RECURSIVE UNSOLVABILITY OF A PROBLEM OF THUE

This paper contains the first unsolvability proof for a problem from classical mathematics - in this case the word problem for semigroups. The algebraically minded reader will readily note that what Post calls a Thue system determines a homomorphism on the free semigroup with the appropriate generators. Post thus constructs a particular semigroup for which it is recursively unsolvable to determine whether a given pair of elements of the free semigroup are mapped onto the same element by the homomorphism.

An independent proof of this result was given by A. A. Markov (C. R. (Doklady) Acad. Sci. U. S. S. R. (n. s.) 55(1947), pp. 583-586.)

The appendix to this paper has already been mentioned in the editorial remarks on p. 115 preceding Turing's paper, "On Computable Numbers, with an Application to the Entscheidungsproblem".

Emil Post

RECURSIVE UNSOLVABILITY OF A PROBLEM OF THUE

Alonzo Church suggested to the writer that a certain problem of Thue [6][1] might be proved unsolvable by the methods of [5]. We proceed to prove the problem recursively unsolvable, that is, unsolvable in the sense of Church [1], but by a method meeting the special needs of the problem.

Thue's (general) problem is the following. Given a finite set of symbols a_1, a_2, $\cdots$, a_μ, we consider arbitrary *strings* (Zeichenreihen) on those symbols, that is, rows of symbols each of which is in the given set. Null strings are included. We further have given a finite set of pairs of corresponding strings on the a_i's, (A_1, B_1), (A_2, B_2), $\cdots$, (A_n, B_n). A string R is said to be a *substring* of a string S if S can be written in the form URV, that is, S consists of the letters, in order of occurrence, of some string U, followed by the letters of R, followed by the letters of some string V. Strings P and Q are then said to be *similar* if Q can be obtained from P by replacing a substring A_i or B_i of P by its correspondent B_i, A_i. Clearly, if P and Q are similar, Q and P are similar. Finally, P and Q are said to be *equivalent* if there is a finite set $R_1, R_2, \cdots, R_r$ of strings on $a_1, \cdots, a_\mu$ such that in the sequence of strings $P, R_1, R_2, \cdots, R_r, Q$ each string except the last is similar to the following string. It is readily seen that this relation between strings on $a_1, \cdots, a_\mu$, is indeed an equivalence relation. Thue's problem is then the problem of determining for arbitrarily given strings A, B on $a_1, \cdots, a_\mu$ whether, or no, A and B are equivalent.

This problem, at least for the writer, is more readily placed if it is restated in terms of a special form of the canonical systems of [3]. In that notation, strings C and D are similar if D can be obtained from C by applying to C one of the following operations:

$$PA_iQ \text{ produces } PB_iQ, \quad PB_iQ \text{ produces } PA_iQ, \quad i = 1, 2, \cdots, n. \tag{1}$$

In these operations the operational variables P, Q represent arbitrary strings. Strings A and B will then be equivalent if B can be obtained from A by starting with A, and applying in turn a finite sequence of operations (1). That is, A and B are equivalent if B is an assertion in the "canonical system"[2] with primitive assertion A and operations (1). Thue's general problem thus becomes the decision problem for the class of all canonical systems of this "Thue type."

This general problem could easily be proved recursively unsolvable if, instead of the pair of operations for each i of (1), we merely had the first operation of each pair.[3] In fact, by direct methods such as those of [3], we easily reduce the decision problem of an arbitrary "normal system" [3] to the decision problem of such a system of "semi-Thue type," the known recursive unsolvability of the

Reprinted from THE JOURNAL OF SYMBOLIC LOGIC, vol. 12 (1947) pp. 1-11.

[1] Numbers in brackets refer to the bibliography at the end of the paper.

[2] Null assertions, however, now being allowed.

[3] That is, using the language of propositions instead of operations, if we merely had an implication where (1) has an equivalence.

certain length conditions being equivalent to the mere existence of a solution of that string equation. In the present paper the modification is a complication, the answer to $hQ'h$ being or not being an assertion of T''' being unaffected by adding to the operations of T''' their inverses.

The writer has often felt that the multiplicity of equivalent formulations of recursiveness has been a deterrent to the general promulgation of this discipline. Yet, the writer's normal systems naturally lead to the unsolvable problem of [5], while the deterministic character of the Turing machine is basic to the above unsolvability proof. From this point of view, the several formulations of recursiveness are so many different instruments for tackling new unsolvability proofs.

