J. Soc. Indust. Appl. Math.
Vol. 7, No. 1, March, 1959
Printed in U.S.A.

THE PRESENT THEORY OF TURING MACHINE COMPUTABILITY*

HARTLEY ROGERS, Jr.

Thank you for inviting me to be present at the SIAM meetings and to speak to you this morning. Subject to the constraints of a single hour lecture, I should like to give you an informal introduction to Turing machine computability or, as I shall prefer to call it, the theory of *general effective computability*. Here, the phrase "effective computability", as you might guess, means that we have to do with operations of the sort performable by digital computers under explicit deterministic programs of instructions. The word "general" means that we have to do with the theory that is obtained when we remove all limitations of either time or memory upon the action of such computers, and it means that within this theory we have to do with questions of existence or nonexistence of computer methods, rather than with matters of efficiency and good design. Such a theory, obviously, will be much simpler than a theory that takes these latter, most practical matters into account. In fact, we might suspect: (i) that such a theory would be so simple as to be uninteresting; and moreover, (ii) that it would be divested of exactly those features which could give it practical significance. I hope to convince you this morning that the first of these suspicions is wrong, and that we obtain a subject matter unusually rich, complex and intriguing. The second suspicion, I shall now admit, is largely true. I can only claim that our theory is a kind of "asymptotic form" of more difficult, realistic theories, and that as such it has, on occasion, proved suggestive and stimulating to researchers in the practical computer field. The main defense for our theory, however, is the same as that for, let us say, such a mildly esoteric subject as the set theory that has grown out of analysis—namely, that it has a naturalness, beauty and relevance to existing mathematics that justify it in itself as an object of study.

My talk will fall into two parts, approximately reflecting two main phases in the historical development of its subject. Part I will outline basic concepts and results that were formulated and obtained in the decade prior to 1943. Part II will suggest, in a limited way, some of the further developments and applications that have appeared at an accelerating pace up to the present time.

Part I

In this part we look at a certain class of mathematical objects, the *recursive functions*, and at various equivalent formal definitions from which

* Received by the editors October 3, 1958. Presented by invitation to SIAM at its summer meeting at Pennsylvania State University, August 1957.

this class can be obtained. To fix our terminology, let N be the set of non-negative integers. Henceforth, the words "number" and "integer" shall refer to members of this set. Consider the class of all mappings of N into itself. The word "function" shall refer to members of this class. "x", "y", "z", $\cdots$ shall denote numbers. "f", "g", "h", $\cdots$ shall denote functions. The recursive functions will constitute a certain subclass of the functions.

Since the formal definitions are too long for us to give them careful treatment, we adopt a compromise. We suggest the content of these definitions through an informal and somewhat anthropomorphic diagram. I believe that all the intuitive essentials will be preserved and that you will be able to grasp without ambiguity the significant results in this first phase of the theory.

Consider a box B inside of which we have a man L with a desk, pencils and paper. On one side B has two slots, marked *input* and *output*. If we write a number on paper and pass it through the input slot, L takes it and begins performing certain computations. If and when he finishes, he writes down a number obtained from the computation and passes it back to us through the output slot. Assume further that L has with him explicit deterministic instructions of finite length as to how the computation is to be done. We refer to these instructions as P. Finally, assume that the supply of paper is inexhaustible, and that B can be enlarged in size so that an arbitrarily large amount of paper work can be stored in it in the course of any single computation. (Indeed, this elasticity might be needed just to store the input number, if that number were sufficiently large.) I think we had better assume, too, that L himself is inexhaustible, since we do not care how long it takes for an output to appear, provided that it does eventually appear after a finite amount of computation. We refer to the system B-L-P as M.

If there is an output for every input, M *represents* (in the obvious sense) a function. It is important to keep in mind that a given function might be representable in several different ways, that is to say, obtainable through several different instructions P. Several examples will reinforce or clarify our picture so far. The function [$f_1(x) = 2x$] is obviously representable. Similarly, the function [$f_2(x) =$ the xth digit in decimal expansion of π] is representable, (though, of course, we would expect that a P for f_2 would be rather longer than the simplest P for f_1). Consider next the function [$f_3(x) = 1$ if a run of *exactly* x successive 7's occurs somewhere in the decimal expansion of π; $f_3(x) = 0$ otherwise]. No one knows whether f_3 is representable, since no one knows whether finite instructions exist for computing it. In contrast to f_3 consider the function [$f_4(x) = 1$ if a run of *at least* x successive 7's occurs somewhere in the decimal expansion of π; $f_4(x) = 0$ otherwise]. After a moment's reflection, you will see that f_4 must either

be the constant function $[f(x) = 1]$ or it must be a function of the form $[f(x) = 1 \text{ for } x \leqq k; f(x) = 0 \text{ for } x > k]$ for some integer k. Whichever case occurs, instructions *exist* for computing f_4. Hence f_4 is representable, though we do not know at this time how to identify the correct P.

