence the higher cognitive forms only appeared when heuristic systems emerged from their primitive algorithmic origins. In the words of one of the field's founding fathers:

> Heuristic problem-solving, when successful, must, obviously, be rated as a higher mental activity than the solving of problems by some more or less automatic procedure. We are, therefore, probably justified in attaching the label of artificial intelligence to machine methods and machine programs that make use of heuristic procedures. ([30], p. 14)

To do justice to such an argument it must obviously be read against the background of formal learning theory. For if 'learning is any change in a system that produces a more or less permanent change in its capacity for adapting to its environment' ([35], p. 118) it would follow that there is no categorial difference between 'biological learning systems' and 'ideal learning machines'. But insofar as the mastery of a rule must constitute the foundation of any theory of learning, and given the striking fact that formal learning theory is itself predicated on Turing's thesis (see [27]), the claim that our chief concern here should be with the nature of algorithms is not without considerable force (see [32], pp. 103ff).

Certainly there can be no denying the paramount importance of algorithms for the evolution of computer science and thence AI. Donald Knuth confides that his

> favorite way to describe computer science is to say that it is the study of *algorithms*. An algorithm is a precisely defined sequence of rules telling how to produce specified output information from given input information in a finite number of steps. A particular representation of an algorithm is called a *program*. . . . Perhaps the most significant discovery generated by the advent of computers will turn out to be that algorithms, as objects of study, are extraordinarily rich in interesting properties and, furthermore, that an algorithmic point of view is a useful way to organize knowledge in general. ([23], p. 38)

Elsewhere he explains that

> an algorithm is a set of rules or directions for getting a specific output from a specific input. The distinguishing feature of an algorithm is that all vagueness must be eliminated; the rules must describe operations that are so simple and well defined they can be executed by a machine. ([22], p. 63)

This orthodox account will no doubt strike many as unproblematic but it does, in fact, present four worrying points. First, it is not at all clear what it means to say that 'rules describe operations'. The hallmark of a rule is that it *describes*

nothing; rather, it fixes the uses of concepts. Nor is it clear what 'simple' means here, or why it matters. Also, Wittgenstein showed that while it may be impossible to remove all possibility of vagueness or ambiguity from a rule (no matter how precisely it is formulated) that does not undermine the institution of rule-following per se (cf. [46], §§143ff). Finally, Knuth does not explain why the simplicity and exactitude of a rule are crucial to its machine execution: is this an allusion to Turing's premise that it is this criterion which enables a machine to follow these rules, or is Knuth merely drawing attention to the property of 'binary encodability' that was considered in the first section?

Inconsequential as these questions might appear, there is a danger that they are the symptoms of a lingering confusion inherited from Turing. For as we shall see, the steps in Turing's programs do indeed behave like descriptions; but far from paving the way for Turing's mechanist interpretation, that is a reason for reconsidering what it means to describe them as 'subrules'. What matters for the moment, however, are simply the ideas that: (i) an algorithm consists of a set of rules which (ii) are all of roughly the same (trivial) complexity and (iii) together yield a specific output from a specific input. On this reading algorithms emerge as a special subset of the class of functions (where functions are seen as rules for mapping arguments onto values); what distinguishes them in function-theoretic terms are (i) and (ii). Whether or not this captures the essence of Hilbert's intentions, it is unlike Turing's approach insofar as there is no explicit claim here that the machine *follows the rules* (although this might be tacitly implied by the emphasis on the simplicity of the operations). To appreciate the full epistemological implications of Turing's conception of algorithms we must address this most basic assumption which lies at the heart of the argument presented at §9 of 'On Computable Numbers'.

As we saw in the preceding section, the essence of Turing's theory of machine intelligence is that this is a function of the complexity of the program which the computer follows rather than the individual steps of the algorithm. That is, the difference between Turing's 'slave' and 'learning' programs lies in the shift from fixed to self-modifying algorithms. In the former the Turing machine repeatedly performs the same elementary steps *ad infinitum* (or *finitum* as the case may be); in the latter it alters its program (e.g., by employing heuristic techniques which enable it to augment its knowledge-base and/or store of rules) and thence the range and sophistication of the tasks it can execute. It was this argument which, according to the mechanist interpretation, enabled Turing 'to face the fact that a "human computer" does need intelligence—to follow rules formulated in a language he must *understand*' ([43], p. 220). In order to provide a 'non-question begging' analysis of computation 'the smallest, or most fundamental, or least sophisticated parts must not be supposed to perform tasks or follow procedures requiring intelligence' ([10], p. 83). On Turing's argument the

> Instructions given the computer must be complete and explicit, and they must enable it to proceed step by step without requiring that it comprehend the result of any part of the operations it performs. Such a program of instructions is an algorithm. It can demand any finite number of mechanical manipulations of numbers, but it cannot ask for judgments about their meaning. ([4], p. 47)

In which case 'Turing's analysis succeeded just because it *circumvented* the problem of how a computer can understand ordinary language' ([43], p. 225). For without 'meanings' to deal with 'these atomic tasks presuppose no intelligence [and] it follows that a non-circular psychology of computation is possible' ([43], p. 220). Hence, the epistemological significance of Turing's version of CT lay in his demonstration that algorithms can be described as complex systems of meaningless subrules each of which can as such be applied purely mechanically.