Though we have not paused to verify this formally, it seems rather obvious that when the problem of [5], and the problem of Thue, are translated via positive integers as suggested in [4], they become decision problems of recursively enumerable sets of positive integers of the same degree of unsolvability as the complete set K, at worst, with respect to many-one reducibility [4]. This indicates how far practice lags behind theory in this field.

Appendix. The following critique of Turing's "computability" paper [7] concerns only pp. 230–248[ii] thereof. We have checked the work through the construction of the "universal computing machine" in detail;[11] but the proofs of the two theorems in the section following are there given in outline only, and we have not supplied the formal details. *We have therefore also left in intuitive form the proofs of the statements on recursiveness, and alternative procedures, we make below.*

Turing's definition of an arbitrary machine is not completely given in his paper, and, at a number of points, has to be inferred from his development. In the first instance his machine is a "computing machine" for obtaining the successive digits of a real number in dyadic notation, and, in that case, starts operating on a blank tape. Where explicitly stated, however, the machine may

[11] One major correction is needed. To the instructions for $\mathfrak{con}_1(\mathfrak{C}, \alpha)$ p. 244, add the line: None *PD*, *R*, *Pα*, *R*, *R*, *R* $\mathfrak{C}$. This is needed to introduce the representation *D* of the blank scanned square when, as at the beginning of the action of the machine, or due to motion right beyond the rightmost previous point, the complete configuration ends with a *q*, and thus make the $\mathfrak{fmp}$ of p. 244 correct. We may also note the following minor slips and misprints in pp. 230–248. Page 236, to the instructions for $\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$ add the line: None *L* $\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$; p. 240 and p. 241, the S.D should begin, but not end, with a semicolon; p. 242, omit the first *D* in (C_2); p. 243, last paragraph, add ":" to the first list of symbols; pp. 244–246, replace $\mathfrak{g}$ by $\mathfrak{q}$; p. 245, in the instruction for $\mathfrak{mf}$, $\mathfrak{mf}$ should be $\mathfrak{mf}_1$; p. 245, in the second instruction for $\mathfrak{sjm}_2$, replace the first *R* by *L*; p. 245, in the first instruction for $\mathfrak{sh}_2$, replace $\mathfrak{sh}_2$ by $\mathfrak{sh}_3$. A reader of the paper will be helped by keeping in mind that the "examples" of pages 236–239 are really parts of the table for the universal computing machine, and accomplish what they are said to accomplish not for all possible printings on the tape, but for certain ones that include printings arising from the action of the universal computing machine. In particular, the tape has $\mathfrak{d}$ printed on its first two squares, the occurrence of two consecutive blank squares insures all squares to the right thereof being blank, and, usually, symbols referred to are on "*F*-squares," and obey the convention of p. 235.[i]

i For page references in this footnote, the reader should subtract 114 from the page given in order to obtain the corresponding page in this anthology.

ii 116–134

start operating on a tape previously marked. From Turing's frequent references to the beginning of the tape, and the way his universal computing machine treats motion left, we gather that, unlike our tape, this tape is a one-way infinite affair going right from an initial square.

Primarily as a matter of practice, Turing makes his machines satisfy the following convention. Starting with the first square, alternate squares are called F-squares, the rest, E-squares. In its action the machine then never directs motion left when it is scanning the initial square, never orders the erasure, or change, of a symbol on an F-square, never orders the printing of a symbol on a blank F-square if the previous F-square is blank and, in the case of a computing machine, never orders the printing of 0 or 1 on an E-square. This convention is very useful in practice. However the actual performance, described below, of the universal computing machine, coupled with Turing's proof of the second of the two theorems referred to above, strongly suggests that Turing makes this convention part of the definition of an arbitrary machine. We shall distinguish between a Turing machine and a Turing convention-machine.

By a uniform method of representation, Turing represents the set of instructions, corresponding to our quadruplets,[12] which determine the behavior of a machine by a single string on seven letters called the standard description (S.D) of the machine. With the letters replaced by numerals, the S.D of a machine is considered the arabic representation of a positive integer called the description number (D.N) of the machine. If our critique is correct, a machine is said to be circle-free if it is a Turing computing convention-machine which prints an infinite number of 0's and 1's.[13] And the two theorems of Turing's in question are really the following. There is no Turing convention-machine which, when supplied with an arbitrary positive integer n, will determine whether n is the D.N of a Turing computing convention-machine that is circle-free. There is no Turing convention-machine which, when supplied with an arbitrary positive integer n, will determine whether n is the D.N of a Turing computing convention-machine that ever prints a given symbol (0 say).[14]

In view of [8], these "no machine" results are no doubt equivalent to the re-

[12] Our quadruplets are quintuplets in the Turing development. That is, where our standard instruction orders either a printing (overprinting) or motion, left or right, Turing's standard instruction always orders a printing and a motion, right, left, or none. Turing's method has certain technical advantages, but complicates theory by introducing an irrelevant "printing" of a symbol each time that symbol is merely passed over.

