Carnegie Mellon University
Research Showcase

Department of Philosophy | Dietrich College of Humanities and Social Sciences

1-1-1996

Godel, Turing, and K-graph machines

Sieg
Carnegie Mellon University

John Byrnes

Follow this and additional works at: http://repository.cmu.edu/philosophy

Recommended Citation
Sieg and Byrnes, John, "Godel, Turing, and K-graph machines" (1996). *Department of Philosophy.* Paper 538.
http://repository.cmu.edu/philosophy/538

This Technical Report is brought to you for free and open access by the Dietrich College of Humanities and Social Sciences at Research Showcase. It has been accepted for inclusion in Department of Philosophy by an authorized administrator of Research Showcase. For more information, please contact research-showcase@andrew.cmu.edu.

NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

Gödel, Turing, and
K-GRAPH MACHINES

by

Wilfried Sieg and John Byrnes

December 1996

Report CMU-PHIL-76

Carnegie Mellon

Philosophy
Methodology
Logic

Pittsburgh, Pennsylvania 15213-3890

Gödel, Turing, and K-Graph Machines[1]

Wilfried Sieg and John Byrnes
Carnegie Mellon University
Pittsburgh

[1]This paper contains a synopsis, we hope an informative and accessible one, of [Sieg 1994] and our [1996]; however, the synopsis is put to work for a particular interpretation of a brief remark of Gödel on Itoing's work. For historical and mathematical details we refer the reader to these papers, but also to [Mundici and Sieg].

Introduction

The notion of effective calculability is central for logic and the philosophy of mathematics, not to speak of computer science, artificial intelligence, and cognitive science. Turing gave in 1936 what is viewed as the most convincing analysis of this informal notion[2]: he was led to the *mathematical concept* of computability by idealized machines and to the *thesis* that every effectively calculable number-theoretic function is computable by such a (Turing-) machine. In support of the thesis, Turing argued that all processes carried out by human computers—when effectively calculating the values of a number-theoretic function—can be performed by his machines. In 1964, Gödel stated briefly and enigmatically:

> *Turing's work gives an analysis of the concept of "mechanical procedure"_____This concept is shown to be equivalent with that of a "Turing machine".*[3]

(We will refer to this statement as (G).) Nowhere in Gödel's writings is there an indication of the nature of Turing's analysis (as leading to a mathematical concept) or of a proof for the claim that the analyzed concept is equivalent to that of a Hiring machine. We explore Turing's informal considerations in section 9 of his [1936] and argue that Gödel's schematic description is correct, once it is properly "triangulated" and explicated. Then we generalize Turing's conceptual analysis and describe K-graph machines introduced in our [1996]. Our goal, very briefly put, is to deepen the understanding of the broad conceptual issues, which are still quite controversial, and to provide an illuminating historical and philosophical perspective; cf. also [Soare].