There remains a major area of vagueness in our definition. It centers on the question: exactly what constitutes admissible instructions P, and exactly how is the behavior of L to depend upon P? Until this is settled, we cannot treat such further questions as: is every function representable; are only a countable infinity of functions representable? It is indeed conceivable that there might be no single satisfactory class of instructions and that any precise definition could be augmented in such a way as to represent functions not previously representable. Several alternative ways to resolve this vagueness were developed in the 1930's in the work of Church, Gödel, Kleene, Post, Turing and others. Any one of these ways can be summarized as follows. First, a finite symbolism (that is, a finite alphabet and precise rules for making arbitrarily long formulas) is given, and an admissible P is taken to be any finite sequence of formulas from this symbolism. Let us call this the *P-symbolism*. Secondly, a finite set of fixed specifications—let us call them *L-P specifications*—is given which stipulate how the behavior of L is to depend upon P. We shall not go more deeply into the L-P specifications, except to remark that they remain constant as P is varied, that they prescribe for L certain simple digital bookkeeping operations, and that these operations are related to the computation and to P in an elementary symbol-at-a-time way so that L can deal with formulas of arbitrary length.[1]

In the approaches of Church and of Kleene, the P-symbolism is such that any admissible P consists of a sequence of simple functional equations, and the L-P specifications are such that L proceeds by using the input

[1] For the curious reader, we elaborate somewhat further. In all of the known approaches, the L-P specifications can be reduced to the following. A finite symbolism is specified for the computations and for the input and output numbers. A rather small set of bookkeeping operations is prescribed for L. These include operations of writing down symbols, operations of moving one symbol at a time backward or forward in the computation to or from symbols previously written, operations of moving backward or forward in P, and terminal operations for transcribing and discharging from B a possible eventual output number. L is also endowed with a fixed finite short-term memory which at any point preserves symbols written or examined on certain preceding operations, and L is provided with a finite set of simple rules according to which the bookkeeping operation next to be performed and the next state of his short-term memory are uniquely determined by the contents of the short-term memory taken together with the symbol written or examined last. Given any input number, L begins by writing that number into his computation and then examining the first symbol in the first formula of P.

number in a sequence of substitution operations among the given equations. In the approach of Turing, the symbolism and specifications are such that the entire B-L-P system can be viewed as a digital computer (with the unlimited supply of paper taking the form of a tape running through the computer). Roughly, to use modern computing terms, L becomes the logical component of the computer, and P becomes its program. In Turing's approach, the entire system M is hence called a *Turing machine.*

Having settled on any such P-symbolism and L-P specifications, we can answer some of the questions mentioned above. The class of functions representable is at most countably infinite, since the class of possible different P's is countably infinite; furthermore, not all functions are representable, since the class of all functions is uncountable. The question remains, how does the class of functions representable depend upon the apparently quite arbitrary choice of symbolism and specifications? Clearly if we start with very limited symbolism and specifications, we may obtain no functions at all, or only a finite number. What, for instance are the relations of the classes generated by the Church, the Kleene and the Turing approaches? How, for instance, do these classes become larger if the symbolism and specifications are augmented? What is the relation of these classes to functions which from an intuitive point of view, mathematicians generally agree to call "effectively computable"? These matters were the subject of energetic investigation in the first period of our theory. Since any precise choice of symbolism and specifications gives a precisely defined class of functions, the investigations took the form of detailed mathematical study. The results were, to a certain extent, unexpected. We now summarize them as a single *basic result.*

Basic Result. First it was shown that the class of functions representable under each of the Church, Kleene and Turing approaches is the same, and is, moreover, the same as that obtained under several other suggested approaches. Second, it was shown that, over certain very broad families of enlargements of the symbolism and specifications, the class remains unchanged. In fact, if certain very reasonable criteria are laid down for what may constitute a symbolism and specifications, it can be demonstrated that the class of functions obtained is always a subclass of the "maximal" class of Church, Kleene and Turing. Third, many functions agreed to be intuitively effective were investigated, and all were shown to be members of the maximal class.

Thus, by a path that is highly noninvariant (i.e. dependent on arbitrary choices), we appear to have arrived at a natural and significant class of functions. They are called the *recursive functions.* The third part of the basic result suggests that we identify the informal intuitive notion of effectiveness with the precise concept of recursiveness. This proposal is

known as *Church's Thesis*. So amply has the thesis been vindicated by detailed investigation of particular cases that, in the literature today, an author often concludes that a particuar function is recursive without giving detailed argument. The situation is analogous to the use of formal set theoretic logic in ordinary mathematics. The investigator often omits much detail, but he must be prepared to supply it if challenged. For brevity, in the remainder of this talk, we shall often go directly from intuitive effectiveness to an assertion of recursiveness. The listener should remember, however, that we have some standard choice of P-symbolism and L-P specifications in mind, and that these give our results precise mathematical content.