The issue arising from this 'analysis of computation' which has captivated philosophers is whether *understanding* can be built up out of a mastery of the 'syntactical instructions' that man and machine both follow (cf. [31]). But the problem which concerned Wittgenstein lay in the prior assumption that by scanning, printing, and erasing symbols on its tape, a Turing machine displays its ability to follow these 'meaningless subrules'. There are two interlocking premises operating in this conception of algorithms which Wittgenstein scrutinized: first, that it is intelligible to speak of a species of *meaningless rule* and, second, that the concept of *mechanically following a rule* is unproblematic. A more subtle task that needs to be addressed first, however, is to clarify the relationship between mastering the subrules of an algorithm and grasping the algorithm itself. To illustrate the distinction between mechanically and humanly effective calculation Wang cites the example of a pupil learning how to apply a Euclidean algorithm correctly without comprehending its overall structure (i.e., without knowing why it gives the correct results):

> Giving an algorithm for solving a class K of problems and a problem belonging to K, anybody can solve the problem provided he is able to perform the operations required by the algorithm and to follow exactly the rules as given. For example a schoolboy can learn the Euclidean algorithm correctly without knowing why it gives the desired answers. ([42], p. 90)

This example does indeed draw attention to an important aspect of the problem: before we can consider the issues raised by the assumption that an algorithm can be calculated mechanically when the agent/machine does not understand why it gives the correct results (or what it means to understand 'why' in this context) we must first clarify what it means to say that a pupil has learned how to apply the Euclidean algorithm. Or in general, the conditions under which one would say that 'the pupil has grasped the rule', 'the pupil has applied the rule correctly'. Even Searle is prepared to concede that a machine can be programmed to apply 'formal' rules correctly; what he balks at is the idea that a machine can be programmed to learn or follow semantic rules. But this seemingly innocuous example highlights an important problem.

This is not the putatively 'sceptical thesis' which Kripke has culled from §§185–242 of *Philosophical Investigations*. Rather, it concerns the last line of the above passage: the supposition that 'a schoolboy can learn the Euclidean algorithm correctly without knowing why it gives the desired answers'. Here Wang credits the schoolboy with the very ability immediately denied him. For to learn how to apply an algorithm correctly involves more than consistently producing the right answers (i.e., as a result of grasping how to execute each of the substeps). In order to say of the schoolboy that he has grasped the rules of calculation we demand more than the set of his answers to license such a judgment.

If this were all that was involved in the ascription of normative behaviour then we would indeed be exposed to a sceptical problem. But the point of the remarks on rule-following examined in the first section is that the criteria for crediting someone with the mastery of a rule are far more complex; we place such a judgment against the background of an agent's ability to explain, justify, correct, etc., their actions *by reference to that rule.* And the crucial problem with Wang's argument is that *learning an algorithm* likewise means something more than mastering each of the subrules without grasping the overall pattern or function of these 'atomic tasks'. Someone who learned all the individual rules for the chess pieces without ever grasping that the point of the game is to mate their opponent's king would not yet have learnt how to play chess. Likewise we would not say that a pupil who, when no remainder was left, responded 'I do not know what to do next', had mastered the rules of division. So too, all one can say in Wang's example is that the schoolboy has learned how to apply what for him is a set of *independent* rules; but to learn how to apply each of these 'atomic rules' does not in itself amount to learning the algorithm.[18]

This still leaves untouched the problem of what is involved in speaking of the schoolboy as learning how to apply each of the subrules of the algorithm. The first thing to notice here is that exactly the same criteria apply to the pupil's mastery of each of the subrules *qua* rules as apply to the algorithm itself. That is, to speak of *mastering a subrule* entails the possession of the abilities outlined above. Hence the simplicity versus complexity of a rule has no bearing on the normative cluster of concepts which underpins rule-following. Clearly Turing's argument must somehow be seen to overcome this obstacle, and this brings us up against one of the two major premises in the argument: the idea that there is such a thing as a 'meaningless subrule'. The obvious question one wants to ask here is, what sort of rule would a *meaningless rule* be: a rule which by definition stipulates nothing? *Prima facie* this looks like a blatant contradiction in terms, in which case one wants to know how such a radical thesis could have been so readily and uncritically accepted.[19] Of the many factors involved in the reception of Turing's thesis, there are two which particularly stand out. The first and undoubtedly most important derived from Hilbert's conceptual framework. As we saw in the preceding section, Gödel maintained that

> The concept of formal system requires that reasoning be completely replaced by 'mechanical operations' on formulas in just the sense made clear by Turing machines. More exactly, a formal system is nothing but a many valued Turing machine which permits a predetermined range of choices at certain steps. The one who works the Turing machine can, by his choice, set a lever at certain stages. This is precisely what one does in proving theorems within a formal system. ([42], p. 84)

Ironically, Wittgenstein would have agreed with the opening premise of this argument, but significantly, not at all in the manner intended.

Gödel's point was that the concept of a 'formal system' entails that of Turing's 'mechanical operations'; that Turing machines manifest the mechanical essence of the symbolic transformations which characterize a formal system. Wittgenstein's response, however, would be that, if anything, Turing machines manifest the *nonmathematical* character of 'formal systems'. This is quite a large

and important issue; among other things it demands careful consideration of the relationship between mathematical propositions, the well-formed-formulas of a formal system, and the manner in which Hilbert tried to bridge the gap between syntax and semantics thus created with so-called 'interpretations'.[20] For our purposes it suffices to note that the reason Wittgenstein would agree that 'The concept of formal system requires that reasoning be completely replaced by "mechanical operations" on formulas in just the sense made clear by Turing machines' is that the concept of *reasoning* is fundamentally incompatible with that of formal system.[21] The basic problem is that it is completely misleading to speak of a 'mechanical deduction'; for the two notions operating here, inference and sign manipulation, cannot be used together. One can speak of comparing the orthography, size, etc., of meaningless marks, but not of *deducing* one string of meaningless marks from another. To grasp that q follows from p just is to understand the nature of the conceptual relationship between the meaning of q and p: to know that *p entails q*. Hence, to return to the passage on calculation examined in the first section, if someone were accidentally to press the knobs '25', '×', and '20' of a calculating machine and obtain the result 500 they would not thereby have *calculated* the product of 25 × 20. For

> You aren't calculating if, when you get now this, now that result, and cannot find a mistake, you accept this and say: this simply shows that certain circumstances which are still unknown have an influence on the result.
>
> This might be expressed: if calculation reveals a causal connection to you, then you are not calculating. . . . What I am saying comes to this, that mathematics is *normative*. ([49] VII, §61)

Secondly, there was the whole point behind Turing's depiction of algorithms as sets of 'meaningless subrules': viz., reducing recursive functions to sets of mechanically calculable components rendered it plausible to conclude that by executing the totality of these tasks the machine would be computing the functions. This comes out quite clearly in the many tributes that have been made to Turing's 'analysis' of the concept of *computation*. Even Wang concedes:

> The intuitive notion of an algorithm is rather vague. For example, what is a rule? We would like the rules to be mechanically interpretable, i.e. such that a machine can understand the rule (instruction) and carry it out. In other words, we need to specify a language for describing algorithms which is general enough to describe all mechanical procedures and yet simple enough to be interpreted by a machine. . . . What Turing did was to analyze the human calculating act and arrive at a number of simple operations which are obviously mechanical in nature and yet can be shown to be capable of being combined to perform arbitrarily complex mechanical operations. ([42], p. 91)

The most important thing to notice about this argument—and indeed, all of the tributes that have been paid to Turing's 'analysis'—is that it assumes the very point which it sets out to establish; and it does this by misrepresenting the nature of calculation *ab initio* by reducing it to a level where the difference between mechanically following a rule and following a 'mechanical rule' is blurred, thereby collapsing 'simple rules' into 'simple mechanical operations'.

There are three different points to be considered in the premise that an

algorithm breaks down the act of following a rule into a set of noncognitive instructions such that they could be followed by a machine. The first question to ask is: why speak of 'subrules' at all here? The answer is that, in order to make his reductionist argument effective, Turing was forced to make it type-homologous. As we shall see, the manner in which the program is presented certainly preserves this appearance. The initial step was thus to maintain that algorithms decompose rules into their normative components. In order to assess this argument we need to consider what these 'subrules' are: i.e., the relation between the subrules and the overall task executed by an algorithm. There is a tendency to suppose that algorithms decompose complex rule-governed operations into a series of utterly simple tasks. But an algorithm is not a *precise formulation* of a pre-existing rule; rather, it is a *different* set of rules from the original which it is intended to supplant. Davis' Doubling Program (see Note 18), for example, is completely different from squaring inasmuch as it employs addition and subtraction to bypass the need for multiplication. To be sure, the outcome of the program corresponds to the results yielded by the rules for squaring; but what matters here is how we learn and apply the two systems of rules. The fact that we can use the rules of addition and subtraction rather than multiplication for the purposes of doubling—as opposed to squaring—does not entail that when squaring we actually do (i.e., tacitly) employ these rules. The correct answer to 'How did you calculate that the square of 125 is 15,625?' is 'I multiplied 125 × 125'; not, 'I (or my brain, or my unconscious mind) added 125 to itself one hundred and twenty-five times'.[22] What is really involved here is simply the fact that one set of rules can prove to be far more efficient than another for different contexts/purposes.

Even more important is the fact that Turing's interpretation harbours a fatal tension. The decomposition cannot, *ex hypothesi*, be entirely homologous: the whole point of reducing rules to subrules is to minimize the normative element which characterizes the rule proper. But here the argument runs into two different obstacles. First of all, the simplicity of a rule must not be confused with the question of its semantic content. A typical rule in Davis' Doubling Program is:

> Go to step i if 0 is scanned.

But given that this is a genuine rule it is anything but *meaningless*. Granted, it is so simple that one can envisage an agent doing it mechanically after a short while, and that is the problem that will be looked at last. For the moment we need only see that, however simple this rule might be, it does indeed tell one what to do. The only way to remove its normative content is to treat it as a *description* of the causal events that occur in the mind/program of the computer which trigger off certain reactions. Thus Turing introduced the premise that 'the behaviour of the computer at any moment is determined by the symbols which he is observing, and his "state of mind" at that moment' ([36], p. 136). On this picture the computer's 'state of mind' is the causal intermediary between 'observed symbols' and subsequent action. But

> *There are no* causal connections in a calculation, only the connections of the pattern. And it makes no difference to this that we work over the proof in

> order to accept it. That we are therefore tempted to say that it arose as the result of a psychological experiment. For the psychical course of events is not psychologically investigated when we calculate. ([49] VII, §18)

Thus the normativity of calculation disappears altogether on Turing's account, and there is no basis whatsoever for referring to the steps in the program as a species of 'meaningless subrule'; rather they function as descriptions of the computer's putative hidden calculating machinery.

Among other things this 'analysis' would entail that, e.g., a thermostat was a rule-following device even though none of the normative concepts which transform mere causal regularity into rule-following behaviour apply.[23] The danger here is to suppose that because normative actions can be mapped on to causal sequences this signifies that the latter are somehow a 'representation' of the former. Were this the case then the way would indeed be open to speaking of physically 'embodied' rules. But the shift from *encoding* to *embodying* marks a categorial departure to causal domains from which there can be no return to normativity. In the former case we may still speak of an agent following the encoded rules (assuming a further mastery of the rules of the code); the point of the latter, however, is to suggest that an agent's actions are causally determined by, e.g., mental events that lie beyond his immediate awareness. But as we saw in the first section, Wittgenstein stressed that whatever might be going on in their minds is irrelevant to the issue of whether or not the monkeys are following a rule; that is solely a question of whether or not they use the rule as a paradigm for the regulation of their conduct.[24] Thus the reason Wittgenstein laid such emphasis on the normativity of calculation à propos Turing's thesis was to clarify that the relation between a computation and the results which conform with it is *internal*, whereas in Turing's mechanical example the relation between input and output is strictly *external*, not conceptual; an account of the program can only explicate why the machine produced its results: not whether or not these were correct. For only the rules of calculation can establish this, and it is for that reason that they are *antecedent* to the machine's operations. That is, they establish the criteria which determine when we shall say that the machine is performing properly or malfunctioning.

Turing was guilty, therefore, either of the illicit assumption that the concept of *following a rule* can be regarded as a cybernetic mechanism, or else of presenting the steps of a Turing machine program in completely misleading form. For the above 'instruction' certainly looks like a rule (i.e., like an *instruction*) but what it actually describes on this causal picture is how:

> A '0' activates the transit mechanism.

