On computable numbers, with an application to the Entscheidungsproblem

(Proc. Lond. Math. Soc., series 2 vol. **42** (1937), pp. 230–265)

– A correction

(ibid. vol. **43** (1937), p. 544–546)

PREFACE

1. Preamble

This is the classic paper which first established Turing's reputation and by which he will longest be remembered. The argument falls into three parts.

A. The notion of a Turing machine is introduced and it is argued that any computation, which can be performed by a human can be imitated by such a machine.
B. It is shown that there is a universal machine which, when provided with a standard description of any Turing machine will imitate the action of that machine.
C. A diagonal argument is used to show that there are questions about the actions of Turing machines which cannot be answered by any machine. By formalizing the action of Turing machines in the lower predicate calculus it is shown that the Entscheidungsproblem is mechanically undecidable.

1.1. *History*

Turing always enjoyed calculating. At school he had devised a method for computing π and had used it to calculate the first 36 decimal places [*Hodges* p. 35]. While walking, bicycling or washing up he would perform mental calculations about mathematical or physical phenomena. Whether calculating mentally or with pencil and paper, Turing was methodical only by fits and starts, and often made mistakes. [When I came to know him later the phrase 'What's a factor of two between friends?' had become a catchword.] But he understood very well what it meant to be totally methodical. Indeed an acceptance – sometimes ready, sometimes reluctant – of the dichotomy between the clearly perceived ideal and the confused actuality was fundamental in Turing's thought.

In the Spring of 1935 Turing attended lectures by M.H.A. Newman on mathematical logic [*Hodges* p. 91–93] in which the Entscheidungsproblem (and

Gödel's incompleteness theorem) were discussed. To prove, what by then was commonly though not universally believed, that there could be no effective or mechanical method for deciding which formulae of the predicate calculus are provable, it was necessary to limit the notion of 'effective method' by giving a precise definition of it. The need for such a definition was what immediately stimulated Turing to analyse the process of computation. But it is plain from the paper that he was just as interested in the positive aspects of his analysis as in the negative results, which it enabled him to prove. In particular, section 10 is entirely concerned with what numbers and functions are computable. Ostensibly the purpose of this section is to show that the defined notion of computable has some of the properties which one would expect any intuitive notion to have; but Turing is interested in their properties for their own sake.

I remember Turing telling me that the 'main idea for the paper' came to him when he was lying in the grass in Granchester meadows in the summer of 1935. I assume that he had by then already conceived of some form of Turing machine, and that what he meant by 'the main idea' was the realisation that there could be a universal machine and that this could permit a diagonal argument. Sometime after this he described the universal machine to his friend David Champernowne. He did not discuss his work with Newman, but gave him a completed typewritten draft of the paper in April 1936 [*Hodges* p. 109]. A little later Newman received from Alonzo Church an off print of his paper [1936] and a pre-print of [1936a], the results of which had been presented to the American Mathematical Society in April 1935. Turing inserted a reference to this in the introduction to his paper, and sent it to the London Mathematical Society where it was received on 8th May. On 31st May Newman wrote to Church:

> An off print which you kindly sent me recently of your paper in which you define "calculable numbers" and shew that the Entscheidungsproblem for Hilbert logic is insoluble, had a rather painful interest for a young man, A.M. Turing, here, who was just about to send in for publication a paper in which he had used a definition of "Computable Numbers" for the same purpose.

Church had certainly obtained the result before Turing; but Turing had written his draft without any knowledge of Church's work. It should be remarked that there is no evidence that Turing had read any of the scanty and sporadic literature concerned with the general theory of mechanical computation. In particular, one can be sure that if Turing had read either account by Babbage of the Analytic Engine (Chapter VIII of 'Pages from the life of a philosopher' [1864]) or the account of Menabrae [1842] translated by the Countess of Lovelace [1843] he would have mentioned Babbage's ideas. He might well have read the article on Calculating Machines in the 11th edition of the Encyclopaedia Britannica. In later years he often consulted the copy of the Encyclopaedia which he inherited from his father – but that article contains only a short and rather dismissive reference to the Analytic Engine: 'a much more powerful machine... intended to perform any series of possible arithmetical operations'. This would hardly have suggested to Turing that Babbage had in fact conceived a universal machine.

2. Discussion

For an overall account and estimation of the paper, the reader is referred to Newman's obituary to Turing reproduced in this volume. My paper 'The confluence of Ideas in 1936' [Gandy 1988] contains an account of the background history of ideas; and a discussion of the contributions made by Hilbert and his school, by Church and his students and by Post. It also includes a discussion of Turing's work and its significance. In what follows I shall occasionally draw on that article without indicating the difference between paraphrase and direct quotation.