There is an extension of the basic result that is useful in further work. The listener will have noted that it is easy to think of procedures which are intuitively effective in the sense that they lead to unique computations, but which, through failure to terminate for some inputs, do not represent functions. Let us use the intuitive notion *quasi-effective* to include all such procedures together with those effective procedures that do represent functions. By a *quasi-function* let us mean a mapping which is defined on some subset of N (possibly empty, possibly $= N$) into N. Clearly any P represents a quasi-function, though it may not, in general, represent a function. The following is now the extension of the basic result: all statements of parts one, two and three of the basic result hold, with "quasi-function" substituted for "function" and with "quasi-effective" substituted for "effective". Thus we have a natural maximal class of *recursive quasi-functions*,[2] and, moreover, an extended version of Church's Thesis; all of the comments made above about Church's Thesis and about our use of it continue to apply verbatim to this extended case.

We now turn to a closer study of the relation between P and the behavior of M. First we observe that there is an effective procedure for enumerating, one at a time, all possible instructions P of a given P-symbolism. This can be achieved, for instance, by setting up a procedure which goes through successive *stages*, and which, in the nth stage, lists all instructions P not previously listed which consist of no more than n formulas, each of which contains no more than n symbols. Each possible P occurs exactly once somewhere in the enumerated sequence. From now on we assume that we have selected a fixed P-symbolism and fixed L-P specifications in one of the known standard approaches, and that we have selected a fixed effective enumeration of the instructions P. The *index* of P shall be the position at which it occurs in the enumeration. Observe that we can effectively: (i) find a P given its index and (ii) find the index of any given P; in either case

[2] These are more commonly called the *partial recursive functions*.

we simply look sufficiently far in the enumeration. If x is the index of P, we shall call the corresponding B-L-P system M_x. Next, let us observe that there is also an effective procedure for enumerating all ordered pairs of integers. For instance, we can take as the zth ordered pair, the ordered pair $\langle x, y\rangle$ where x and y are the unique solution to the equation $z = \frac{1}{2}(x^2 + 2xy + y^2 + 3x + y)$. Let us choose such an enumeration and associate with it (in the obvious way) the notations $z = \tau(x, y)$, $x = \pi_1(z)$, $y = \pi_2(z)$. τ, π_1 and π_2 are effectively computable.[3] Having made the above choices, we see the following procedure is intuitively quasi-effective: given any z, find the instruction P whose index is $\pi_1(z)$; then carry out the computation made by $M_{\pi_1(z)}$ upon the input $\pi_2(z)$; if and when an output occurs, take it and make it the final output of the whole procedure. The extended version of Church's Thesis now suggests that there is a u such that M_u represents the above quasi-function, that is, it suggests the following theorem.

THEOREM I. *There exists a u such that for any x and y, M_u gives the same output to input $\tau(x, y)$ as M_x gives to input y.*

The theorem can be formally proved; and an appropriate detailed construction of the P for M_u, while tedious, presents no serious difficulties. For our purposes today, we shall consider an appeal to Church's Thesis as constituting proof. A single M_u, which can be used thus to simulate any other M, was called by Turing a *universal machine*. Various forms of Theoem I were discovered in the first period of the theory.

Another problem considered early in the development of the theory was the question of whether or not there is an effective procedure for identifying those instructions P which represent functions. Church's Thesis suggests the following precise reformulation: is there a recursive function f such that $f(x) = 1$ if M_x represents a function and $f(x) = 0$ if M_x does not represent a function? The answer to this was soon shown to be negative by the following argument. If such an f exists, then, using instructions for f together with instructions for the M_u of Theorem I, we can find instructions for the function [$g(x) = 0$, if $f(x) = 0$; $g(x) = y + 1$ where y is the output of M_x with input x, if $f(x) = 1$]. Let w be an index for these instructions. Then M_w represents a function, and M_w with input w yields an output y which, by definition of w, equals $y + 1$. The result follows from this contradiction. You will observe that this argument is similar to the diagonal proof by which Cantor shows the real numbers to be uncountable. In our

[3] For the interested reader, we remark here that the existence of a one-one effective mapping from $N \times N$ onto N means that *dimension* will not have a place in our theory analogous to that which it has in the theory of continuous functions on the real numbers. The study of recursive functions of several variables reduces directly to that of recursive functions of one variable.

theory, such arguments are often called "diagonal". Some of our later methods—for instance, the proof of Theorem V below—can be viewed as more subtle varieties of diagonal argument. In fact, our theory has been described as a "theory of diagonalization." As we shall see, the theory is considerably richer than such a description might at first lead us to believe.

The above negative answer is given in a somewhat stronger form in the following theorem.