But then this has nothing whatsoever to do with rule-following; it simply shows how to break down a complex mechanical action—viz., registering twice as many '1s' as were originally configured—into its subcomponents. Moreover, the terms chosen here are entirely apposite, for the latter are indeed subject to 'breakdown', but not to negligence, mental lapses, or plain misunderstanding. Hence they are immune from error, but not because they are infallible, for to be capable of making mistakes once again *presupposes* rule-following abilities. It is not surprising, therefore, that Turing should have glossed over the conceptual demarcation involved here when he argued that

> the machine has certain advantages over the mathematician. Whatever it does can be relied upon, assuming no mechanical 'breakdown', whereas the mathematician makes a certain proportion of mistakes. . . . My contention is that machines can be constructed which will simulate the behaviour of the human mind very closely. They will make mistakes at times. . . .[25]

One cannot emphasize enough the importance of Turing's demonstration that, given their (binary) encodability, recursive functions are ideally suited to mechanical implementation. But to mechanize rule-governed actions is to substitute, not subsume. Even to speak of the latter operations as *calculations* only serves to distort the categorial shift which occurs when mechanical devices displace normative activities.[26]

To be sure, this objection proceeds from the concept of calculation as this existed *prior* to Turing. And in Davis' words:

> What Turing did around 1936 was to give a cogent and complete logical analysis of the notion of "computation". Thus it was that although people have been computing for centuries, it has only been since 1936 that we have possessed a satisfactory answer to the question: "What is a computation?" ([8], p. 241)

The picture of 'logical analysis' which inspires this interpretation—the idea that depth analysis reveals what speakers in some sense *really* understand when they grasp a concept—was, of course, one of the major themes that Wittgenstein attacked in the 1930s. By the time he encountered 'On Computable Numbers' he had thoroughly repudiated, not only the notion of tacit knowledge which underpins this theory, but even more important, the widespread misconstrual of conceptual legislation for 'philosophical analysis' which sustains it. (For the definitive exegesis on Wittgenstein's development, see [1].) These too are topics which bear heavily on Turing's interpretation of his version of CT, but for present purposes we need only consider the basis of Davis' claim. Davis continues:

> Turing based his precise definition of computation on an analysis of what a human being actually does when he computes. Such a person is following a set of rules which must be carried out in a completely mechanical manner. Ingenuity may well be involved in setting up these rules so that a computation may be carried out efficiently, but once the rules are laid down, they must be carried out in a mercilessly exact way. ([8], p. 243)

In other words, that an agent's ability to calculate hinges on the mechanical application of subrules, regardless of whether or not the latter are 'meaningless'. Indeed, some have even speculated that the subrules of the algorithm might be treated as meaningful in some unique categorial sense which would allow one to speak of symbolic systems as possessing their own intrinsic 'meanings' and hence, of 'brain symbols as having original meaning' ([15], p. 119). On this line of reasoning what matters is solely how the rules are followed, not the mysterious nature of their quasi-semantic profile.

Wittgenstein's response to this argument (the distortion it inflicts on the notion of meaning is another matter) was to return to the distinction between calculation and performing an experiment first broached in the investigation into the nature of proof ([49], I and cf. [33], Chap. IV). Since the basic theme of this

discussion—which is intended to contrast the normative character of calculation with the causal framework of conducting an experiment—has already been touched on (see Section 1) we shall only consider here Wittgenstein's further reaction to the picture of mental machinery which buttresses Turing's argument. After introducing the reader to Turing's computability thesis at the beginning of *Gödel, Escher, Bach*, Hofstadter remarks:

> Here one runs up against a seeming paradox. Computers by their very nature are the most inflexible, desireless, rule-following beasts. Fast though they may be, they are nonetheless the epitome of unconsciousness. How, then, can intelligent behaviour be programmed? Isn't this the most blatant of contradictions in terms? One of the major theses of this book is that it is not a contradiction at all. ([17], p. 26)

The route which Hofstadter follows is literally in the footsteps of Turing. For that reason alone the terms of the above passage should be scrutinized long before one considers his subsequent efforts to surmount the paradox; that at any rate is what Wittgenstein demanded from its forerunner. The first thing to notice is that it is not at all clear what the opening part states. What does it mean to describe a computer as 'desireless': something which, as it happens, has no desires, or rather something of which it is logically absurd to speak of desires? Secondly, Hofstadter assumes the very point which, according to Wittgenstein, is the crux of the issue: that computers are 'rule-following beasts'. We shall only look now, however, at the third element in this triad of assumptions: that the issue somehow hinges on a computer's lack of 'consciousness'. It is the same point that Davis is making, and almost certainly what Turing had in mind at §9, which was essentially that there can be mechanical analogues of a (human) computer's unconscious behaviour whilst mechanically calculating.

Wittgenstein did not deny the possibility of an agent's performing certain calculating steps mechanically; rather, he insisted that this has nothing to do with the issue of consciousness versus unconsciousness:

> One follows the rule *mechanically*. Hence one compares it with a mechanism.
>
> "Mechanical"—that means: without thinking. But *entirely* without thinking? Without *reflecting*. ([49] VII, §60)

This is a theme which recurs throughout Wittgenstein's remarks on rule-following (cf. e.g., the 'reading machine' argument at [46], §157). Clearly it must be possible to speak of following a rule mechanically; we do so all the time (e.g., in driving a car). The question is, why call such behaviour rule-following: what distinguishes it from genuine autonomic conduct? Suppose we were dealing with the latter; if an agent were asked why he ϕed, his answer (should he be capable of proffering one) would be in strictly causal terms.[27] An agent's ϕing only counts as rule-following, however, if he offers a rule as the *grounds* for his ϕing. For as the above passage makes clear the *esse* of rule-following lies in the logical possibility of reflection. Certainly when we calculate, many of the familiar rules are performed unreflectingly; but it is our ability to articulate these rules if called upon to do so which warrants our calling such behaviour calculation. The point here is not that to follow a rule demands that one be conscious of the

fact, therefore; rather, it is to be capable of justifying (correcting, explaining, etc.) one's past actions *by reference to the rule.*