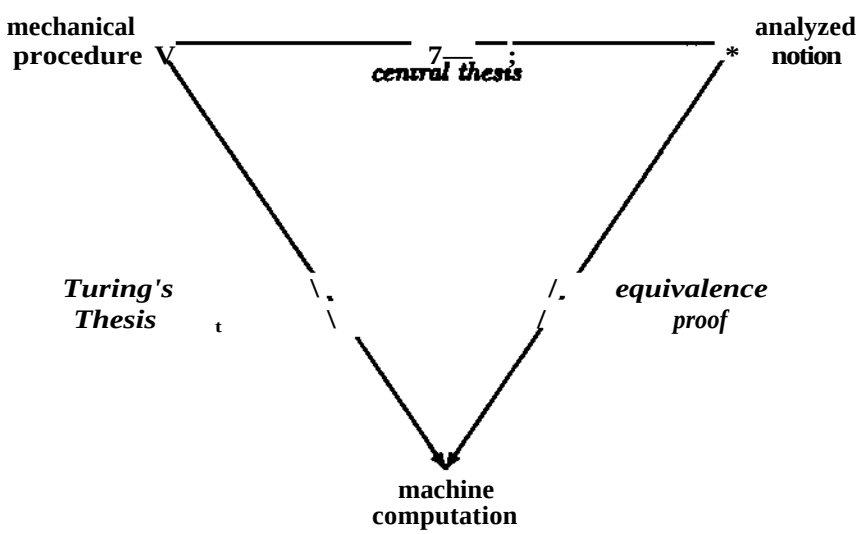
1 Mechanical procedures

In order to give the most general formulation of the incompleteness theorems and thus to allow for their broadest philosophical interpretation, a general concept of "formality" is required. For Gödel that meant to provide a precise characterization of effective calculability for number theoretic functions. That was a primary, motivating concern for Gödel; cf. [Gödel 1951] and [Davis 1982]. To approach such a characterization, Gödel introduced in his Princeton Lectures of 1934 the class of *general recursive functions*, i.e., those functions whose values can be calculated in an equational calculus from basic equations by means of very elementary rules. Clearly, any system that is adequate for the formulation of (weak parts of elementary) number theory will allow such calculations. Gödel noticed in 1936, to his own surprise, that no extension by higher types (even transfinite ones) would allow the calculation of more functions; this is the basis for his assertion that the concept of computability has a certain *absoluteness:*

[2]Indeed, it can be argued that he gave the *only* convincing analysis.
[3]The underlining is ours.

> Thus the notion 'computable' is in a certain sense 'absolute[1], while almost all metamathematical notions otherwise known (for example, provable, definable, and so on) quite essentially depend upon the system adopted, (p. 399)

In his lecture for the Princeton bicentennial conference, Gödel re-emphasized the significance of absoluteness. And yet, Gödel found only Turing's analysis convincing and claimed that the latter's work provides "a precise and unquestionably adequate definition of the general concept of formal system" [1964, p. 369]. As a formal system is simply defined to be a mechanical procedure for producing theorems, the adequacy of the definition rests clearly on Turing's analysis of the concept "mechanical procedure". The analyzed concept must obviously be a sharp, mathematical notion to enter a rigorous proof showing its equivalence to the concept of a Turing machine. Thus, there is still a need for a thesis to connect it to the informal concept; we call that thesis the *central thesis.* The methodological point is, for sure, that the central thesis should be less problematic than the full thesis. Gödel's schematic description, quoted as (G) in our *Introduction,* can now be represented diagramatically as in the figure below; we also say that it can be "triangulated".



To facilitate the step from the intended informal to the analyzed mathematical concept, we should first ask: What is the historical and systematic context in which "mechanical procedures" have been used informally? In a broad sense the underlying, largely epistemological issues go back to at least Leibniz. In a narrower sense they were emphasized in the logical developments at the turn of the

19th century. They were developed most sharply by Hilbert in the twenties for metamathematical investigations including the *Entscheidungsproblem*. Gödel described matters lucidly with respect to the "formalization" of mathematics in his [1933o]:

> The first part of the problem [i.e., the reduction of mathematical proofs to a minimum number of axioms and primitive rules of inference] has been solved in a perfectly satisfactory way, the solution consisting in the so-called formalization of mathematics, which means that a perfectly precise language has been invented, by which it is possible to express any mathematical proposition by a formula. Some of these formulas are taken as axioms, and then certain rules of inference are laid down which allow one to pass from the axioms to new formulas and thus to deduce more and more propositions, the outstanding feature of the rules of inference being that they are purely formal, i.e., refer only to the outward structure of the formulas, not to their meaning, so that they could be applied by someone who knew nothing about mathematics, or by a machine. (p. 45)

The implicitly formal character of his theories allowed Gödel later to assert the absoluteness of the concept of computability. This implicit feature was made explicit in the marvelous analysis of computations in "deductive formalisms" presented in Supplement II of Hilbert and Bernays's *Grundlagen der Mathematik II*: formality restrictions are imposed by "recursiveness conditions", the crucial one requiring the proof predicate of deductive formalisms to be primitive recursive. This analysis can be viewed as providing the proper underpinning for Gödel's absoluteness, but for sure only relative to the recursiveness conditions. At issue is then the question, why such conditions should be accepted. The very same question can be raised with respect to Church's (semi-circular) argument for the identification of effective calculability with general recursiveness in [1936], when requiring the steps in logical calculi to be general recursive.[4]

