Epistemological Investigations into the Foundations of Artificial Intelligence

Rainer P. Born, Linz

O. Hofstadter

After introducing the reader to Turing's version of Church's thesis, i.e. the thesis of computation, D. Hofstadter formulates the following questions as motivation for his book: "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."

[26, my emphasis]

In a recent article contained in an issue of the Notre Dame Journal of Formal Logic, entirely devoted to Church's thesis, S.G. Shanker analyses the development leading up to such statements as Hofstadter's and traces it back, respectively relates it, to Turing's original article. Shanker's thesis, which I would like to take up and develop further, is that the rudiments of Turning's later work on what has, since 1956, been called "Artificial Intelligence" (AI for short), are already contained in the 1936 article. This is also partially true for Turing's attempts to reach a definition of usage and

1) Precise wording of Church's thesis: "We now define the notion, already discussed, of an effectively calculable function of positive integers by identifying it with the notion of a recursive function of positive integers (or of I-definable functions 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." Church, Alonzo, 'An unsolvable problem of elementary number theory', American Journal of Mathematics 58 (1936), 345-363, reprinted in Davis, Martin, ed., The Undecidable, Hewlett, New York 1965, 88-107.


My considerations are deeply influenced by Shanker's article. I have tried to give explicit cross-references wherever possible. Nevertheless, my considerations should be considered as a possible answer to the fundamental question which Shanker poses at the end of his article.

4) Turing, Alan M., 'On Computable Numbers, with an Application to the Entscheidungsproblem', Proceedings of the London Mathematical Society, Second Series, 42 (1937), 230-265. The paper was delivered in 1936, and will therefore be referred to as Turing's 1936 paper.

5) As an informal/informative definition, I would like to use Feigenbaum's: "AI is that part of computer science concerned with designing intelligent computer systems, i.e. systems that exhibit the characteristics which we associate with intelligence in human behaviour - e.g. understanding language, learning, reasoning, solving problems etc." in Barr, A. and Feigenbaum, E. A., The Handbook of Artificial Intelligence, vol 1, Stanford 1981, 3.
therefore sensible employment of the term 'intelligence'. Shanker basically analyses the roots of the so-called 'mechanistic thesis' of the mind (often understood as functionalism in the sense of Hilary Putnam). The prospectus of Shanker's analysis, however, is that it still remains to investigate Turing's influence on those persons "who were to tread in his footsteps; particularly in the areas of learning programs and automata studies" [644]. One of the examples for Turing's influence is Hofstadter, especially with respect to his study of AI. In addition to this, Shanker stresses the importance of an analysis of the interconnection between prose and actual meaning respectively interpretation of scientific results, "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". [641]

The present paper aims to contribute to this very problem. Before I enter into further detail, however, let me return to the introductory quote from Hofstadter. In this quote, Hofstadter follows [more or less verbatim] in Turing's footsteps, although Turing never explicitly states that computers are "the epitome of unconsciousness". Nevertheless, according to Shanker, it is what Turing had in mind in § 9 of his 1936 paper, namely "that there can be mechanical analogues of a (human) computer's unconscious behaviour whilst mechanically computing" [640]. At the end of his paper, Shanker (in Wittgenstein's wake) attacks Hofstadter for assuming "the very point which, according to Wittgenstein, is the crux of the issue: that computers are 'rule-following beasts'" [640]. However, an attack (in the sense of Wittgenstein's attack on an epistemological problem in Turing's 1936 paper) at a philosophy like that should aim deeper. Basically, it depends on an analysis of Turing's paper, which is based on the identification of 'the mechanical following of rules' with 'the following of mechanical rules'.