For all of these reasons Wittgenstein indicated that the basic fallacy committed by Turing in §9 was to move from the indisputable simplicity of the subrules of an algorithm to the conclusion that such procedures are intrinsically mechanical. For the fact that we might follow such rules unreflectingly in no way licenses the inference that they are noncognitive and a fortiori such that a machine could *follow* them. To be sure, Turing presented a picture of algorithms in which 'ingenuity may well be involved in setting up these rules so that a computation may be carried out efficiently, but once the rules are laid down, they must be carried out in a mercilessly exact way.' But then exactly the same thing may be said of the law! Whether it should be treated as an empirical fact that all such rules (i.e., algorithms) can be so encoded as to be mechanically executed or as a convention (where a set of rules that was not binary encodable would not be called an 'algorithm'), all that matters here is that a mechanical sign-manipulator is no more a 'rule-following beast' than an abacus. Which returns us to Wittgenstein's insistence that 'Turing's machines are really *humans* who calculate'. As if to corroborate Wittgenstein's insight, Turing was later to write:

> It is possible to produce the effect of a computing machine by writing down a set of rules of procedure and asking a man to carry them out. Such a combination of a man with written instructions will be called a 'Paper Machine'. A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine. ([40], p. 9)

If this later argument seems to suffer from acute circularity it is entirely because of the pressures built into 'On Computable Numbers'. It was noted at the outset that, from Wittgenstein's point of view, 'On Computable Numbers' must be seen as a hybrid paper: it begins with a mathematical analysis of functions but then strays into a philosophical inquiry into the cognitive abilities of those who compute them. In the first part Turing provided a unique insight into the criterion implicit in earlier attempts to delimit the range of number-theoretic functions for which there are algorithms; in the second he shifted to a discussion of calculation which is pursued in quasi-epistemological, not mathematical terms. And as such it is open to Wittgenstein's objection, the crux of which is that the only way Turing could synthesize these disparate elements was by investing his machines with cognitive abilities *ab initio*: that is, by assuming the very premise which he subsequently undertook to defend.

The logical next step in this investigation would be to consider Turing's influence on those who were to tread in his footsteps: particularly in the areas of learning programs and automata studies. To conclude this prolegomenon, however, we have still to resolve why Turing should have presented these mechanical operations in quasi-cognitive terms in the first place? That is, why he prefaced his mathematico-logical achievement with the analogy that we may compare a Turing machine to a man in the process of computing? To dismiss §9 as nothing more than an example of the philosophical confusions that can arise when mathematicians venture into prose in order to interpret the significance of their results would be to abandon what is perhaps the most important feature of this entire issue: viz. the manner in which the inner dynamics of the

exercise upon which Turing embarked were responsible for his subsequent involvement in the Mechanist Thesis. For as has been stressed, Turing did not approach CT in order to inculcate the Mechanist Thesis; on the contrary the latter evolved from his version of CT, and it is for that reason that the foundations underlying Turing's thought are as significant as the content of his computability results. Indeed, the very controversy which continues to surround CT is proof of the enduring strength of that framework, and the problems obdurately tied to it. It was ultimately because formal systems were seen to demand an epistemological counterpart to the role which understanding plays vis-à-vis mathematical propositions that Church's convention was found wanting. And so it was that, because of the framework he inherited, Turing was led into the justification presented at §9, and his 'analysis' hailed as an absolute definition of an interesting epistemological notion'. Those who would build a computationalist empire on the basis of such troubling precedents would do well to pause first on the significance of Wittgenstein's ever-timely warning that 'one keeps forgetting to go right down to the foundations. One doesn't put the question marks *deep* enough down' ([51], p. 62e).

NOTES

1. Cf. the discussion of Russell's paradox in *Remarks on the Foundations of Mathematics* [49], I, App 3 and VII, and [33], Chap. VI.

2. An intriguing question for Wittgenstein scholars is: how far back can the RFM argument be traced? In particular: can any signs of it be seen in *Lectures on the Foundations of Mathematics* [48]? The problem is that this quotation comes from a 1947 typescript and given Wittgenstein's method of compiling his manuscripts it is impossible to say when it was first written. It seems likely that it would have been at the time of first reading Turing's paper, and given the later date of this typescript this might well indicate after the 1939 lectures. This seems to be substantiated by the fact that no such sentiment is expressed in [48] even though Wittgenstein touched on a number of areas which impinge on it; and since the argument underpinning the above passage is developed at length in RFM [49] this would confirm that Wittgenstein only read 'On Computable Numbers' after his 1939 lectures (which were concerned with the foundations crisis). This is not a terribly important issue, and there is probably no way of deciding it one way or another, but there are grounds to suggest that if not wrong this reading is at least seriously misleading. For the fact is that throughout *Lectures on the Foundations of Mathematics* Wittgenstein addressed a series of interrelated questions—the nature of proof, mathematical propositions, calculation, a calculus—which led him to attack near the end of his lectures the idea that there is such a thing as 'logical machinery' (Lecture XX). He was attacking here what he had earlier called the *Bedeutungskörper* conception of meaning: the Fregean idea that the meaning of a mathematical concept determines how it can be extended. Hence it is a crucial part of his larger attack on the notion that a proof or a rule compels us to proceed in a certain way. But what is particularly interesting is that, when he did address Turing's thesis in RFM, he presented his objections in the context of this earlier argument. And since the purpose of this earlier argument was to clarify the normative character of mathematical propositions and proofs it is clear that the same idea underpins his remarks on Tur-

ing's thesis. So it is highly likely that the arguments developed in the 1939 lectures were to some extent discreetly directed at Turing's thesis.

3. After he had introduced Turing machines in Section 1 he added: 'No real attempt will be made to justify the definitions given until we reach §9' ([36], p. 117).

4. It is interesting to note that this caution slips slightly in what follows:

> However, by altering its m-configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined by the m-configuration q_n and the scanned symbol G(r). This pair q_n, G(r) will be called the "configuration": thus the configuration determines the possible behaviour of the machine. ([36], p. 117)

Here 'remember' and 'behaviour' are without inverted commas but 'seen' (i.e., perception) retains them.

5. Turing's presentation of Turing machines in 'On Computable Numbers' shows exactly where his interests lay. He may have started off with the purpose of proving the impossibility of Hilbert's approach to the *Entscheidungsproblem*, but Turing's real interest was to clarify the nature of *effective procedures* in terms of the notion of *computability*. This will be discussed at greater length in Section 2.