THEOREM II. *The function* $[f(z) = 1$ *if* $M_{\pi_1(z)}$ *with input* $\pi_2(z)$ *yields an output, and* $f(z) = 0$ *otherwise*] *is not recursive.*

Both of the above forms follow as corollaries from Theorem V which we shall prove below. Further comments about them will be made at that point. Theorem II, in various forms, is sometimes referred to as yielding "the recursive unsolvability of the halting problem for Turing machines", (the word "halt" being associated with the appearance of an output).

Before leaving the first phase of our theory, let us develop a few more elementary ideas, from which we can obtain among other things, a proof for Theorem II. "A", "B", "C", $\cdots$ shall denote subsets of N. "$\bar{A}$" shall denote the set of numbers in N but not in A. "W_x" shall denote the set of all outputs of M_x. We make the following definitions. A set A is *recursive* if the function $[f(x) = 1$ for x in A; $f(x) = 0$ for x in $\bar{A}]$ is recursive. This corresponds intuitively to the existence of an effective procedure for deciding whether or not any number is in A. A set A is *recursively enumerable* if either A is empty or A is the range (set of all outputs) of some recursive function. This corresponds intuitively to the existence of an effective procedure for listing all members of A. The following two theorems can be proved without difficulty.

THEOREM III. *A is recursive if and only if both A and $\bar{A}$ are recursively enumerable.*

THEOREM IV. *A is recursively enumerable if and only if there exists an x such that $A = W_x$.*

I leave their proofs to you for your amusement, with the remark that justification by Church's Thesis will be permissible, that the second is harder than the first and that it will help if you treat the cases A finite and A infinite separately.

The question now arises: can we have a set which is recursively enumerable but not recursive? To settle this and to provide, at the same time, material for later illustrations, we introduce the following slightly more sophisticated concept. A set A is *productive* if there exists a recursive function f such that for all x, if W_x is contained in A, then $f(x)$ is in A but not in W_x; f is then called a *productive function* for A. Note that since every recursively enumerable set is equal to W_x for some x, a productive set cannot be recursively enumerable. We can now prove the following.

THEOREM V. *Let* $A = \{z \mid \mathrm{M}_{\pi_1(z)}$ *with input* $\pi_2(z)$ *yields an output*$\}$. *Then* A *is recursively enumerable, and* $\bar{A}$ *is productive.*

Proof. Consider the following procedure: given any z carry out the computation on z of M_u of Theorem I until an output by M_u occurs; when this happens, give output z. This is a quasi-effective procedure. The set of its outputs is the desired A. Hence, going to a P for this procedure we find an index w such that M_w has A as its set of outputs. Hence, by Theorem IV, A is recursively enumerable.

Assume now that there exists a recursive function g with the following property (*): for any z and any x, $\tau(x, x)$ appears as an output of M_z if and only if $\mathrm{M}_{g(z)}$ with input x yields an output. I claim that if such a g exists, then $\bar{A}$ is productive, with the function $f(z) = \tau(g(z), g(z))$ as productive function. f is clearly recursive. Assume (**): W_z contained in $\bar{A}$. Then $f(z)$ must be in $\bar{A}$ but not in W_z. For $f(z)$ in A implies by definition of A that $\mathrm{M}_{g(z)}$ with input $g(z)$ yields an output, which in turn implies by (*) that $f(z) = \tau(g(z), g(z))$ is an output of M_z, which in turn implies by (**) that $f(z)$ is in $\bar{A}$, a contradiction; and $f(z)$ in W_z implies by (*) that $\mathrm{M}_{g(z)}$ with input $g(z)$ yields an output, which implies by definition of A that $f(z) = \tau(g(z), g(z))$ is in A, which contradicts (**). It remains to show that a recursive function g exists with property (*). Given any fixed z, consider the following quasi-effective procedure. For any x, begin computing M_z with input 0; after a few steps, start the computation for M_z with input 1; then go back and work further on input 0; then start input 2; then work further on inputs 0 and 1; etc. In this way we can obtain and effectively list all the outputs of M_z. (This should give you a hint for proving Theorem IV.) If and when we find $\tau(x, x)$ as one of these outputs, give 0 as our final output. Instructions $\hat{\mathrm{P}}$ can be found for this procedure without difficulty. In fact, a rather natural version of $\hat{\mathrm{P}}$ will employ the M_u of Theorem I for computing M_z; hence this $\hat{\mathrm{P}}$ will itself depend effectively on the parameter z. That is to say, if we now vary z, we see that there is a recursive function g such that for any z, $g(z)$ is the index of the corresponding $\hat{\mathrm{P}}$. Thus g has property (*) and we are through. Several corollaries follow.

COROLLARY. *There exists a set which is recursively enumerable but not recursive.* (This follows by Theorem III and the fact that a productive set cannot be recursively enumerable.)

COROLLARY. *Theorem II follows* (since A is not recursive).