8) Of course, it depends on the respective concept of philosophy whether, and how, one wants to attack Hofstadter's endeavour (the basis of H's endeavour). If we accept the working philosophy of computer scientists working in AI (as emotive force), then we may accept Hofstadter's questions as genuine and try to help him in finding some balance between common sense (philosophy) and (restricted) scientific views. They have a completely different value, however, if we deal with the basic philosophical problems initiated by Turing's paper.
9) Followers of Wittgenstein of course deny that a sensible use of the expression "mechanical rule" is possible.
Apart from Wittgenstein's seemingly cryptic remark that Turing's machines "are humans who calculate"\(^{10}\), however, I will only marginally concern myself with the problems of rule-following, a central concern of Wittgenstein's philosophy. An analysis of mechanical analogues (as a motif of a mechanistic philosophy) seems to be more important, and more promising, for the comparison between Turing's analysis of the concept of computation and its analogue within AI, that of rational argumentation. Furthermore, I shall try to discover how (epistemologically speaking) a search for such analogues came about. First of all, however, I shall concentrate on Turing's 1936 paper.

1 Technical Reconstruction of Turing's Paper of 1936

For a short reconstruction of the basic ideas of Turing's paper 'On Computable Numbers …', we shall concentrate on Martin Davis' 'What is Computation?'.\(^{11}\) One of the reasons for doing so is that this procedure allows us to exemplify the reception of Turing's language within the scientific community. After this reconstruction, there will be ample space to analyse the epistemological situation of Turing's approach, especially in § 9 of Turing's original paper.

Davis writes: "Turing based his precise definition of computation on an analysis of what a human being actually does when he computes." \([243, my emphasis]\). Of course, the term 'actually does' does not imply that a (human) calculator consciously knows what she/he does. Rather, it is the result of a theoretical analysis of the (human) calculation process,\(^{12}\) viewed through the eyes of a mathematician. I therefore suggest calling Davis' use of 'actually does' the\(\text{\textit{etico-explanatory}},\) and distinguish it from a more effective, literally descriptive and therefore more or less action guiding usage.

Davis goes on to interpret the H-computing behaviour in the following way: "Such a person is following a set of rules which must be carried out in a completely mechanical manner." \([243, my emphasis]\) For the mathematician, a 'set of rules' is an explanatory hypothesis for the identical

---


\(^{12}\) In the following, this will be called H-calculating process or H-operationalization of 'computing'.
reproduction of the behaviour (in our case the computing behaviour) of an agent in a controlled environment. It does by no means describe what a calculator actually thinks while calculating, how she/he/it applies these rules, or whether she/he/it is able to formulate these rules on request. The latter is a demand on the usage of 'following a rule' which would result from an application of Wittgenstein's philosophy (Philosophical Grammar). The fact that these rules are applied in a 'completely mechanical manner' implies that they are applied without reflection, but not necessarily without consciousness. "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".[243]¹³

However, Davis (following Turing) goes on to analyse the behaviour of a computer as if 'set of rules' and similar expressions were a literal description and not merely an analytical interpretation: "If we watch a human being calculating something (whether he is carrying out a long division, performing an algebraic manipulation, or doing a calculus problem), we observe symbols being written, say on a piece of paper." [243, my emphasis]

He goes on to argue that the behaviour of a person doing calculations changes once she/he perceives the various symbols which appear as a result of the calculation process. Having coined the reader to regard the H-calculating process from the mathematical point of view, Davis is now able to write: "The problem which Turing faced and solved was this: How can one extract from this process what is essential and eliminate what is irrelevant?" [243]

According to Davis, Turing's actual analysis starts at this point. Davis then specifies three essential abstractions as "series of [structural] restrictions on the calculator's [formal] behavior" [243].¹⁴ The result [of a mathematical calculation], then, are a few "very simple basic steps performed over and over again many times" [243].

¹³) Provided, of course, that one wants regularity within the results. This point aptly expresses our closeness as well as relative distance to Wittgenstein's approach. To my mind, explanatory 'rules' respectively structures are often, and wrongly, understood as rules in Wittgenstein's sense of the word. The difference between explanatory and descriptive languages and the confusion between the two seems to be the actual driving force behind the changing of rules for the use of words. Some of these rules are simply rules to be followed, others have only an explanatory status within a particular scientific discipline, considered as Wittgenstinian language game with real-world reference.

¹⁴) Words in brackets are inserted to clarify the philosophical context.
These restrictions are:

(i) **Independence of representation medium**, e.g. restriction to a one-dimensional (linear) tape.