[13] "Genuinely prints," that is, a genuine printing being a printing in an empty square. See the previous footnote.

[14] Turing in each case refers to the S.D of a machine being supplied. But the proof of the first theorem, and the second theorem depends on the first, shows that it is really a positive integer n that is supplied. Turing's proof of the second theorem is unusual in that while it uses the unsolvability result of the first theorem, it does not "reduce" [4] the problem of the first theorem to that of the second. In fact, the first problem is almost surely of "higher degree of unsolvability" [4] than the second, in which case it could not be "reduced" to the second. Despite appearances, that second unsolvability proof, like the first, is a *reductio ad absurdum* proof based on the definition of unsolvability, at the conclusion of which, the first result is used.

cursive unsolvability of the corresponding problems.[15] But both of these problems are infected by the spurious Turing convention. Actually, the set of n's which are D.N's of Turing computing machines as such is recursive, and hence the condition that n be a D.N offers no difficulty. But, while the set of n's which are not D.N's of convention-machines is recursively enumerable, the complement of that set, that is, the set of n's which are D.N's of convention-machines, is not recursively enumerable. As a result, in both of the above problems, neither the set of n's for which the question posed has the answer yes, nor the set for which the answer is no, is recursively enumerable.

This would remain true for the first problem even apart from the convention condition. But the second would then become that simplest type of unsolvable problem, the decision problem of a non-recursive recursively enumerable set of positive integers [4]. For the set of n's that are D.N's of unrestricted Turing computing machines printing 0, say, is recursively enumerable, though its complement is not. The Turing convention therefore prevents the early appearance of this simplest type of unsolvable problem.

It likewise prevents the use of Turing's second theorem in the above unsolvability proof of the problem of Thue. For in attempting to reduce the problem of Turing's second theorem to the problem of Thue, when an n leads to a Thue question for which the answer is yes, we would still have to determine whether n is the D.N of a Turing convention-machine before the answer to the question posed by n can be given, and that determination cannot be made recursively for arbitrary n. If, however, we could replace the Turing convention by a convention that is recursive, the application to the problem of Thue could be made. An analysis of what Turing's universal computing machine accomplishes when applied to an arbitrary machine reveals that this can be done.

The universal computing machine was designed so that when applied to the S.D of an arbitrary computing machine it would yield the same sequence of 0's and 1's as the computing machine as well as, and through the intervention of, the successive "complete configurations"—representations of the successive states of tape versus machine—yielded by the computing machine. This it does for a Turing convention-machine.[16] For an arbitrary machine, we have to interpret a direction of motion left at a time when the initial square of the tape is scanned as meaning no motion.[17] The universal computing machine will then yield again the correct complete configurations generated by the given machine. But *the space sequence of* 0*'s and* 1*'s printed by the universal computing machine will now be identical with the time sequence of those printings of* 0*'s and* 1*'s by the given machine that are made in empty squares.* If, now, instead of Turing's

[15] Our experience with proving that "normal unsolvability" in a sense implicit in [3] is equivalent to unsolvability in the sense of Church [1], at least when the set of questions is recursive, suggests that a fair amount of additional labor would here be involved. That is probably our chief reason for making our proof of the recursive unsolvability of the problem of Thue independent of Turing's development.

[16] Granted the corrections given in footnote 11.