2.1. *Turing's analysis of computation*

Turing considers 'computable (real) numbers'; in fact, what he is considering is total computable functions of a positive integral argument with values 0 and 1. He starts off his detailed analysis (pp. 249–258) by saying:

> The real question at issue is "What are the possible processes which can be carried out in computing a real number?"

This is significantly different from the question 'What is a computable function?' which other authors asked. Turing, so to speak, pointed himself in the true direction.

2.1.1. He then considers the actions of an abstract human being who is making a calculation; he pictures him as working on squared paper as in "a child's arithmetic book". He argues – too briefly – that nothing will be lost by supposing that the calculation is carried out on a potentially infinite tape divided into squares in each of which a single symbol (or none) may be written.

2.1.2. By considering the limitations of our sensory and mental apparatus Turing arrives at the following restrictions on the actions of a computor.[6]

> (1) There is a fixed upper bound to the number of distinct symbols which can be written on a square.
> (2) There is a fixed upper bound on the number of contiguous cells whose contents the computor can take in – 'at a glance' as one might say – when he is deciding what to do next.

[6] I use 'computor' for a human being, 'computer' for a machine.

(Turing shows by an example that for a normal human being – the reader – this bound, for a linear arrangement, is less than 15. On pp. 250–251 Turing considers the possibility that besides looking at the currently observed squares the computor might also look at some 'immediately recognised' specially marked squares. But there must be a fixed upper bound on the number of immediately recognised squares, and so – without detailed argument – he claims that they can be, in effect, adjoined to the observed squares.)

(3) At each step the computor may alter the contents of only one square, and there is a fixed upper bound to the distance the computor can move to reach this square from the observed squares; so we may suppose that it is one of them.

(4) There is a fixed upper bound to the distance between the squares observed at one stage and those observed at the next stage.

(5) There is a fixed upper bound to the number of 'states of mind' of the computor: his 'state of mind', together with the contents of the observed squares, uniquely determine the action he takes (printing and moving his field of observation), and his next 'state of mind'. In place of 'state of mind' Turing admits that the computor might leave an instruction on how to continue (p. 253).

2.1.3. Thus the computor must follow a fixed, finite, totally explicit set of instructions satisfying the above restrictions. It is then easy to see that his action can be simulated by a Turing machine – which at each step observes a single square, can alter only the contents of that square, and moves by at most one square.

2.1.4. It is worth emphasising that Turing's analysis is quite explicitly concerned with calculations performed by a human being; there is no reference to machines other than those which he introduces to imitate the actions of a human computor. In subsequent discussions of Turing's work this fact has sometimes been obscured by a play on the uses of the word 'mechanical', which has often since the seventeenth century been applied in a loose figurative sense to human actions. [The earliest example (1607) given in the Oxford English Dictionary refers to farriers who mistreat sick horses by rule of thumb]. Of course, it is not surprising that Turing does not mention machines. Numerical calculation in 1936 was carried out by human beings; they used mechanical aids for performing standard arithmetical operations, but these aids were not programmable. According to Randall ([1982], p. 160) the first general purpose programme-controlled machine to be built and used was Zuse's machine completed in 1941; even this (and other machines of the same generation) did not fully allow for conditional branching.

Indeed, Turing's analysis does not directly apply to (discretely acting) machines, since it takes no account of the possibilities of parallel action. If one considers (as in Newtonian theory) the possibility of instantaneous action at a distance, then the alteration of the record need not be local nor locally determined. However, if (as in the theory of relativity) one supposes that there is an upper bound to the velocity of propagation of physical influences, then one can establish Turing's thesis for machines as well as for human beings (see Gandy [1981]).

2.1.5. What makes Turing's analysis so breathtaking is its combination of generality, directness and simplicity. These are characteristic of his way of thinking; but the particular manner in which they come to the fore in this paper is a result of his having asked himself the question 'What is a computable real number?' rather than, say, 'What functions are computable?'. The latter question leads most naturally to a consideration of the various different ways in which calculable functions may be specified; thus Hilbert and his school investigated various kinds of recursive definition (see Hilbert [1926] and Ackermann [1928]). Had Turing started with this line of thought, his machines might have been described in terms of a high-level language (such as LISP, say) rather than the extremely simple machine – language which he actually uses. But then the argument that all conceivable methods of calculation had been covered would have been far less direct and less cogent.