We want to emphasize that this work is of real interest, focusing as it does on one core notion, namely that of a *reckonable function*. (Hilbert and Bernays talk of a *regelrecht auswertbare Funktion*, i.e., a function whose values can be determined by following rules.) Each of these analyses explicates effective calculability by computability in a "formal calculus"; however, neither argues in a convincing way for the particular restrictions on the steps that are allowed, that is, for the central thesis involved. That is particularly evident in those cases where the restrictions are made only implicitly: in Gödel's proposal only simple substitution rules are used; in Church's argument, an unsupported claim is made that the steps have to be general recursive. Church recognized the special character of Turing's proposal immediately in his 1937 review of Turing's paper. Church contrasted in it Turing's mathematical notion for effective

[4]Church used in his [1936] also a notion of "calculable in a logical calculus" to give an explication of the informal notion "effective calculability"; to obtain the desired result, he had to assume, however, that the proof steps in the calculus are recursive; cf. section 7 of [Church 1936] and [Sieg 1994], pp. 85–7.

calculability (via idealized machines) with his own (via A-definability) and with Gödel's (via the equational calculus); he asserted: "Of these [notions], the first has the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately —" But neither here nor anywhere else does Church argue for this immediate evidence.

2 Computors & machines

What is the crucial and novel aspect of Turing's analysis? Let us emphasize outright that Turing's analysis is neither concerned with *machine computations* nor with general *human mental processes.*[5] Rather, it is *human mechanical computability* that is being analyzed, and the special character of this intended notion motivates the restrictive conditions that are brought to bear by Turing! To point out vividly the distinguishing feature of Turing's analysis, we mention briefly Post's simultaneous definition of "finite combinatory processes". In their description, Post makes use of human workers operating in a "symbol space" and carrying out, over a two letter alphabet, exactly the kind of operations Turing machines can. However, the fact that a human being is carrying out the computation is not used at all in Post's arguments for his model.[6] By contrast, Turing exploits in a radical way that a *human computor is* performing *mechanical procedures* on *symbolic configurations:* the immediate recognizability of symbolic configurations is demanded so that basic (computation) steps cannot be further subdivided. This demand and the evident limitation of the computer's sensory apparatus lead to the formulation of boundedness and locality conditions. Less importantly, Turing requires also a determinacy **condition (D),** i.e., the computor carries out deterministic computations, as his internal state together with the observed configuration fixes uniquely the next computation step. The **boundedness conditions** can be formulated as follows:

(B.I) *there is a fixed bound for the number of symbolic configurations a computor can immediately recognize;*

(B.2) *there is a fixed bound for the number of a computer's internal states that need to be taken into account.*[7]

For a given computor there are consequently only boundedly many different combinations of symbolic configurations and internal states. Since his behavior is, according to (D), uniquely determined by such combinations and associated

[5]The former mistaken view is taken, for example, in [Mendelson] and also in [Church 1937]; the latter equally mistaken view is propounded, for example, in [Webb] and also in [Gödel 1972].

[c]The support for his model Post sees as being provided inductively by considering wider and wider formulations and reducing them to the restrictive formulation given in his [1936]; cf. [Sieg 1994], pp. 91-2.

