THE JOURNAL OF SYMBOLIC LOGIC
Volume 12, Number 1, March 1947

RECURSIVE UNSOLVABILITY OF A PROBLEM OF THUE

EMIL L. POST

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:

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

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

Received October 26, 1946. Presented to the American Mathematical Society November 2, 1946.

[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.

decision problem for the class of all normal systems then, no doubt, leading to the recursive unsolvability of the decision problem for the class of all semi-Thue systems. The crux of our method for handling the Thue systems themselves is to find such a reduction of a known unsolvable problem to a system of semi-Thue type that when, for each i, the second of the two operations in (1) is added to the semi-Thue system, no new assertions are thereby added to the system. The known unsolvable problem is thus reduced to the resulting Thue system, as desired. Such a reduction turns out to be possible for a certain unsolvable problem arising in the theory of Turing machines.

We shall adopt the following formulation of a Turing machine [7].[4] A two-way infinite linear tape is provided, ruled off into squares. Time is a one-way infinite sequence of discrete moments. A square will either be blank, or have at most one symbol printed upon it. At any moment the machine "scans" one of the squares. At such a moment the machine is capable of performing one of the following atomic acts: moving one square to the left, moving one square to the right, printing on the scanned square one of a given finite number of symbols $S_1, \cdots, S_m$, or a blank. Following Turing, we take "printing" here to mean "overprinting," that is, the letter or blank printed replaces any letter that may have been on the scanned square. Printing a blank is then equivalent to erasing, when the scanned square is not blank. The machine, furthermore, is capable of assuming but a finite number of internal states, internal configurations or m-configurations with Turing, $q_1, q_2, \cdots, q_R$. At any moment, the letter or blank on the scanned square together with the internal configuration of the machine determines the atomic act to be performed by the machine and the new internal configuration of the machine, or else, the machine then stops. At the initial moment a finite, possibly null, number of squares have S's printed on them, the machine scans a particular square and has a particular internal configuration.

Symbolically, the machine may be given as follows. Let S_0 be used to represent a blank square. For the start of the action of the machine we need only consider the smallest unbroken piece of the tape containing the initially marked squares and the scanned square, replace these squares by their markings, or by S_0 if blank, and insert the symbol q_{i_1} of the initial internal configuration prior to the S of the scanned square to yield the representation

$$S_{j_1}S_{j_2}\cdots S_{j_{k-1}}q_{i_1}S_{j_k}\cdots S_{j_K}. \tag{2}$$

A finite number of quadruplets of symbols of the three forms,

$$q_iS_jLq_l, \qquad q_iS_jRq_l, \qquad q_iS_jS_kq_l, \tag{3}$$

will then determine the behavior of the machine. Here, q_i and S_j represent the internal configuration of the machine, and the symbol or blank on the scanned

[4] Apart from the Turing convention, discussed in the appendix, this differs from Turing's formulation of an automatic machine in the nature of the tape, and in Turing's use, in his standard form [7, p. 240], of the composite operation "print and move" where we just have "move." A number of comparisons with [2] will occur to a reader of that note.

square, at any moment; L, R, or S_k the correspondingly determined atomic act of motion left, right, or printing of S_k ; q_l the determined new internal configuration of the machine. It is fundamental that the pairs q_i , S_j of the several quadruplets are distinct, for they are to determine uniquely the consequent behavior of the machine. The machine will then continue acting deterministically from the initial moment on unless, and until, a q_iS_j is reached for which there is no quadruplet (3), in which case it will stop.