2.1.6. Turing concentrated on implementation rather than specification, and found that this made it possible to set an exact limit on what is calculable. He did, however, use the notion of specification as a supporting argument – for predicates in §9.II, and for functions in the first paragraph of §10. A predicate G is specified by giving a formula $\mathbf{A}$ (of first-order predicate calculus using only relational symbols) which implicitly defines it together with the requirement that G be *formally reckonable* in the predicate calculus; i.e. that the appropriate formula $G(x)$ or $\neg G(x)$ can be inferred from $\mathbf{A}$ and the formula which expresses that x is the n-th natural number. One can say that the specification $\mathbf{A}$ of G must be implemented by formal proofs. But this supporting argument lacks the generality of Turing's direct proof of his thesis. Firstly, recursions involving higher type objects may be used in specifying calculable functions; see, for example, p. 389 of Hilbert [1926]. Secondly, one might wish to interpret 'formal reckonability' as allowing proofs in some formal system more powerful than the predicate calculus; this possibility is considered by Church ([1936], p. 357) at the corresponding point in *his* argument.

2.2. *Philosophical significance of Turing's analysis*

2.2.1. *Formal systems.* Turing's work makes it possible to give a satisfactory and definitive characterisation of a formal system (or theory). Namely, a formal system is one whose expressions are built up from a finite list of primitive symbols and for which there is a Turing machine which will test all its theorems (or, more generally, all its inferences). Turing does not refer to that possibility in this paper, but the characterisation is of fundamental importance for the philosophy of mathematics. It allows Gödel's proof of incompleteness to be applied to any formal system, which contains a certain amount of elementary number theory, and was much emphasised at times by Gödel in 1963–1965 (*Volume 2 of his Collected Works* [1990]). Hence, it sets limits to what can be rigorously proved in any formalisation of mathematics. In his paper on ordinal logics, which appears later in this volume, Turing gives a model for the different roles, which are played in mathematics by intuition and by formal proofs. This notion allows one to generalise the notions of formal specification and formal reckonability described in **2.1.6** above; but one cannot, of course, use the fact that they give an equivalent definition of computability as a supporting argument for Turing's thesis – this would result in a vicious circle.

Note: It should be mentioned that in the early 1920's Post had developed a quite different (but eventually seen to be equivalent) characterisation of formal system. This was not published until Post [1943]; for an historical account see Post [1965].

2.2.2. *Wittgenstein's paradox.* Turing's analysis can be applied quite generally to characterise the notion of a *rule* – of calculation, of inference, of procedure, of construction - and so on. In mathematics, such rules are usually designed so that they may be applied in a potentially infinite number of distinct situations. This gives rise to a puzzle or paradox with which Wittgenstein was much concerned. Namely, suppose one has witnessed someone (perhaps oneself) apply a given rule in a finite number of cases; how can one be sure that one has applied the given rule and not some other rule which agrees with it so far? Hence, how can one ever prove that a particular rule has been correctly applied? The paradox is not immediately resolved by requiring that the particular rule be clearly stated; for the statement cannot list what is to be done in all the infinite number of situations in which the rule may be applied.

Turing's analysis does not solve this paradox, but it does make clearer where the heart of the problem lies. I will call a finite list of actions which are to be taken in suitable circumstances a *recipe*; such, for example, is the set of instructions which one might give out for travelling from one place to another. A recipe may well include alternatives. The table of instructions (or programme) for a Turing machine constitutes a recipe. What Turing showed was that applying a rule in a given situation can always be reduced to the iterated application of a recipe. Thus the essential puzzle is: 'How can one prove that someone has followed a recipe correctly?'; or 'How can one prove that a situation (or even an action) is the same on one occasion as it was on another?' Wittgenstein certainly recognised this form of the puzzle; he discusses, for example, how one can *know* that a colour word has been correctly used. His solution draws attention to the fact that correctness is to be judged by the ability to communicate with others, and so cannot be applied to the behaviour of a totally isolated individual. But he sometimes talked as if the puzzle about rules (and proofs) was different from, was not merely a rhetorical embellishment of, the puzzle about recipes. In particular, so it seems to me, he did so in the lectures on the foundations of mathematics (recorded in Wittgenstein [1976]) which he gave in 1939 and at which Turing was a vocal participant. I find it surprising that Turing seems never to have made the point which I have just discussed. A full and clear account of Wittgenstein's paradox and his solution of it will be found in Kripke [1982].