6. And informally; see his 1935 letter to Kleene, quoted in Davis [9], p. 9.

7. The equivalence is Church's; cf. Church [6], p. 224 and Kleene [20], p. 56.

8. Certainly it is a suggestion which would meet with stiff resistance today, if for no other reason than the familiar examples of primitive recursive functions which are not effectively calculable. But the reasons it should not be attributed to Church are slightly different and, if anything, even more significant. Church did not claim that effective procedures really *are* lambda-functions; what he said was that they are lambda-definable. In other words, we can design a lambda-function which will do the same thing as any effectively calculable function. Church seems to have seen this as a species of 'translation': i.e. that we can map a lambda-function onto any kind of number-function. (The danger here is that of supposing that a recursive function π somehow *analyses* what we really understand by the number-theoretic function.)

9. This was the argument Church pursued in 'The Constructive Second Number Class'; he there defends his proposed definition of effective ordinals on the grounds that 'those who do not find this convincing the definition [sic] may perhaps be allowed to stand as a challenge, to find either a less inclusive definition which cannot be shown to exclude some ordinal which ought reasonably to be allowed as constructive, or a more inclusive definition which cannot be shown to include some ordinal of the second class which cannot be seen to be constructive' ([5], p. 224).

10. Suppose 'a mistake is not possible. But what kind of possibility is that? Mustn't mistake be *logically* excluded?' (Wittgenstein [45] §194; cf. Shanker [33], Chaps. I, II, III and VII passim).

11. So much is clear from Church's review of 'On Computable Numbers' and the account he gave of Turing's thesis in 'The Constructive Second Number Class' [6], (see p. 227; and cf. Davis [8]).

12. Turing was convinced the conjecture was false; in 1936 E. C. Titchmarsh had used a mechanical calculator to demonstrate that the first 104 zeros all lie on the real line;

Turing's plan was to examine the next few thousand zeros in the hope of coming across a counterexample to Riemann's Hypothesis.

13. See Hamming [14], pp. 8f; cf. also Knuth's explanation—which reads as a direct comment on the early founders of recursion theory—that

> computing machines (and algorithms) do not only compute with *numbers*: they can deal with information of any kind, once it is presented in a precise way. We used to say that sequences of symbols, such as names, are represented in a computer as if they were numbers; but it is really more correct to say that numbers are represented inside a computer as sequences of symbols. [23]

14. Hodges reports that:

> Shannon had always been fascinated with the idea that a machine should be able to imitate the brain; he had studied neurology as well as mathematics and logic, and had seen his work on the differential analyser as a first step towards a thinking machine. They found their outlook to be the same: there was nothing sacred about the brain, and that if a machine could do as well as a brain, then it *would* be thinking—although neither proposed any particular way in which this might be achieved. . . . Once Alan said at lunch, "Shannon wants to feed not just *data* to a Brain, but *cultural* things! He wants to play *music* to it!" And there was another occasion in [t]he executive mess, when Alan was holding forth on the possibilities of a "thinking machine". His high-pitched voice already stood out above the general murmur of well-behaved junior executives grooming themselves for promotion with the Bell corporation. Then he was suddenly heard to say: "No, I'm not interested in developing a *powerful* brain. All I'm after is just a *mediocre* brain, something like the President of the American Telephone and Telegraph Company". ([16], p. 251)

15. This is corroborated in Michie's memoir on 'Turing and the Origins of the Computer':

> The game of chess offered a case of some piquancy for challenging with irreverent shows of force the mastery which rests on traditional knowledge. At Bletchley park, Turing was surrounded by chess-masters who did not scruple to inflict their skill upon him. The former British champion Harry Golombek recalls an occasion when instead of accepting Turing's resignation he suggested that they turn the board round and let him see what he could do with Turing's shattered position. He had no difficulty in winning. Programming a machine for chess played a central part in the structure of Turing's thinking about broader problems of artificial intelligence. In this he showed uncanny insight. As a laboratory system for experimental work chess remains unsurpassed. But there was present also, I can personally vouch, a Turing streak of iconoclasm: what would people say if a machine beat a master? How excited he would be today when computer programs based on his essential design are regularly beating masters at lightning chess, and producing occasional upsets at tournament tempo!
>
> Naturally Turing also had to build a chess program (a 'paper machine' as he called it). At one stage he and I were responsible for hand-simulating and recording the respective operations of a Turing–Champernowne and a Michie–Wylie paper machine pitted against each other. Fiasco again! We both proved too inefficient and forgetful. Once more Alan decided to go it alone, this time by programming the Ferranti Mark 1 computer to simulate both. ([25], p. 35)

16. For a discussion of Wittgenstein's remarks on 'transitional impossibility proofs', particularly as this relates to Gödel's theorem, see [34], §1.

17. Davis concludes that Gödel found in Turing's thesis a satisfactory rendering of the 'generally accepted properties' of effective calculability of which he had complained

to Church in 1934 ([9], p. 14). This, while obviously true, is simply the problem which concerns us, not the answer.

18. For example, in the case of the Doubling Program which Martin Davis outlines in [8], pp. 246ff, we would want to say that someone has only mastered the program when they understand *why* it is called a 'doubling program' and how this relates to the shift from steps 9 to 10 when the program terminates.

19. Such an argument might be traced back to the Vienna Circle's treatment of tautologies as rules fixing the use of symbols (as in, e.g., the case of mathematical propositions). For here would be a standard and familiar example of one kind of 'meaningless rule'; at least, a (pseudo-)proposition devoid of all sense but which functions as a rule for the use of symbols. But this too Wittgenstein rejected. Tautologies themselves cannot be rules; if they were then all tautologies would stipulate the same thing: viz., nothing. Rather, it is how we use tautologies which is the rule, or to be more precise, that in calling certain constructions tautologies we formulate what are commonly referred to as the rules of thought. The tautology '$p \rightarrow p$' is manifestly different than the tautology '$\neg\neg p = p$'; by calling both constructions *tautologies* we register rules for the use of '$\rightarrow$' and '$\neg$' respectively. In *Lectures on the Foundations of Mathematics* there are long discussions on the nature of mathematical propositions *as opposed to* tautologies, and on the relation of the tautology '$p. \& p \rightarrow q. \rightarrow q$' to the rule of inference modus ponens. In a difficult section Wittgenstein warned that Russell mistakenly treated the former as the latter. But it is a tautology—i.e., devoid of all cognitive content; hence it could not itself serve as a rule of inference or a 'law of thought'. Rather it is *that* '$p. \& p \rightarrow q. \rightarrow q'$ *is a tautology* which is the law of thought. Russell, he explained, mistakenly wrote the Law of Excluded Middle as '$p \vee \neg p$', but the Law of Excluded Middle should be written as '"$p \vee \neg p$" = Taut.' For whereas '$p \vee \neg p$' says nothing, we use the rule of logical grammar '"$p \vee \neg p$" = Taut.' as one of the fundamental defining criteria for the meaning of 'proposition'.