(ii) **Binary encodability**, i.e. only zeros and ones are allowed to be written onto the tape (here Davis goes one step further than Turing).

(iii) **Restricted number of perceived symbols**, i.e. only one symbol can be scanned per time unit.

Yet the question arises what such a restricted "human" calculator can actually do? Davis' answer is that "he can replace a O by a 1 or a 1 by a O on the square he is scanning at any particular moment, or he can decide to shift his attention to another square. …" Also the calculator may observe the symbol in the square being scanned and make a decision accordingly. And presumably this decision should take the form: "Which instruction shall I carry out next?" Finally, the calculator may halt, signifying the end of the computation." [245-46]

Thus one can say: "Any computation can be thought of as being carried out by a human calculator, working with strings of zeros and ones written on a linear tape, who executes instructions of the form:

Write the symbol 1. -- Write the symbol 0. -- Move one square to the right. -- Move one square to the left. -- Observe the symbol currently scanned and choose the next step accordingly. -- STOP. [246]

---

15) "Turing assumed that this shifting of attention is restricted to a square which is the immediate neighbour, either on the left or on the right, of the square previously scanned. Again, this is obviously no essential restriction: if one wants to shift one's attention to a square three to the right, one simply shifts one to the right three successive times." [246]
A special sequence (list) of such instructions can be regarded as a (computer) program. In the Turing-Post "programming language"\(^{16}\), the instructions are expressed in the following way.

```
PRINT 1
PRINT 0
GO RIGHT
GO LEFT
GO TO STEP i IF 1 is SCANNED
GO TO STEP j IF 0 is SCANNED
STOP
```

A very simple example for a Turing-Post-program is Davis' 10-line doubling program which doubles the numbers of ones (input data).

```
1. PRINT 0
2. GO LEFT
3. GO TO STEP 2 IF 1 IS SCANNED
4. PRINT 1
5. GO RIGHT
6. GO TO STEP 5 IF 1 IS SCANNED
7. PRINT 1
8. GO RIGHT
9. GO TO STEP 1 IF 1 IS SCANNED
10. STOP
```

Now that we have alerted our sensitivity for the problem of computability and its solution, I would like to suggest, as a thesis, that for the "computability" of numbers, Turing has achieved what Descartes with his analytical geometry has achieved for the understanding and construction of numbers.

\(^{16}\) According to Davis, Emil L. Post's "work in computability theory included the independent discovery of turing's analysis of the computation process, …” [244].
geometrical figures. In proposing this thesis, however, I aim above all at an understanding of the content of the result of Turing's mathematical analysis of the process of computation.

In order to clarify what I intend with the above comparison between Turing and analytical geometry, let me dwell a little longer on one important aspect of the latter: Let us suppose that we concern ourselves with geometrical figures, e.g. a circle or a triangle, which can be drawn, i.e. generated with the help of a ruler and compasses. If we are doing analytical geometry, we replace the geometrical objects by sets of numbers which in general are determined by formulas or equations. Given a coordinate system and a circle drawn round its zero point at the distance r, we can say that the line of the circle consists of the set \( C(r) \) of pairs of points \((x,y)\) for which it holds that \( x^2 + y^2 = r^2 \).

I.e. \( C(r) := \{ (x,y) \in \mathbb{R} \times \mathbb{R} / x^2 + y^2 = r^2 \} \), with \( \mathbb{R} \) designating the set of real numbers, and \( \mathbb{R} \times \mathbb{R} \) the set of pairs of real numbers, i.e. the Cartesian product of real numbers.

The pre-requisite for all this, however, is that one has analysed, at least conceptually, the drawing of a circle in such a way that each point (in the Euclidean plane, given some system of coordinates) can be represented by a pair of numbers \((x,y)\), its coordinates. Within the "program" of analytical geometry, one can generate the line of a circle by applying the following steps: Choose an \( a \), compute \( a^2 \), replace, within the formula \( x^2 + y^2 = r^2 \), the variable \( x \) by \( a \), i.e. replace \( x^2 \) by \( a^2 \), calculate the value of \( y = b \) with the help of \( a \) and \( r \), so that \( b = (r^2 - a^2)^{1/2} \), identify the point \((a,b)\) within some given system of coordinates.