2.3. *Turing machines and electronic computers*

2.3.1. *The influential ideas.* The title of Turing's paper and the fact that its first section is headed 'computing machines' encouraged people concerned with the design of computers to read it, or at least to look at it; but, the ideas it contains would have been equally important if Turing, like Post, had avoided the use of the word 'machine'. I think that these ideas, in order of importance, are as follows:

(i) The elementary steps are extremely simple, and have specifications of a fixed length.

(ii) The universal machine is a stored-program machine; that is, unlike Babbage's all-purpose machine, the mechanisms used in reading a program are of the same kind as those used in executing it.

(iii) Conditional instructions are no different from unconditional ones.

(iv) The operation is easily adapted to binary storage and working.

Notes: (1) Turing presumably realised when he wrote the paper that, like Post, he could have used 'mark' and 'blank' as the only symbols; but that had he done so his table for the universal machine would have been totally unreadable.

(2) Kleene used binary Turing machines in his book [1952].

It is (i) and (iv) that made it possible for McCulloch and Pitts to show that the control mechanisms of a Turing machine can be simulated by a finite network of 'neurons' (gates with delays). The λ-calculus and the equational calculus both use 'substitute a given term for a given variable in a given expression' as an elementary step; this cannot have a total specification of fixed length. On the other hand, the λ-calculus is a stored program device, since there is no difference between a program (a λ-term) and the successive stages in its computation. Although the earlier designs for computers (in particular, the EDVAC) only allow rather restricted use of conditional instructions, the use of gates with two or more inputs does in fact reflect the conditional nature of the elementary steps of a Turing machine.

2.3.2. *Their effect on U.S. developments.* There is some controversy about the exact extent of the influence of Turing's ideas on the design of electronic computers in the U.S.A. I record a few historical facts:

(1) By the late 1930's von Neumann had become familiar with Turing's ideas and was enthusiastic about them (Randell [1972], p. 10; Hodges [1983], p. 145; Davis [1987]). There is no evidence, however, that there was ever a meeting at which they exchanged ideas about the possibilities of designing 'universal' electronic computers. They did meet early in 1947, but by that time they had written their respective reports. In a perceptive footnote ([1983], fn 5.26 on pp. 555–556) Hodges argues that such a meeting would not, in any case, have been likely to have been of great importance; each would develop his own ideas.

(2) In 1945 von Neumann wrote his 'First draft of a report on the EDVAC'. In this he makes considerable use of the idealised neuronal networks of McCulloch and Pitts [1943]. He does not explicitly refer to Turing [1936–7] in that paper (although in his Hixon lecture in 1948 [1951] he gives Turing his due). As with Turing's universal machine, the program is stored in a special part of the memory. This report circulated widely and was influential. Turing read it in the summer of 1945.

(3) Most of the essential ideas for the design of electronic computers – binary working, use of logical circuits and stored programs – were developed independently by various people, from 1936 onwards; see Randell [1982], chapters VII and VIII, and Burks [1980]. The most important and influential place for the construction of electronic computers in the 1940's was the Moore School of Engineering; the first working large-scale (18,000 valves) electronic computer was the ENIAC. The original proposal (in April 1943) was by Mauchly and Eckert. The machine was finished at the end of 1945. In the meantime, plans were being made for 'EDVAC-type' machines; von Neumann became a consultant there in September 1944. Much of the work on ENIAC and EDVAC was classified. It has become, it seems, impossible to discern a linear flow of ideas; probably there was no such thing. Much acrimony (see Eckert [1980] and Mauchly [1980]) and a protracted legal battle developed from the question of who told what to whom.

(4) According to Randell [1980], Turing and Newman and a group of mathematicians and engineers at Bletchley Park discussed in 1942–43, out of office hours, Turing's universal machine, Babbage's plans for the Analytic Engine, and the possibilities of artificial intelligence. Turing played some part in the design of the first machines built under Newman's direction. The final machines ('Colossi') can claim to be the first medium-sized (2,000–3,000 valves) fairly flexible, pro-

grammable electronic computers. (The Mark II Colossus came into service in June 1944.)

(5) Turing worked at the National Physical Laboratory from 1945 to 1948 designing a computer. His report on the ACE [1945] was submitted in March 1946. Turing was influenced by von Neumann [1945], but he describes, in fair detail, the design of a quite specific machine, and, ahead of the times, proposed that many of the operations of the machine should have been effected by writing subroutines, rather than by building special single-purpose units. In his lecture [1947] to the London Mathematical Society he traces the connections between the design of the ACE and his 1936–7 paper.

The ideas of the ACE report are discussed at length by Hodges [1983], pp. 317–333.