20. In brief, Wittgenstein attacked Hilbert's idea that the concepts defined by the axioms of a system are without content until assigned an interpretation; i.e., that pure geometries 'define' and applied geometries determine the meaning of primitive concepts. To this Wittgenstein objected that 'the axioms of geometry have the character of stipulations concerning the language in which we want to describe spatial objects. They are rules of syntax. The rules of syntax are not about anything; they are laid down by us' ([50]). That is, the meaning of primitive mathematical concepts is constituted by the rules governing their use. In maintaining that pure geometry lays down the logical form of primitive concepts without understanding their meaning Hilbert had confused the *application* with an 'interpretation' of these concepts. Thus, where Hilbert regarded pure and applied geometry as independent of one another Wittgenstein responded that the essence of geometrical as well as arithmetical concepts is that they *appear in mufti*: just as $2 + 2 = 4$ whether we are adding pebbles or people, so too all the points on the circumference of a Euclidean circle must be equidistant from the centre whether we are measuring the spatial configurations of tables, chairs, or mugs (cf. [34], §4).

21. Cf. Haugeland's account of the 'paradox of mechanical reasoning':

> Reasoning (on the co[m]putational model) is the manipulation of meaningful symbols according to rational rules (in an integrated system). Hence there must be some sort of manipulator to carry out those manipulations. There seem to be two basic possibilities: either the manipulator pays attention to what the sym-

> bols and rules *mean* or it doesn't. If it does pay attention to the meanings, then it can't be entirely mechanical—because meanings (whatever exactly they are) don't exert physical forces. On the other hand, if the manipulator does not pay attention to the meanings, then the manipulations can't be instances of reasoning—because what's reasonable or not depends crucially on what the symbols mean. ([15], p. 39)

22. One suspects that herein lie the seeds of a perplexing problem: if multiplication really consisted in this kind of Turingesque algorithm we would be confronted with the mystery of how the brain was able to compute this sum so quickly. The answer, no doubt, would be that it must use parallel processing and optimal heuristics!

23. Needless to say, it did not take long for this argument to surface. Here too a serious study of the evolution of cybernetics and automata study is a prerequisite for clarifying the foundations of AI.

24. That is not to say that there is no such thing as 'calculating in your head'; only that such uses of the term are themselves derivative on the normative practices which constitute calculation (i.e., the abilities to explain, correct, etc., the answers arrived at. See Wittgenstein [52], I, §655; cf. §§96f, 649ff).

25. Alan Turing, 'Intelligent machinery, a heretical theory', quoted in Sarah Turing [41], p. 129.

26. One of the factors which most complicates this issue is the ambiguous nature of computer programs: itself a product of their evolution. In one respect a program can indeed be seen as a set of instructions (e.g., upwards to the user, and originally downward to machine operators when they were in service). With the development of compilers, however, the 'downward' function of the program has been restricted to that of a set of ciphers (cf. [32], pp. 88ff).

27. In other words, this argument brings us down to Wittgenstein's discussion of the logico-grammatical difference between reasons and causes. See [2], Chap. 4.

REFERENCES

[1] Baker, G. P. and P. M. S. Hacker, *Wittgenstein: Understanding and Meaning*, Basil Blackwell, Oxford, 1980.

[2] Baker, G. P. and P. M. S. Hacker, *Rules, Grammar and Necessity*, Basil Blackwell, Oxford, 1986.

[3] Carpenter, B. E. and R. W. Doran (eds.), *A. M. Turing's ACE Report of 1946 and Other Papers*, The MIT Press, Cambridge, Massachusetts, 1986.

[4] Chaitin, G., "Randomness and mathematical proof," *Scientific American*, vol. CCXXXII (1975).

[5] Church, Alonzo, "An unsolvable problem of elementary number theory," *American Journal of Mathematics*, vol. 58 (1936), reprinted in [7].

[6] Church, Alonzo, "The constructive second number class," *Bulletin of the American Mathematical Society*, vol. 44 (1938).

[7] Davis, Martin, *The Undecidable*, Raven Press, New York, 1965.

[8] Davis, Martin, "What is a Computation?", in *Mathematics Today*, ed., Lynn Arthur Steen, Springer Verlag, New York, 1978.

[9] Davis, Martin, "Why Gödel didn't have Church's Thesis," *Information and Control*, vol. 54 (1982).

[10] Dennett, Daniel, "Why the law of effect will not go away," in *Brainstorms*, The Harvester Press, Sussex, 1981.

[11] Gödel, Kurt, "On formally undecidable propositions of *Principia mathematica* and related systems I" (1931), in *Kurt Gödel: Collected Works*, vol. 1, eds., Solomon Feferman et al., Oxford University Press, Oxford, 1986.

[12] Gödel, Kurt, "On undecidable propositions of formal mathematical systems" (1934), in *Kurt Gödel: Collected Works*, vol. 1, eds., Solomon Feferman et al., Oxford University Press, Oxford, 1986.

[13] Gödel, Kurt, "Remarks before the Princeton bicentennial conference on problems in mathematics, 1–4" (1946), first published in [7].

[14] Hamming, R. W., "We would know what they thought when they did it," in *A History of Computing in the Twentieth Century*, eds., N. Metropolis, J. Howlett, and Gian-Carlo Rota, Academic Press, New York, 1980.