Of course, if someone argues that she/he does not draw a circle in that way, she/he is certainly right. Nevertheless, we can say that the formula \( x^2 + y^2 = r^2 \) can be understood as a correct (mathematical) analysis, or rather analytical encoding/representation, of a circle. It is important to note, however, that the set of points \( C(r) \) defined by this formula is an analytical replacement, and generates a circle in a completely different way. If we keep this in mind, the parallel to Turing's analysis of the process of computation should be clear: Turing-computability, i.e. the possibility to generate a number with the help of a Turing-Post-program, is analogous to the analytical generation of a geometrical figure, and also just an analysis via replacement. By no means does it give us a description of how we, as human beings, compute/calculate, yet it enables us to grasp an important (mathematical) property of the (human) way of computing/calculating, just as analytical geometry does.

17) Unless a modern computer drawing-program is used.
for drawing a geometrical figure. I propose to call this "grasping" theoretico-explanatory, in order to stress the fact that no literal description is at stake.

**Two Level Model**

< simple mechanical operations >

<table>
<thead>
<tr>
<th>T-computable</th>
<th>algorithm</th>
<th>generated by analytical means</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \delta a = \alpha \rightarrow \delta b = \beta ]</td>
<td>encoding</td>
<td>[ \delta F \rightarrow \delta G ]</td>
</tr>
</tbody>
</table>

\[ a \rightarrow b = m \]

< simple rules >

generated with ruler and compasses

H-computable

realm of numbers

A number \( \delta a \) from \( L \) is Turing-computable, if there exists a Turing-Post-program/algorithm for a Turing-machine such that \( \delta a \) can be generated.

realm of geometrical figures

A set \( \delta F \) of pairs of points from \( L \) is called analytically reproduceable, if there is a formula/equation such that \( \delta F \) is the set of all pairs of points which hold for the formula in question.

\[ L \] (which will later stand for a formal language) designates those sequences of zeros and ones which can be written upon the tape of a Turing-machine. \( R \) designates the set of numbers used by a human calculator.\(^{18}\) If \( a \) is an element of \( R \), let \( \delta a \) designate the associated binary expression of \( a \).

For reasons of analogy, let \( R \) also designate the set of geometrical figures from the Euclidian plane which can be drawn effectively, i.e. by means of compasses and ruler. In this case, \( L \) should consist of the associated analytical sets of points - pairs of numbers \((x,y)\) - for which it is true that the point \( P \) form the euclidian plane which is depicted by \((x,y)\) is element of some given figure \( F \) from \( R \). For

\[^{18}\text{I have consciously chosen not to employ any of the common denominations such as } \mathbb{R} \text{ for real number or } \mathbb{Z} \text{ for integers.}\]
reasons of analogy, and in order to press the point, we write $\delta F$. Thus, in both cases, we assume the existence of a mapping $f : \mathbb{R} \to \mathbb{L}$; so that $E \mapsto f(E) = \delta E$; $a \mapsto f(a) = \delta a$.

All arrows in the diagram are mathematical mappings. Some of them, $a \mapsto b$, $\alpha \mapsto \beta$ can be understood as operations, e.g. if $b$ is computed from $a$, in which case we can consider " $\mapsto$ " as actualized by an algorithm. " $\mapsto$ " represents the rules used by humans to generate some $b$ out of a given $a$. Thus " $\mapsto$ " designates T-computability, and " $\mapsto$ " computability by humans.\(^{19}\)

After introducing the two-level model, let us now consider what exactly it means if we say that Descartes and, by analogy, Turing, have grasped a figure in the Euclidian plane respectively the process of H-computability completely. Such a statement refers to the mapping $f : \mathbb{R} \to \mathbb{L}$; $a \mapsto f(a) = \delta a$ and is concerned with the (questionable) assumption that $<\mathbb{L}, f>$ has grasped an essential aspect of $\mathbb{R}$.