We can now readily set up a semi-Thue system whose assertions will represent the successive states of the tape, and the relation of the machine thereto, as (2) represented these at the initial moment. However, for simplicity, the portion of the tape represented, while including the marked squares and the scanned square, need not now be the smallest such portion.[5] Because of the particular needs of the semi-Thue form, we introduce a new symbol h. Each assertion of the semi-Thue system will then be in the form hPh with P free from h. If A represents the string (2) of S's and one q, the initial assertion of the semi-Thue system will be hAh. For each quadruplet (3) corresponding to moving one square to the left, we introduce the operations

$$PS_nq_iS_jQ \text{ produces } Pq_lS_nS_jQ, n = 0, 1, \cdots, m, \tag{4}$$

$$Phq_iS_jQ \text{ produces } Phq_lS_oS_jQ. \tag{5}$$

Note that (4) takes care of all cases where the scanned square is not the leftmost square of the part of the tape represented at the given moment, (5) where the scanned square is that leftmost square. Due to the form hPh of all assertions of the system, when (5) is applicable, the h of the premise thereof must be the leftmost of these two h's, so that P will be identified with the null string. The S_o of the conclusion then takes care of the necessary extension of the portion of the tape represented when the motion is one square to the left of that portion. Likewise, for each quadruplet (3) corresponding to motion of one square to the right we introduce

$$Pq_iS_jS_nQ \text{ produces } PS_jq_lS_nQ, n = 0, 1, \cdots, m, \tag{6}$$

$$Pq_iS_jhQ \text{ produces } PS_jq_lS_ohQ; \tag{7}$$

while for each quadruplet (3) corresponding to the printing of S_k over the scanned square we have

$$Pq_iS_jQ \text{ produces } Pq_lS_kQ. \tag{8}$$

Clearly both premise and conclusion of each operation thus introduced is of the form PBQ with fixed B, so that we do thus have a semi-Thue system. An obvious induction yields the form hPh with P free from h for each assertion. Likewise, each assertion has one and only one q therein. Finally, it is readily

[5] It could be made the smallest such portion by using more operations. There would then be a 1–1 correspondence between the intrinsic states of tape versus machine and the representations thereof.

verified from the deterministic character of the Turing machine, and from the forms of the above operations, that at most one of these operations is applicable to any string having no more than one occurrence of a q therein, and then in only one way.

The unsolvable problem that is to yield the unsolvability of the problem of Thue would seem to be furnished by the following result of Turing's [7, p. 248]: "There can be no machine $\mathfrak{H}$ which, when supplied with the S.D of an arbitrary machine $\mathfrak{M}$, will determine whether $\mathfrak{M}$ ever prints a given symbol (0 say)." There are, however, difficulties in using this result as given due to peculiarities of Turing's development. (The matter is discussed in the appendix.) We therefore proceed independently of Turing as follows.

We start with the known recursive unsolvability of the decision problem for the class of normal systems on two letters a, b.[6] It suffices here to think of this problem as consisting of a class of questions, each question Q being symbolized by a string on a given finite set of letters. By methods such as those used by Turing in setting up his universal computing machine [7], we then set up the quadruplets (3) of a fixed Turing machine with certain letters S_1, S_2, $\cdots$, S_m, and internal configurations q_1, q_2, $\cdots$, q_R, and give an effective method for translating each question Q into a Q' of form (2) to serve as the initial state of tape versus machine, the construction being such that the following is true. The answer to question Q is yes, or no, according as the constructed machine, when applied to Q', does, or does not, in the course of its operation print a certain fixed letter S_p. This letter S_p is not present in the Q' of any Q. Since such methods are fully exploited by Turing in [7], we do not give the details of this construction.[7]

Now, given Q, form the semi-Thue system T' with initial assertion $hQ'h$, and operations (4)–(8) corresponding to this Turing machine. Then, the answer to Q is yes, or no, according as some assertion of T' involves the letter S_p, or no assertion involves that letter. We now modify T' as follows. Delete all operations in T' such that the S_j of the premise, the symbol on the scanned square of the Turing machine, is S_p. Since, when S_p is first printed, it can appear so only as the S_k of (8), and thus would be the S_j of a next operation, the deductive processes of the semi-Thue system will now stop the first time S_p appears in an assertion. We now add operations which, in deterministic fashion, will erase all of this assertion except for the two h's and the q, while changing this q. For this purpose, we introduce two new "internal configurations"

[6] See [5, footnote 2]. The specific form of this problem, however, need not be known by the reader for an understanding of the present argument.