COROLLARY. *There exists a productive set.*

COROLLARY. *The set* $B = \{x \mid \mathrm{M}_x$ *represents a function*$\}$ *is not recursive.* (For if it were, we could find a procedure for settling any question of the form: does M_y with input z yield an output—in contradiction to Theorem II. We could do this by associating with any such question an M_x whose instructions tell us, for *any* input t, to give the same output as M_y gives for

input z. x depends effectively on y and z. Recursiveness of B would now supply an answer.)

The final corollary gives a partial explanation for the fact that the recursive functions, natural as they are, were such a long time in being discovered. The recursive functions are embedded in the recursive quasi-functions, and the general problem of distinguishing, from an instruction, whether or not we have a function, is effectively unsolvable.

One central result of the first phase of our theory remains to be described. An important tool in later research, it is essentially a fixed-point theorem, whose use in the solution of implicit function problems is similar to the use of fixed-point theorems in analysis. It is known as the *Recursion Theorem* and is due to Kleene.

THEOREM VI. *For any recursive function f, there exists an integer y such that $M_{f(y)}$ and M_y represent the same quasi-function.*

Proof. For a fixed z, consider the following quasi-effective procedure: given any x, compute M_z for input z; if and when any output, call it w, occurs, compute M_w for input x; give the result of this as final output. Since instructions for the above procedure, using M_u from Theorem I, can be made to depend effectively on parameter z, we have a recursive function g such that $M_{g(z)}$ gives that procedure. Now the function $f(g(z))$ is a recursive function. Let v be an index for it. Consider $y = g(v)$. Then M_y is $M_{g(v)}$ which by our construction represents the same quasi-function as M_w where w is the output of M_v with input v. But M_v with input v gives $f(g(v))$. Hence $w = f(g(v)) = f(y)$. Thus M_y represents the same quasi-function as $M_{f(y)}$, and the proof is done. Note, incidentally, that the proof gives us an effective way of going from instructions for f to the value y. Hence we have a corollary.

COROLLARY. *There exists a recursive function h such that for any x, if x is an index of instructions for a recursive function f, then $M_{f(h(x))}$ and $M_{h(x)}$ represent the same quasi-function.*

This concludes our survey of the first part of the theory. A final comment is in order. If the P-symbolism and L-P specifications are given in detail, all of the above informal proofs can be replaced by detailed constructions within the P-symbolism. As we have remarked before, this means that, independent of Church's Thesis, the theorems can be given precise content. In the first period of the theory, much work went into carrying out these constructions in detail. In the course of this work, the natural and invariant quality of the objects concerned emerged more and more clearly. The naturalness of these objects is now so well confirmed that the above informal proofs would certainly be accepted today, at least in informal communication between researchers. In fact, if you have a firm intuitive grasp of our concept from my presentation so far, you are equipped to do re-

search yourselves, without ever getting into the details of a particular P-symbolism. (It is for this reason that I used the original anthropomorphic diagram.) You are in a position to do research much as the high school algebra student is in a position to do research in number theory. This natural, almost primitive quality of our subject matter—it has been called one of the few *absolute* concepts to emerge from modern work on foundations of mathematics—explains much of the recent enthusiasm of investigators about the subject. It also explains the somewhat extravagant remark with which Post concluded his excellent address to the American Mathematical Society in 1944 [1]. "Indeed, if general recursive function is the formal equivalent of effective calculability, its formulation may play a role in the history of combinatory mathematics second only to that of the formulation of the concept of natural number."

Part II

To pick 1943 as an epoch in our theory is somewhat arbitrary. For want of a better dividing point, I choose it because of papers of Kleene, Mostowski and Post [1, 2, 3], completed at about that time, which helped lead investigators into new and exciting areas, and which have served as a continuing stimulus to their efforts. I now go on to survey the further developments of the theory, most but not all occurring after 1943. For simplicity, I group them into six categories, though this grouping leads to some omission and obscures certain important interconnections. We can only give a few, necessarily simple, illustrations in each category. The names of researchers mentioned represent a rather arbitrary selection. Many equally deserving workers will be unnamed. The categories are: logic and foundations; recursive unsolvability; recursive invariance; recursive structures; recursive analysis; and intrinsic definitions of recursiveness.

In the first area—applications to *logic and foundations*—the date 1943 is of little significance. Recursive function theory has had a revolutionary impact on this area from the beginning. It has been of immeasurable value: first, in supplying a natural frame of reference in which to restate and gain new insights into work already done; and second, in suggesting fruitful ways to approach old problems and to formulate new ones. We give several brief illustrations.

Given any logical system, the possible formulas of its symbolism can be effectively enumerated in a manner analogous to that suggested above for enumerating all instructions P. Just as with the P's, this gives an effective way for going from formula to number and from number to formula. Once a fixed such enumeration is chosen, the number associated with a formula is usually called its *Gödel number*. In what follows, we shall sometimes refer to a set of formulas as if it were a set of integers. This will be an abbreviated

way of referring to the set of corresponding Gödel numbers. In all such cases, it will be possible to show that the result in question does not depend upon the particular effective enumeration used to get the Gödel numbers.