As far as analytical geometry is concerned, we could argue that the analysis (of geometrical figures $F$) is complete iff for each figure $F$ belonging to $\mathbb{R}$ there exists at least one formula (or equation) which can reproduce the set of pairs of points $f(F) = \delta F$ in a controlled manner. Our intuition tempts us to say: there is no figure in $\mathbb{R}$ which could not be grasped by some formula, or: for each set of points (in $\mathbb{L}$) which represents a geometrical figure there exists a formula which is able to reproduce the set in question.\(^{20}\)

Thus, one could say cum grano salis -- and this amounts to some kind of definition -- :

A figure $F$ (from $\mathbb{R}$) is a geometrical figure if the set of points $P$ from $\mathbb{R}$ is consists of can be represented by some (analytical) formula.\(^{21}\)

At this point, the parallel to Church's definition, which equates effectively calculable ($\lambda$-definable) functions with the recursive functions of Kleene, should become clear. Since these effective functions coincide with Turing-computable functions, however, we are back at functions which can be generated by Turing-machines. By extending our analogy, we could say (or even argue): Turing's

\(^{19}\) In an equivalent way, there may be thought up means of transforming figures into others.

\(^{20}\) Of course, modern (mathematical) theory of measures considers figures (in general to be constructed by means of the axiom of choice) which are not Lebesque-measureable. This contradicts our intuition of the concept of measures.

\(^{21}\) Of course, the two levels can deliberately be mixed up.
analysis of the H-calculation process is complete iff for each H-calculable element \( a \) from \( \mathbb{R} \) there exists a TP-program (i.e. an algorithm) such that \( f(a) = \delta a \) can be generated via TP, i.e. \( \delta a \) is T-computable.

The basic intuition for this argument is that Turing's analysis has grasped the H-process of calculating completely, if there does not exist an H-computation (HC in short) of some number \( b = HC(a) \) from \( \mathbb{R} \) for which there does not exist a TP-program, so that the application of TP to \( \alpha = \delta a \) would yield the value \( \beta = \delta b \). My assumption is that in the case of T-computability, just as in the case of analytical geometry, one will get used to working with definitions by replacement or explications by replacement, especially so since this is already happening in the technical realizations of the computer industry.

Having thus deviated a little to clarify and explicate the terms we are using, let us now consider a further, central discovery by Turing, namely that it is possible to encode the instructions of a TP-program as a sequence of zeros and ones (i.e. in the binary code) on the tape of a Turing-machine. This means that a further TP-program (TP*) can operate, which is able to provide Turing-computations on the given (encoded) TP program. In terms of the two-level model (and our analogy to analytical geometry), this means that for some given \( \delta F \), we can specify a new formula in \( L \), which can generate "\( \delta F \)" (now understood as a figure of \( \mathbb{R} \)). Let TP be a program/algorithim (in \( L \)) containing a step S which transforms \( \alpha = \delta a \) into \( \beta = \delta b \), that is, it produces \( \beta \) out of \( \alpha \) (i.e. \( S(\alpha) = \beta \)). Let \( \delta \) applied to \( S(\alpha) = \beta \) (i.e. \( \delta (S(\alpha) = \beta) \)) be the binary code A as element of \( L \). We then can apply some program TP* to A, such that the result is some B from \( L \), i.e. \( TP^*(A) = B \).

---

22) I have chosen the designation S deliberately. Actually, we consider \( S(a) = b \) as an element of \( \mathbb{R} \) because it can be regarded as a jumingly possible action to generate b from a with the help of S. To a certain extent, we take the role of S ourselves, an action which can be traced back to calculating b from a in a straightforward way.
There are of course many ways of encoding a TP-program. Davis gives the following example:

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruktion</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>PRINT 0</td>
</tr>
<tr>
<td>001</td>
<td>PRINT 1</td>
</tr>
<tr>
<td>010</td>
<td>GO LEFT</td>
</tr>
<tr>
<td>011</td>
<td>GO RIGHT</td>
</tr>
<tr>
<td>101...01</td>
<td>GO TO STEP i IF 0 IS SCANNED</td>
</tr>
<tr>
<td>110...10</td>
<td>GO TO STEP j IF 1 IS SCANNED</td>
</tr>
<tr>
<td>100</td>
<td>STOP</td>
</tr>
</tbody>
</table>