[7] This work was carried through before the definitive study of Turing's paper [7], referred to in the appendix, was made. As a result, some differences of method appear. A minor difference is that where Turing uses the method of "marking" a sequence of symbols [7, p. 235] to distinguish it, we introduce the *effect* of movable physical markers; two, indeed, suffice. A major difference is that instead of the m-configuration functions of Turing's skeleton tables [7, p. 236], we introduce a symbolism and technique based on the concept of a subset of directions of a given set of directions. Both differences were suggested by [2]. They may, perhaps, better be exploited in a more general setting.

q_{R+1} and q_{R+2}. We further alter the operations of T' by changing the q_l of each operation (8) for which S_k is S_p to q_{R+1}, and add the following operations:

$$PS_nq_{R+1}Q \text{ produces } Pq_{R+1}Q,\ n = 0, 1, \cdots, m;\ n \neq p. \tag{9}$$

$$Phq_{R+1}Q \text{ produces } Phq_{R+2}Q. \tag{10}$$

$$Pq_{R+2}S_nQ \text{ produces } Pq_{R+2}Q,\ n = 0, 1, \cdots, m. \tag{11}$$

Note that as a result of the previous changes, when S_p first appears in an assertion, the q therein is q_{R+1}. Operations (9) then serve to erase the S's of that assertion to the left of q_{R+1}, (10) then changes q_{R+1} to q_{R+2}, (11) erases the S's to the right of q_{R+2}. Finally, therefore, the assertion becomes $hq_{R+2}h$, to which no further operation is applicable. Call the resulting semi-Thue system T''. Clearly, for T'' it is also true that at most one of its operations is applicable to any string having no more than one occurrence of a q therein, and then in only one way. It follows that the answer to Q is yes, or no, according as $hq_{R+2}h$ is, or is not, an assertion in T'', the operations of T'' operating one by one in deterministic fashion, and, in the former case, terminating in $hq_{R+2}h$.

The proof of the reducibility of our initial unsolvable problem to the problem of Thue essentially becomes the proof of the following two lemmas.[8] By the *inverse* of an operation of the form *PAQ produces PBQ* we shall mean the operation *PBQ produces PAQ*. Let T''' be the semi-Thue system with primitive assertion $hq_{R+2}h$ and operations the inverses of those of T''. We then have:

LEMMA I. The primitive assertion $hq_{R+2}h$ of T''' is an assertion of T'' when, and only when, the primitive assertion $hQ'h$ of T'' is an assertion of T'''.

Proof. D is a result of applying "*PAQ produces PBQ*" to C when, and only when, C is a result of applying the inverse operation "*PBQ produces PAQ*" to D. For both statements are equivalent to the existence of strings P and Q such that $PAQ = C$, $PBQ = D$. If, then, operations $O_1, O_2, \cdots, O_n$ of T'' lead from its primitive assertion $hQ'h$ through assertions $C_1, C_2, \cdots, C_{n-1}$ to the assertion $hq_{R+2}h$, the inverses of these operations, all in T''', will in reverse order lead from $hq_{R+2}h$, the primitive assertion of T''', through $C_{n-1}, \cdots, C_2, C_1$ to $hQ'h$; and conversely.

As a result of Lemma I, the answer to question Q is yes, or no, according as $hQ'h$ is, or is not, an assertion of T'''. Note that while the initial assertion of T'' depended on Q, T''' is the same for all Q's. Now let T be the Thue system obtained from the semi-Thue system T''' by adding to the latter the inverse of each of its operations. We then have:

LEMMA II. The class of assertions of T is identical with the class of assertions of T'''.