As first illustration, we take the Gödel incompleteness theorem, one of the most celebrated results in modern logic. Our theory gives several suggestive and concise ways of restating parts of this theorem. We make the preliminary observation that in any of the usual systems of logic, the set of provable formulas is recursively enumerable (by a procedure of listing possible proofs analogous to our suggested procedure for listing P's); in fact, recursive enumerability of the provable formulas is now generally accepted as a necessary condition for a formalism to be called a "logical system". By *elementary arithmetic*, we shall mean the set of all formulas expressible using numerals, number variables, "+", "·", "=", parentheses, and quantifiers over the number variables—in the usual way. We give these formulas the usual interpretation over the integers. Our first partial restatement of the Gödel Theorem now is: *the true formulas of elementary arithmetic form a productive set.* The use of the concept of truth as distinct from provability is justifiable in the sense that our whole discussion can be conceived of as carried out in some general set theory in which truth can be defined by the well known foundational methods of Tarski. The concept of truth is avoided, however, in the further restatement: *given any methods of proof as strong as the usual ones (e.g. Peano's axioms) then the set of nonprovable formulas of elementary arithmetic either is empty or is productive.* The second formulation directly implies the first; I leave this to you and omit details. Note how much information the first formulation gives us: there is an effective procedure such that given any effective enumeration of true formulas, we can explicitly find from the instructions for that enumeration itself, a true formula not enumerated.

The logical system known as *lower predicate calculus* furnishes a second example. *Church's Theorem* and the *Gödel Completeness Theorem* are two of the most basic results about this system. Church's Theorem states that the set of universally true formulas is not recursive, while the Gödel Completeness Theorem can be partially restated as the assertion that the same set of formulas is recursively enumerable.

This concludes our direct treatment of logic and foundations, though, as we shall see, the other categories have substantial overlap with it.

The second topic, *recursive unsolvability*, has also been of interest from the beginning. It early became clear that our theory gives a natural and precise sense in which certain kinds of "problem" can be described as "unsolvable by effective methods"; namely, we call a problem *recursively unsolvable* if it is equivalent to the problem of identifying members of a nonrecursive set. Thus Church's Theorem, above, asserts that the "prob-

lem" of identifying universally true formulas of lower predicate calculus is recursively unsolvable. The corollaries to Theorem V in Part I above give similar results about the problem of identifying instructions that represent functions and the problem of identifying ordered pairs $\langle x, y \rangle$ such that M_x with input y yields an output.

A further result of this kind is contained in recent work of Novikov and of Boone who show that the word problem for groups is recursively unsolvable. They exhibit a particular finitely generated group with finitely many relations such that, if all possible words are taken as the basic Gödel-numbered formulas, the set of words reducible to the identity is a nonrecursive set.

A challenging open question in this area is Hilbert's 10th problem: the problem of deciding the existence of diophantine solutions to polynomial equations. The set of equations with solutions is clearly recursively enumerable. Is it recursive? Several investigators have conjectured not, have carried their attempted proofs quite far, but have fallen short of a final answer.

A number of other nonrecursive sets of specific mathematical or logical interest are known. Tarski and Robinson have developed useful methods for demonstrating unsolvability in a broad class of formal theories. Substantial work in discovering positive solvability results for certain logical systems has also been done. Thus, while the lower predicate calculus formulation of group theory is unsolvable, that for abelian groups is solvable. While that for the integers is unsolvable, that for the real numbers is solvable. Despite Church's Theorem, large subclasses of the universally true formulas of lower predicate calculus have been the subject of solvability investigations.

One of the most widely studied and satisfying areas of recent work is the third area—the area of *recursive invariance*. Before going into it, we make an observation about our previous discussion. The listener will have noted that some of our theorems and illustrations have depended upon a number of apparently arbitrary choices—for instance Theorem V involves the choices of both an enumeration of P's and an enumeration of ordered pairs. Consider the collection $\mathcal{G}$ of all recursive functions which map N one-one onto itself. I leave it to you to verify that $\mathcal{G}$ forms a group under ordinary composition of functions; it is called the group of *recursive permutations*. Our previous results now acquire a more invariant significance in the sense that they can be shown to hold under any recursive permutations of the original arbitrary enumerations, as you can easily check.