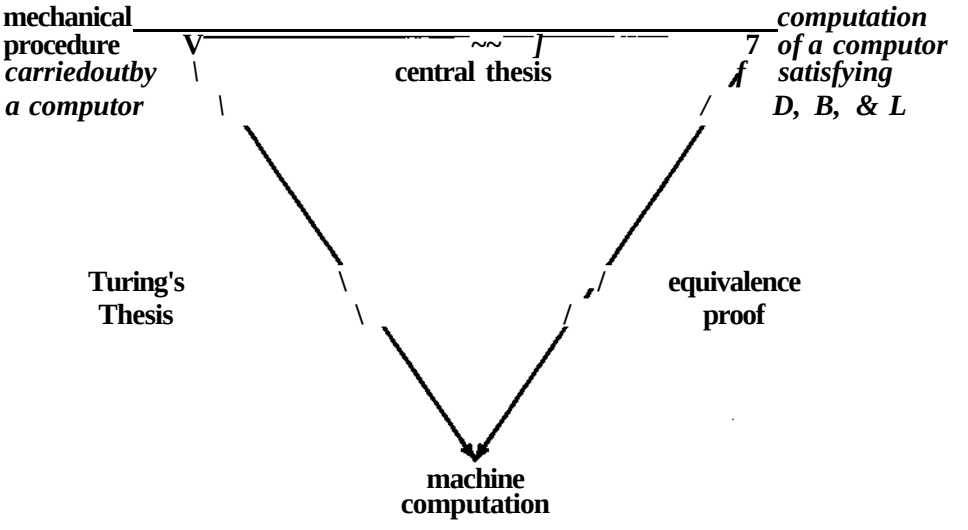
***This condition (and the reference to internal states) can actually be removed and was removed by Turing; nevertheless, it has been a focus of critical attention; cf. the Concluding Remarks.**

operations, the computor can carry out at most finitely many different operations. These operations are restricted by the following locality conditions:

(L.I) *only elements of observed configurations can be changed;*

(L.2) *the distribution of observed squares can be changed, but each of the new observed squares must be within a bounded distance of an immediately previously observed square.*[8]

Turing's computor proceeds deterministically, must satisfy the boundedness conditions, and the elementary operations he can carry out must be restricted as the locality conditions require. Every number-theoretic function such a computor can calculate, Turing argues, is actually computable by a Turing machine over a two-letter alphabet. This explication of Turing's analysis allows us to annotate Gödel's triangle in the following way:



Thus, on closer inspection, Turing's Thesis that the concept "mechanical procedure" can be identified with madiine computability is seen as the result of a two part analysis. The first part yields axioms expressing boundedness conditions for symbolic configurations and locality conditions for mechanical operations on them, together with the centm/ *thesis* that any mechanical procedure can be carried out by a computor satisfying the axioms. The second part argues for the *claim* that every number-theoretic function calculable by such a computor is

[8]This is almost literally Itaing's formulation; obviously, it takes for granted a particular structure underlying the symbolic configurations.

computable by a Turing machine. In lining's presentation these quite distinct aspects are intertwined and important steps in arguments are only hinted at. Notice also that in the very formulation of the second locality condition, features of the precise model of computation are appealed to, namely, to express that the computor's attention can be shifted only to symbolic configurations that are not "too far away" from the currently observed configuration.

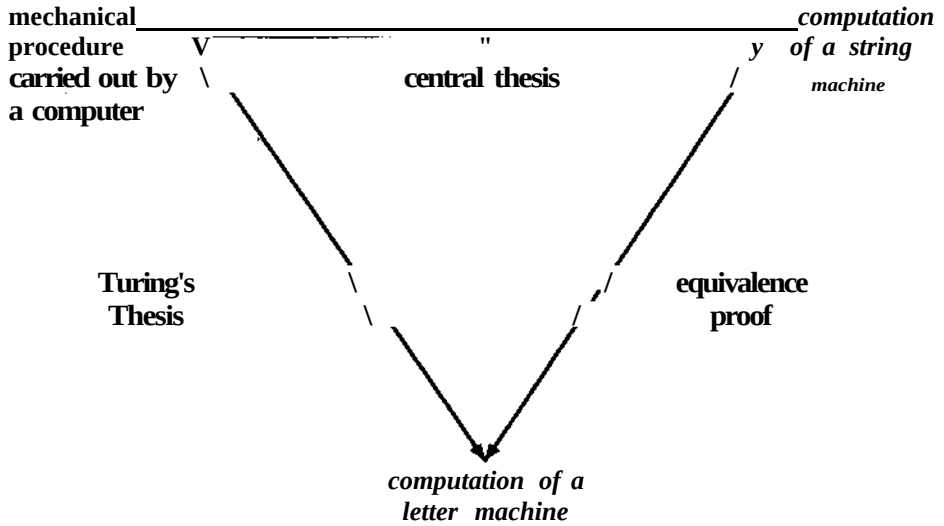
3 Turing's analysis sharpened

The central thesis makes use of restrictive boundedness and locality conditions on computors. However, to formulate these conditions precisely and to make the further mathematical considerations possible, the symbolic configurations are taken by Hiring to be linear; here are the starting-point and the dimension-lowering step:

> Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, t.e. on a tape divided into squares. [1936; p. 135]

In his further reductive argument Hiring constructs machines that mimic the work of computors on linear configurations directly, and then he observes: "The machines just described do not differ very essentially from computing machines as described in §2, and corresponding to any machine of this type a computing machine can be constructed to compute the same sequence, that is to say the sequence computed by the computer [in our terminology: computer]." The more general machines are not described in mathematical detail; if one wants to do so, it is best to use the presentation of Hiring machine programs by Post production rules. That was done first by Post in his superb 1947 paper, was adopted by Turing in his 1953 paper "Solvable and unsolvable problems", and is presented in Davis' classical textbook *ComputabUity and Undecidability.*

As the basic operations that modify the tape content lead from letters to letters, we call the basic Hiring machines also *letter machines.* To formulate and prove rigorously the more general form of Turing's Theorem we have to specify the intended finite symbolic configurations and the mechanical operations that can be carried out on them. The generalized machines (mimicking the computor's operation on linear configurations) can simply be given by production rules that allow the replacement of finite strings by finite strings; such machines are then called *string machines.* Thus, the claim really established by Hiring amounts to showing that computations by string machines can be carried out by letter machines. (For details, we refer the reader to our [1996], pp. 103-4.)



In the historical context in which Hiring found himself, he asked exactly the right question: What are the elementary processes a computor carries out (when calculating a number)? Note that Turing was concerned with *symbolic processes,* not—as the other proposed explications—with processes directly related to the evaluation of (number theoretic) functions. Indeed, the general problematic required an analysis of the idealized capabilities of a computor, and it is precisely this feature that makes the analysis epistemologically significant. The separation of conceptual analysis and rigprous proof is essential for clarifying on what the correctness of 'Airing's central thesis rests, namely, on recognizing that the boundedness and locality conditions are true for a computor and, also, for the particular precise, analyzed notion.

4 Turing's analysis extended

Turing intended in 1936 to give an analysis of mechanical procedures on two-dimensional configurations; but such processes are *not* described in detail, let alone proved to be reducible to computations on strings or letters. In 1953, Turing considers even three-dimensional configurations and mechanical operations on them. He starts out with a description of puzzles: square piece puzzles, puzzles involving the separation of rigid bodies or the transformation of knots; i.e., puzzles in two and three dimensions. "Linear" puzzles are described as Post production systems and are called *substitution puzzles.* They are viewed by Taring as a "normal" or "standard" form of describing puzzles; indeed, a form of Turing's Thesis is formulated as follows:

> Given any puzzle we can find a corresponding substitution puzzle which is equivalent to it in the sense that given a solution of the one we can easily find a solution of the other. If the original puzzle is concerned with rows of pieces of a finite number of different kinds, then the substitutions may be applied as an alternative set of rules to the pieces of the original puzzle. A transformation can be carried out by the rules of the original puzzle if and only if it can can be carried out by substitutions——(p. 15)