Proof. Each assertion of T''' is, of course, an assertion of T. For the converse, let operations $O_1, O_2, \cdots, O_n$ of T lead from its primitive assertion $hq_{R+2}h$, through assertions $C_1, C_2, \cdots, C_{n-1}$, to an assertion C of T. If n is zero, C is $hq_{R+2}h$, and hence an assertion of T'''. Otherwise, note that the operations of T, being those of T''' and their inverses, are the combined operations of T'''

[8] These lemmas can be made more general.

and of T''. Now we saw that no operation of T'' is applicable to $hq_{R+2}h$, the deductive processes of T'' terminating in $hq_{R+2}h$ if leading thereto. Hence, operation O_1 must be in T'''. Assume O_{m+1} to be the first O not in T''', and hence in T''. Since O_m is in T''', its inverse is in T''. As O_m operates on $C_{m-1}(hq_{R+2}h$, if m is one) to yield C_m, the inverse of O_m is applicable to C_m yielding C_{m-1}. That is, both the inverse of O_m, and O_{m+1}, are operations in T'' applicable to C_m, and yielding C_{m-1} and C_{m+1} (C, if $m + 1 = n$) respectively. Now, since the premise and conclusion of each operation in T, and the primitive assertion of T, have exactly one occurrence of a q therein, the same is true of every assertion of T. But we saw that at most one operation of T'' is applicable to a string with a single occurrence of a q therein, and then in only one way. It follows that O_{m+1} is in fact that inverse of O_m, and hence C_{m+1} is C_{m-1} all over again. We may therefore delete operations O_m and O_{m+1} from the given sequence of operations, and still have a sequence of operations leading from $hq_{R+2}h$ to C. By repeating this process, we finally obtain such a sequence of O's with each O in T'''. The arbitrary assertion C of T is therefore also an assertion of T'''.[9]

Hence, $hQ'h$ is an assertion of T''' when and only when it is an assertion of T. Finally, then, the answer to Q is yes, or no, according as $hQ'h$ is, or is not, an assertion of the Thue system T. In terms of the language of Thue, we have then a fixed set of pairs of strings $(A_1, B_1), \cdots, (A_n, B_n)$ leading to a definition of equivalence of strings such that the answer to Q is yes, or no, according as $hQ'h$ is, or is not, equivalent to the fixed string $hq_{R+2}h$.[10] Certainly, then, a solution of the problem of Thue in its full generality would thus lead to a solution of the "decision problem for the class of normal systems on a, b." By the use of Gödel representations, the recursive unsolvability of the latter problem then easily leads to the recursive unsolvability of the problem of Thue.

A few concluding remarks may be in order. The methods of [5], and of the present paper, do have something in common, a something we may call *the method of the irrelevant modification*. Once an unsolvable problem has been obtained by a *reductio ad absurdum* argument based on the definition of solvability, the usual method of proving a new problem unsolvable is to reduce a known unsolvable problem to this given problem. In the method of the irrelevant modification, the known unsolvable problem is reduced to a problem which on modification becomes the given problem, while that modification does not affect the answers to the individual questions. In [5] the modification is a simplification, the existence of a solution of a certain string equation subject to

[9] Briefly, then, the effect of operations of T'' on deductive processes of T is to unravel work done by operations of T'''. Note that while the deductive processes of T'' give rise to a single sequence of assertions starting with $hQ'h$ and terminating in $hq_{R+2}h$, if leading thereto, the deductive processes of T''' give rise to a tree of assertions, elements not necessarily distinct, stemming from $hq_{R+2}h$, and containing the above sequence in reverse when that sequence terminates in $hq_{R+2}h$.

[10] By the method of the next to the last paragraph of [5], this definition of equivalence could be transformed into a definition of equivalence for strings on the two letters a, b. We have not paused to prove the recursive unsolvability of the resulting special case of the problem of Thue.

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 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.

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.

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

[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.

[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.

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

THE CITY COLLEGE
COLLEGE OF THE CITY OF NEW YORK

[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.