The area of *recursive invariance* is chiefly concerned with the exploration of recursive function theory as a part of mathematics in its own right. As Dekker has suggested, the group $\mathcal{G}$ is useful in describing its subject matter;

much of the theory can be described as the study of those properties of sets of integers which are invariant under members of $\mathfrak{G}$. The properties of recursiveness, recursive enumerability and productiveness are all invariant in this way, as you may check. We call such properties *recursively invariant*. Two sets are called *isomorphic* if one is an image of the other under a member of $\mathfrak{G}$. The isomorphism classes are hence the basic objects of our theory. An interesting recent result of Myhill is the theorem that the properties A recursively enumerable and $\bar{A}$ productive are a complete family of invariants; that is to say, any two sets possessing both properties are isomorphic. This theorem takes on special significance in view of the fact that virtually all known unsolvability results yield sets possessing these properties. A natural question then to ask is: are any two nonrecursive, recursively enumerable sets isomorphic? Post, in his paper of 1944, shows that this is not true by constructing a recursively enumerable set A such that $\bar{A}$ contains no infinite recursively enumerable set (and hence is not productive, as you can show). The interesting problem remains, what sort of structural similarity *can* be found among the nonrecursive recursively enumerable sets? Post proposed a weaker notion of similarity for use in this study, the notion of *Turing equivalence*. Two sets are Turing equivalent if each is *reducible* to the other. A is *reducible* to B if the function [$f(x) = 1$ for x in A; $f(x) = 0$ for x in $\bar{A}$] can be computed by an M provided that (i) L has available two lists, possibly infinite, of the members and nonmembers of B respectively; and (ii) the P symbolism and L-P specifications are appropriately modified in such a way that L can be instructed to consult his B lists in the course of his computation. Turing equivalence gives us larger equivalence classes. Until very recently, all known nonrecursive recursively enumerable sets A could be shown to possess the property: *for any recursively enumerable B, B is reducible to A*. Hence all were in the same equivalence class. The problem of whether or not this was necessarily so acquired the name *Post's Problem*. The foundational significance of the problem can be seen from our earlier comment relating provability in a logical system to recursive enumerability. The problem remained unsolved until last year when Friedberg and Muchnik each gave an ingenious construction showing that the recursively enumerable sets lie in an infinity of distinct classes.

The Turing equivalence classes are sometimes called *degrees of unsolvability*; they are partially ordered under the reducibility relation. The papers of Kleene and Mostowski helped initiate their study, and demonstrated certain natural relationships between the degree of a set and the logical complexity necessary to a formal definition of that set. Much interesting work at the present time concerns this last topic. Essentially the problem is one of relating logical complexity, as measured by type and location of

quantifiers, to "height" in the reducibility ordering as measured in some appropriate weakly constructive way. Both Kleene and Mostowski have continued to contribute important ideas to this work.

As a final illustration, let us turn for variety to a proof. We illustrate the use of Theorem VI, the fixed point result, in the area of recursive invariance. A set A is called *completely productive*, if there exists a recursive function f such that for any z, $f(z)$ lies either in the intersection of A and $\overline{W}_z$ or in the intersection of W_z and $\bar{A}$. As Dekker has commented, a completely productive set is a set which fails to be recursively enumerable, and does so in the strongly constructive sense that the "counter-example" $f(z)$ can be effectively exhibited, given any instructions P with index z. It is immediate that a completely productive set is productive. Is the converse true? Myhill has announced a positive answer. We give a proof in the following theorem.

THEOREM VII. *If A is productive, then A is completely productive.*

Proof. Let f be a productive function for A. Let z be fixed. Take any y. Consider the quasi-effective procedure: for any input x, search for $f(y)$ in W_z ; when it appears, give output $f(y)$. The index for this procedure will depend effectively on y. Applying Theorem VI, we see that for some y, this procedure itself has index y. Turning to the corollary to Theorem VI, we see that this "fixed point" y can be found effectively from z; hence there is a recursive function h such that this $y = h(z)$; that is, $\mathrm{M}_{h(z)}$ has as its outputs either the empty set or the single number $f(h(z))$, according as $f(h(z))$ belongs to $\overline{W}_z$ or W_z . You may now verify that, with respect to the function $[\hat{f}(z) = f(h(z))]$, A is completely productive.

The fourth area, *recursive structures*, is concerned with the study of interrelations between recursive function theory and classical mathematics. It includes the study of recursive representations of the classical concepts, of recursive analogues of the classical concepts and of direct imposition of recursive structure upon the classical concepts. This work has been pursued with increasing intensity in recent years, and has become a fascinating part of the theory.

The classical set theory of Cantor is a good illustration, and the results obtained have obvious foundational significance. Essentially, we seek to find a model of the Cantor theory within the class of all sets of integers by using the Cantorian concepts subject to the restriction that all correspondences and mappings be recursive. The ordinal numbers were studied first. Investigators have included Church, Kleene and, more recently, Markwald and Spector. If, for instance, we consider all well-orderings of integers which, as relations (sets of ordered pairs), form recursive sets; and if we then consider the ordinals represented by these orderings, we obtain a proper segment of the classical first and second number classes which is, in many

respects, a successful "effectivized" model of the latter classes. This segment turns out to be, for instance, the largest segment for which notations exist and which is closed under limits of recursive increasing sequences of order type ω. This segment also proves to be a useful tool in measuring "height" in the reducibility ordering of degrees of unsolvability.