[15] Haugeland, John, *Artificial Intelligence: The Very Idea*, The MIT Press, Cambridge, Massachusetts, 1985.

[16] Hodges, Andrew, *Alan Turing*, Burnett Books, London, 1983.

[17] Hofstadter, Douglas, *Gödel, Escher, Bach: An Eternal Golden Braid*, Penguin Books, Middlesex, 1980.

[18] Kleene, Stephen C., "Recursive predicates and quantifiers," *Transactions of the American Mathematical Society*, vol. 53 (1943).

[19] Kleene, Stephen C., *Introduction to Metamathematics*, D. Van Nostrand, Princeton, 1950.

[20] Kleene, Stephen C., "Recursive function theory," *Annals of the History of Computing*, vol. 3 (1981).

[21] Kleene, Stephen C., "The theory of recursive functions, approaching its centennial," *Bulletin of the American Mathematical Society*, vol. 5 (1981).

[22] Knuth, Donald, "Algorithms," *Scientific American*, vol. CCXXXIV (1977).

[23] Knuth, Donald, "Computer science and mathematics," *Science*, vol. 194 (1976), reprinted in *Mathematics: People, Problems, Results*, vol. III, eds., Douglas M. Campbell and John C. Higgins, Wadsworth International, Belmont, California, 1984.

[24] Kripke, Saul, *Wittgenstein on Rules and Private Language*, Basil Blackwell, Oxford, 1982.

[25] Michie, Donald, *Machine Intelligence and Related Topics*, Gordon and Breach Science Publishers, New York, 1982.

[26] Neumaier, Otto, "A Wittgensteinian view of artificial intelligence," in *Artificial Intelligence: The Case Against*, ed., Rainer Born, Croom Helm, London, 1986.

[27] Osherson, Daniel N., Michael Stob and Scott Weinstein, "Ideal learning machines," *Cognitive Science*, vol. 6 (1982).

[28] Post, Emil, "Absolutely unsolvable problems and relatively undecidable propositions," in [7].

[29] Rosser, J. B., "Highlights of the history of the lambda-calculus," *Annals of the History of Computing*, vol. 6 (1984).

[30] Samuel, Arthur L., "Artificial intelligence: A frontier of automation," *The Annals of the American Academy of Political and Social Science*, vol. 340 (1962).

[31] Searle, John, "Minds, brains and programs," *The Behavioral and Brain Sciences*, vol. 1 (1980), reprinted in *Artificial Intelligence: The Case Against*, ed., Rainer Born, Croom Helm, London, 1986.

[32] Shanker, S. G., "The decline and fall of the mechanist metaphor," in *Artificial Intelligence: The Case Against*, ed., Rainer Born, Croom Helm, London, 1986.

[33] Shanker, S. G., *Wittgenstein and the Turning-Point in the Philosophy of Mathematics*, Croom Helm, London, 1987.

[34] Shanker, S. G., "Wittgenstein's remarks on the significance of Gödel's theorem," in *Gödel's Theorem in Focus*, ed., S. G. Shanker, Croom Helm, London, 1988.

[35] Simon, Herbert A., *The Sciences of the Artificial*, 2nd Ed., The MIT Press, Cambridge, Massachusetts, 1984.

[36] Turing, Alan, "On computable numbers, with an application to the *Entscheidungsproblem*," *Proceedings of the London Mathematical Society*, vol. 42 (1936–1937), reprinted in [7].

[37] Turing, Alan, "Systems of logic based on ordinals," *Proceedings of the London Mathematical Society*, vol. 45 (1939), reprinted in [7].

[38] Turing, Alan, "Proposal for development in the mathematics division of an automatic computing engine (ACE)," in [3].

[39] Turing, Alan, "Lecture to the London Mathematical Society on 20 February 1947," in [3].

[40] Turing, Alan, "Intelligent machinery" (1948), in *Machine Intelligence 5*, eds., B. Meltzer and D. Michie, Edinburgh University Press, Edinburgh, 1969.

[41] Turing, Sarah, *Alan Matheson Turing*, Heffers, Cambridge, 1959.

[42] Wang, Hao, *From Mathematics to Philosophy*, Routledge & Kegan Paul, London, 1974.

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

[44] Wittgenstein, Ludwig, *The Blue and Brown Books*, Basil Blackwell, Oxford, 1960.

[45] Wittgenstein, Ludwig, *On Certainty*, eds., G. E. M. Anscombe and G. H. von Wright, trans., Denis Paul and G. E. M. Anscombe, Basil Blackwell, Oxford, 1969.

[46] Wittgenstein, Ludwig, *Philosophical Investigations*, 3rd Ed., Basil Blackwell, Oxford, 1973.

[47] Wittgenstein, Ludwig, *Philosophical Grammar*, ed., Rush Rhees, trans., Anthony Kenny, Basil Blackwell, Oxford, 1974.

[48] Wittgenstein, Ludwig, *Lectures on the Foundations of Mathematics: Cambridge, 1939*, ed., Cora Diamond, The Harvester Press, Sussex, 1976.

[49] Wittgenstein, Ludwig, *Remarks on the Foundations of Mathematics*, eds., G. H. von Wright, R. Rhees, and G. E. M. Anscombe, trans., G. E. M. Anscombe, 3rd Ed., Basil Blackwell, Oxford, 1978.

[50] Wittgenstein, Ludwig, *Ludwig Wittgenstein and the Vienna Circle*, conversations recorded by Friedrich Waismann, ed., Brian McGuinness, trans., Joachim Schulte and Brian McGuinness, Basil Blackwell, Oxford, 1979.

[51] Wittgenstein, Ludwig, *Culture and Value*, ed., G. H. von Wright, trans., Peter Winch, Basil Blackwell, Oxford, 1980.

[52] Wittgenstein, Ludwig, *Remarks on the Philosophy of Psychology, Volume I*, eds., G.E.M. Anscombe and G. H. von Wright, trans., G. E. M. Anscombe, Basil Blackwell, Oxford, 1980.

Department of Philosophy
Atkinson College
York University
North York, Ontario M3J 1P3