For example: The code for a programming step of type 5, i.e. "GO TO STEP 3 IF 0 IS SCANNED" is "1010001". Davis' complete doubling program can be coded in the following way.

```
Begin       Step 1   Step 2   Step 3   Step 4   Step 5
1           000   010    110110  001   011
Step 6      Step 7   Step 8   Step 9   Step 10   End
110111110   001   011    11010  100   111
```

For the following considerations, it is essential that "the code of a Turing-Post program can be deciphered in a unique, direct and straightforward way, yielding the program of which it is the code". [252, my emphasis]

The crucial point, however, is the possibility to construct a universal program U, so that "there exists a single (appropriately constructed) Turing-Post program which can compute whatever that is computable." [252, my emphasis] ²³

²³) "Compute", of course, refers to Turing-computability and not to computability in a human sense. The latter would only hold if Turing-computability completely grasps H-computability, a fact which to my understanding is only an empirical hypothesis!
Such a universal Program U can imitate the behaviour of some give Turing-Post-program TP simply by writing the code of TP (in our notation δTP), i.e. the sequence of zeros and ones (which represents TP in \[
\]) onto the tape and permit U to operate on it. The non-blank part of the tape has to consist of δTP, followed by an input string v on which U can operate.

\[
\begin{array}{cccccc}
1 & 0 & \underbrace{1} & 111 & 11 \\
\text{begin} & \text{coded instructions of} & \text{end} & \text{input}
\end{array}
\]

doubling program

[253]

This means that U should imitate the behaviour of the doubling program if the input is 11. At the end of the application of U [Turing computation] the tape should look just as after the operation of the original program TP, i.e. it should read \(\ldots 11 \ 11 \ 00 \ \ldots\) . A universal TP-program U should run in this way for every TP-program.

To sum up, we can again quote Davis: "U is to begin its computation presented with a tape whose nonblank portion consists of code (TP) \([\delta TP \text{ in unserer Notation}]\) some Turing-Post programm TP (initially scanning the first symbol, necessarily 1, of this code) followed by a string v. U is then supposed to compute exactly the same result as the programm TP would get when starting with the string v as the nonblank part of the tape (scanning the initial symbol of v). Such a programm U can then be used to simulate any desired Turing-Post programm TP by simply placing the string code (TP) \([= \delta TP]\) on the tape." [253]

Davis now asks which reason we could possibly have to believe in the existence of such a program. For reasons of illustration, one should imagine "how a human calculator could do what U is supposed to do."[253]

"Faced with the tape contents on which U is supposed to work, such a person could begin by scanning this string of zeros and ones, from left to right, searching for the first place that 3 consecutive ones appear. This triple 111 marks the end of code (TP) \([= \delta TP]\) on one sheet of paper and the input string on another. As already explained, he can decode the string code (TP)\(=\) δTP and obtain the actual Turing-Post program TP. Finally, he can "play machine", carrying out the

---

24) I have adapted the notation, the original has P instead of TP.
instruction of TP, applied to the given input string in a robotlike fashion. If and when the computation comes to a halt, our calculator can report the final tape contents as output." [253]

Davis' description shows how at least a human calculator/computer could achieve what U should accomplish. According to Turing's analysis of the process of computation, however, we are lead to the conviction that there must be a Turing-computation (a TP-program) which can carry out the process described above. The evidence for the existence of such a (universal) program, however, is not a mathematical proof, but the validity of Turing's analysis of the H-process of computation.

In the same way, we could develop a TP-program corresponding to U, just as Turing did in his 1936 paper in a different but entirely equivalent context.

Finally, let us stress once more that the code for a given TP-program can be regarded as a 'program' which performs the computations for which TP was intended.

25) In addition to this, we have also to think of the two-level model and the formulation of the problem of completeness.