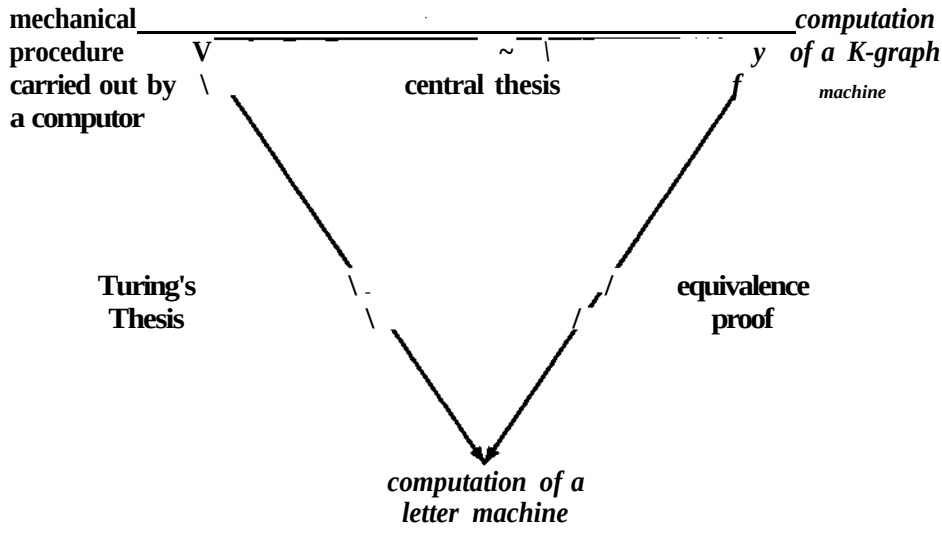
Turing admits that this formulation is "somewhat lacking in definiteness" and claims that it will remain so; he characterizes its status as lying between a theorem and a definition: "In so far as we know a *priori* what is a puzzle and what is not, the statement is a theorem. In so far as we do not know what puzzles are, the statement is a definition which tells us something about what they are." Of course, Hiring continues, one could define puzzle by a phrase beginning with "a set of definite rules", or one could reduce its definition to that of "computable function" or "systematic procedure". A definition of any of these notions would provide one for puzzles.

In neither place does Turing even try to characterize more general configurations. This gap in Turing's work was the starting-point of our [1996], which is sketched in the remainder of this section. The informal idea behind our way of describing mechanical procedures on general symbolic configurations has three distinct components: the computor operates on finite *connected* configurations; such configurations contain a unique *distinguished element* (corresponding to the scanned square of a Turing machine); the operations *substitute neighborhoods* of the distinguished element by appropriate other neighborhoods, and such substitutions are given by a finite list of generalized Post production rules. In this way, we generalize the notions and yet remain within Turing's general analytic framework.

In order to give precise mathematical shape to these informal ideas, we introduce K-graph machines, i.e., machines that operate on K-graphs. (K stands for Kolmogorov, as the essential idea for these machines was extracted from Kolmogorov and Uspensky's analysis of algorithms.) K-graph machines do not just operate on strings of symbols with one scanned element, but on finite connected and labeled graphs with a single central vertex. The graphs satisfy the *principle of unique location,* i.e., distinct paths containing the central vertex have distinct associated sequences of labels. Operations of K-graph machines obey the *principle of local action,* i.e., replace distinguished subgraphs with the central vertex by finite graphs (also containing the central vertex). A program is a finite list of generalized production rules specifying such replacements.

The starting-point of Turing's analysis was (the representation of) algorithms carried out in a "child's arithmetic book". K-graph machines capture it in a most natural way: in [1996] we indicate how the elementary school algorithm for column addition can be *directly* presented as a K-graph machine computation. Letter machines are easily seen to be K-graph machines and, that is the main technical result in our [1996], K-graph machines can be simulated

by letter machines. In triangular terms, K-graph machines take the place of string machines.



Given this mathematical analysis, Turing's central thesis is turned into the thesis that K-graph machines, clearly satisfying appropriately formulated boundedness and locality conditions, do the work of human computors directly. The theorem thus reduces mechanical processes to Hiring machine computations. The plausibility of Turing's Thesis rests exclusively on that of the central thesis involved in its triangulation. The claim, "Mechanical processes are (very easily seen to be) computations of K-graph machines", is most plausible.

Concluding Remarks

Gödel, in a Note from 1972, spotted a "philosophical error" in Turing's work and claimed that Turing's argument in the 1936 paper was intended to show that "mental procedures cannot go beyond mechanical procedures"; he considered the argument as "inconclusive". The reference to Turing's paper is to page 250 and obviously, from Gödel's subsequent argument, to the passage:

> We will also suppose that the number of states of mind which need be taken into account is finite. The reasons for this are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be "arbitrarily close"

> and will be confused. Again, the restriction is not one which seriously affects computation, since the use of more complicated states of mind can be avoided by writing more symbols on the tape.