A similar attack on the cardinal numbers has been begun by Dekker and Myhill. Using a deceptively straightforward modification of the Cantor definition, they have uncovered a rich theory. In their model, they have found, for instance, an intermediate class of *semi-finite* infinite cardinals whose arithmetic is much closer to that of the integers than is the arithmetic of the classical cardinals.

The approach to classical set theory is still in its early stages, and much remains to be done. For example, a satisfactory theory linking cardinal and ordinal numbers does not yet exist. On a somewhat broader front, studies are being made, by Wang, Kreisel and others, of general axiomatic set theories that are subject to certain kinds of recursive constraints. The prospect of further work in this whole region is an exciting one.

Another illustration of recursive structures is found in the work of Addison in exploring the relation of recursive invariance to parts of the point set theory developed by the Polish school of topology. A number of close analogues are found to exist between the two theories; they give promising suggestions for further results in both fields.

A further illustration is the work of Rabin on recursive structure in algebraic systems. A recursive structure is imposed on algebraic objects in a manner similar to that in which topological structure is often imposed. Various new and suggestive ideas appear. A chief value of the work is that it gives a natural way of handling notions of effectiveness that already exist, somewhat awkwardly, in the algebraic theory. For example, a short, elegant formulation and proof can be given for Van der Waerden's theorem on the existence of a factorization algorithm for separable extensions over a field possessing such an algorithm.

The fifth area, *recursive analysis*, could quite properly be included under recursive structures. It has, however, been the object of independent exploration for some time, and I accord it separate, though brief, mention. It begins with the attempt to impose recursive structure on the usual theory of real numbers. A class of real numbers called the *recursive reals* is defined by requiring that each have a decimal expansion representable by a recursive function. This class proves to be a natural one in the sense that it is invariant with respect to reasonable alternative definitions; and it can be shown to form an algebraically closed field. A function r of a real variable is called *effective*, if there exists a recursive function f such that,

when x is a Gödel number for an initial segment of the decimal expression for any real number t, then $f(x)$ is a Gödel number for an initial segment of the decimal expression for $r(t)$, and furthermore such that the output segments become arbitrarily long if longer and longer input segments are used. I leave it to you to show, using Theorem V, that every effective function is continuous. If you have got sufficiently into the spirit of the corollaries to Theorem V, you will not find the proof difficult. If we develop a theory that limits itself to the recursive reals as the only reals eligible for any purpose, discontinuous functions can be made to reappear.

The sixth and last category, *intrinsic definitions of recursiveness*, is an interesting and challenging one. Few results exist in this area, however; and I must admit that including it as a separate topic is a matter of personal taste. A whole complex of problems are concerned. First of all, the listener will have noted that while various of our definitions prove to be *invariant* (with respect to the group $\mathfrak{G}$), they are not *intrinsic*—that is, they depend upon *some* choice. The original definition of recursive function is a good example. It depends upon the choice of a P-symbolism and L-P specifications. Can a more intrinsic formulation be found? Some small progress toward this is made in [4]. Secondly, the more general question arises: can we somehow abstract our theory away from integers, functions, etc.? Can we find a development which will be to the present recursive function theory as abstract algebra is to the theory of numbers? (This problem, incidentally, is the rock upon which attempts completely to algebraize modern logic have foundered up to the present time; though the partial systematizations of Tarski, Halmos and others have been successful as far as they go. In attempts to find general algebraic analogues to the incompleteness theorems, there has been a residue of recursive function theory that has not been eliminable.) To thus incorporate recursive function theory into general abstract algebra, at least in part, would be a considerable achievement and would doubtless give much new insight into the theory as we know it.

This concludes Part II of my talk. In closing let me make several comments about the literature. If you are interested in looking further into the subject, the basic theory of Part I is perhaps most directly accessible in the forthcoming book of Davis [5], which uses Turing's approach. This reference is also an excellent introduction to recent unsolvability results and contains material from other areas as well. Kleene's *Introduction to Metamathematics* [6] gives a comprehensive treatment of the basic theory and contains much valuable material in the areas of logic, foundations, and recursive invariance. Many results, however, exist only in the journals at the present time. Both the Davis and Kleene volumes have good bibliographies.

REFERENCES

1. E. L. Post, *Recursively enumerable sets of positive integers and their decision problems*, Bull. Amer. Math. Soc., 50 (1944), pp. 284–316.
2. S. C. Kleene, *Recursive predicates and quantifiers*, Trans. Amer. Math. Soc., 53 (1943), pp, 41–73.
3. A. Mostowski, *On definable sets of positive integers*, Fund. Math., 34 (1947), pp. 81–112.
4. H. Rogers, Jr. *Gödel numberings of partial recursive functions*, J. Symb. Logic, 23 (1958).
5. M. Davis, *Computability and Unsolvability*, McGraw-Hill, New York, 1958.
6. S. C. Kleene, *Introduction to Metamathematics*, Van Nostrand, New York, 1952.

Massachusetts Institute of Technology