[17] This modification of the concept of motion left is assumed throughout the rest of the discussion, with the exception of the last paragraph.

convention we introduce the convention that the instructions defining the machine never order the printing of a 0 or 1 except when the scanned square is empty, or 0, 1 respectively, and never order the erasure of a 0 or 1, Turing's arguments again can be carried through. And this "(0, 1) convention," being recursive, allows the application to the problem of Thue to be made.[18] Note that if a machine is in fact a Turing convention-machine, we could strike out any direction thereof which contradicts the (0, 1) convention without altering the behavior of the machine, and thus obtain a (0, 1) convention-machine. But a (0, 1) convention-machine need not satisfy the Turing convention. However, by replacing each internal-configuration q_i of a machine by a pair q_i, q_i' to correspond to the scanned square being an F- or an E-square respectively, and modifying printing on an F-square to include testing the preceding F-square for being blank, we can obtain a "(q, q') convention" which is again recursive, and usable both for Turing's arguments and the problem of Thue, and has the property of, in a sense, being equivalent to the Turing convention. That is, every (q, q') convention-machine is a Turing convention-machine, while the directions of every Turing convention-machine can be recursively modified to yield a (q, q') convention-machine whose operation yields the same time sequence and spatial arrangement of printings and erasures as does the given machine, except for reprintings of the same symbol in a given square.

These changes in the Turing convention, while preserving the general outline of Turing's development and at the same admitting of the application to the problem of Thue, would at least require a complete redoing of the formal work of the proof of the second Turing theorem. On the other hand, very little added formal work would be required if the following changes are made in the Turing argument itself, though there would still remain the need of extending the equivalence proof of [8] to the concept of unsolvability. By using the above result on the performance of the universal computing machine when applied to the S.D of an arbitrary machine, we see that Turing's proof of his first theorem, whatever the formal counterpart thereof is, yields the following theorem. There is no Turing convention-machine which, when supplied with an arbitrary positive integer n, will determine whether n is the D.N of an arbitrary Turing machine that prints 0's and 1's in empty squares infinitely often. Now given an arbitrary positive integer n, if that n is the D.N of a Turing machine $\mathfrak{M}$, apply the universal computing machine to the S.D of $\mathfrak{M}$ to obtain a machine $\mathfrak{M}^*$. Since $\mathfrak{M}^*$ satisfies the Turing convention, whatever Turing's formal proof of his second theorem is, it will be usable intact in the present proof, and, via the new form of his first theorem, will yield the following usable result. There is no machine which, when supplied with an arbitrary positive integer n, will

[18] So far as recursiveness is concerned, the distinction between the Turing convention and the (0, 1) convention is that the former concerns the history of the machine in action, the latter only the instructions defining the machine. Likewise, despite appearances, the later (q, q') convention.

determine whether n is the D.N of an arbitrary Turing machine that ever prints a given symbol (0 say).[19]

These alternative procedures assume that Turing's universal computing machine is retained. However, in view of the above discussion, it seems to the writer that Turing's preoccupation with computable numbers has marred his entire development of the Turing machine. We therefore suggest a redevelopment of the Turing machine based on the formulation given in the body of the present paper. This could easily include computable numbers by defining a computable sequence of 0's and 1's as the *time sequence* of printings of 0's and 1's by an arbitrary Turing machine, provided there are an infinite number of such printings. By adding to Turing's complete configuration a representation of the act last performed, a few changes in Turing's method would yield a universal computing machine which would transform such a time sequence into a space sequence. Turing's convention would be followed as a matter of useful practice in setting up this, and other, particular machines. But it would not infect the theory of arbitrary Turing machines.

References

[1] Alonzo Church, *An unsolvable problem of elementary number theory*, ***American journal of mathematics,*** vol. 58 (1936), pp. 345–363.[a]

[2] Emil L. Post, *Finite combinatory processes—formulation 1*, this JOURNAL, vol. 1 (1936), pp. 103–105.[a]

[3] Emil L. Post, *Formal reductions of the general combinatorial decision problem*, ***American journal of mathematics,*** vol. 65 (1943), pp. 197–215.

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

[5] Emil L. Post, *A variant of a recursively unsolvable problem*, ibid., vol. 52 (1946), pp. 264–268.

[6] Axel Thue, *Probleme über Veränderungen von Zeichenreihen nach gegebenen Regeln*, ***Skrifter utgit av Videnskapsselskapet i Kristiania,*** I. Matematisk-naturvidenskabelig klasse 1914, no. 10 (1914), 34 pp.

[7] A. M. Turing, *On computable numbers, with an application to the Entscheidungsproblem*, ***Proceedings of the London Mathematical Society,*** ser. 2 vol. 42 (1937), pp. 230–265.[a]

[8] A. M. Turing, *Computability and λ-definability*, this JOURNAL, vol. 2 (1937), pp. 153–163.

[19] It is here assumed that the suggested extension of [8] includes a proof to the effect that the existence of an arbitrary Turing machine for solving a given problem is equivalent to the existence of a Turing convention-machine for solving that problem.