Crucial is the remark, "the restriction is not one which seriously affects computation"; the notion to be explicated is for Turing *mechanical* computation, not mental procedure. The charge of a "philosophical error" is particularly surprising as Turing discusses (on pp. 253-4) a modification of the earlier argument and avoids altogether the introduction of 'the 'state of mind' by considering a more physical and definite counterpart to it". The further discussion in Turing's paper is both amusing and informative:

> It is always possible for the computer [computer, in our terminology] to break off from work, to go away and forget all about it, and later to come back and go on with it. If he does this he must leave a note of instructions (written in some standard form) explaining how the work is to be continued. This note is the counterpart of the "state of mind". We will suppose that the computer works in such a desultory manner that he does never more than one step at a sitting. The note of instructions must enable him to carry out one step and write the next note. Thus the state of progress of the computation is completely determined by the note of instructions and the symbols on the tape.

In sum, it is right that Turing does not give a conclusive argument for Gödel's claim, but it has to be added in all fairness that Turing did not (intend to) argue for it. Even in his later work, dealing explicitly with mental processes, Turing does not argue that "mental procedures cannot go beyond mechanical procedures", when the latter are made precise by (Turing-) machine computations. But that is a different, exciting topic.

References

[1] Church, A. An unsolvable problem of elementary number theory; American Journal of Mathematics 58, 345-363. (1936)

[2] Church, A. Review of [Turing 1936]; Journal of Symbolic Logic 2(1), 4(M1. (1937)

[3] Davis, M. Why Gödel didn't have Church's Thesis; Information and Control 54, 3-24. (1982)

[4] Gödel, K. The present situation in the foundations of mathematics; in: *Collected Works III*, 45-53. (1933)

[5] Gödel, K. On undecidable propositions of formal mathematical systems; in: *Collected Works* /, 346-371. (1934)

[6] Gödel, K. Über die Länge von Beweisen; in: *Collected Works I,* 396-399. (1936)

[7] Gödel, K. Remarks before the Princeton bicentennial conference on problems in mathematics; in: *Collected Works II,* 150-153. (1946)

[8] Gödel, K. Some basic theorems on the foundations of mathematics and their implications; in: *Collected Works III,* 304-323. (1951)

[9] Gödel, K. Some remarks on the undecidability results; in: *Collected Works II,* 305-306. (1972)

[10] Hilbert, D. and Bernays, P. *Grundlagen der Mathematik II;* Springer Verlag, Berlin. (1939)

[11] Kolmogorov, A. and Uspensky V. On the definition of an algorithm; AMS Translations 21, 2, 217-245. (1963)

[12] Mendelson, E. Second thoughts about Church's Thesis and mathematical proofe; The Journal of Philosophy 87 (5), 225-233. (1990)

[13] Mundici, D. and Sieg, W. Paper machines; Philosophia Mathematica 3, 5-30. (1995)

[14] Post, E. Recursive unsolvability of a problem of Thue; Journal of Symbolic Logic 12, 1-11. (1947)

[15] Sieg, W. Mechanical procedures and mathematical experience; in: *Mathematics and Mind* (edited by A. George); Oxford University Press, Oxford, 71-117. (1994)

[16] Sieg, W. and Byrnes, J. K-graph machines: generalizing Turing's machines and arguments; in: *Gödel '96* (edited by P. Hajek); Lecture Notes in Logic 6, Springer Verlag, Berlin, 98-119. (1996)

[17] Soare, R. Computability and recursion; Bulletin of Symbolic Logic 2 (3), 284-321.(1996)

[18] Hiring, A. On computable numbers, with an application to the Entscheidungsproblem; Proc. London Math. Soc., series 2, 42, 230-265. (1936)

[19] Truing, A. Solvable and unsolvable problems; Science News 31, 7-23. (1953)

[20] Wang, H. *From Mathematics to Philosophy;* Routledge & Kegan Paul, London. (1974)

[21] Webb, J. C. *Mechanism, Mentalism, and Metamathematics;* Reidel Publishing Company, Dordrecht. (1980)

[22] Webb, J. C. Introductory Note to [Gödel 1972], in: *Collected Works II,* pp. 292-304. (1990)