(6) At the same time, in Manchester, Newman was responsible for, and contributed to, the design of what became the Manchester Mark 1 machine (Newman [1948]). Turing joined him in the autumn of 1948. His chief interest was in using the machine (and helping others to use it). Although he wrote the 'Programmer's Handbook' he did not contribute to the design of the machine or of programming languages for it. Hodges [1983, p. 401] lists some of the things which Turing could have worked on but didn't.

2.3.3. *Summary.* Although it may be difficult to trace the precise influence of this paper on the design and development of high-speed digital computers, its fundamental importance for the theory of computation is clear. Turing machines (and modifications of them) still provide the standard setting for the definition of the complexity of computation in terms of bounds on time and space; together with the neural nets of McCulloch and Pitts they provided the foundations of the theory of automata; together with the generated sets of Post [1943] they provided the foundation for the theory of formal grammars.

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

have valuable applications. In particular, it is shown (§ 11) that the Hilbertian Entscheidungsproblem can have no solution.

In a recent paper Alonzo Church† has introduced an idea of "effective calculability", which is equivalent to my "computability", but is very differently defined. Church also reaches similar conclusions about the Entscheidungsproblem‡. The proof of equivalence between "computability" and "effective calculability" is outlined in an appendix to the present paper.

1. *Computing machines.*

We have said that the computable numbers are those whose decimals are calculable by finite means. This requires rather more explicit definition. No real attempt will be made to justify the definitions given until we reach § 9. For the present I shall only say that the justification lies in the fact that the human memory is necessarily limited.

We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions $q_1, q_2, \ldots, q_R$ which will be called "*m*-configurations". The machine is supplied with a "tape" (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the r-th, bearing the symbol $\mathfrak{S}(r)$ which is "in the machine". We may call this square the "scanned square". The symbol on the scanned square may be called the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its *m*-configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined by the *m*-configuration q_n and the scanned symbol $\mathfrak{S}(r)$. This pair q_n, $\mathfrak{S}(r)$ will be called the "configuration": thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (*i.e.* bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the *m*-configuration may be changed. Some of the symbols written down

† Alonzo Church, "An unsolvable problem of elementary number theory", *American J. of Math.*, 58 (1936), 345–363.

‡ Alonzo Church, "A note on the Entscheidungsproblem", *J. of Symbolic Logic*, 1 (1936), 40–41.

will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to "assist the memory". It will only be these rough notes which will be liable to erasure.

It is my contention that these operations include all those which are used in the computation of a number. The defence of this contention will be easier when the theory of the machines is familiar to the reader. In the next section I therefore proceed with the development of the theory and assume that it is understood what is meant by "machine", "tape", "scanned", etc.

2. *Definitions.*

Automatic machines.

If at each stage the motion of a machine (in the sense of § 1) is *completely* determined by the configuration, we shall call the machine an "automatic machine" (or a-machine).

For some purposes we might use machines (choice machines or c-machines) whose motion is only partially determined by the configuration (hence the use of the word "possible" in § 1). When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator. This would be the case if we were using machines to deal with axiomatic systems. In this paper I deal only with automatic machines, and will therefore often omit the prefix a-.

Computing machines.

If an a-machine prints two kinds of symbols, of which the first kind (called figures) consists entirely of 0 and 1 (the others being called symbols of the second kind), then the machine will be called a computing machine. If the machine is supplied with a blank tape and set in motion, starting from the correct initial m-configuration, the subsequence of the symbols printed by it which are of the first kind will be called the *sequence computed by the machine*. The real number whose expression as a binary decimal is obtained by prefacing this sequence by a decimal point is called the *number computed by the machine*.

At any stage of the motion of the machine, the number of the scanned square, the complete sequence of all symbols on the tape, and the m-configuration will be said to describe the *complete configuration* at that stage. The changes of the machine and tape between successive complete configurations will be called the *moves* of the machine.

⟦20⟧

Circular and circle-free machines.

If a computing machine never writes down more than a finite number of symbols of the first kind, it will be called *circular*. Otherwise it is said to be *circle-free*.

A machine will be circular if it reaches a configuration from which there is no possible move, or if it goes on moving, and possibly printing symbols of the second kind, but cannot print any more symbols of the first kind. The significance of the term "circular" will be explained in § 8.

Computable sequences and numbers.

A sequence is said to be computable if it can be computed by a circle-free machine. A number is computable if it differs by an integer from the number computed by a circle-free machine.

We shall avoid confusion by speaking more often of computable sequences than of computable numbers.

3. *Examples of computing machines.*

I. A machine can be constructed to compute the sequence 010101.... The machine is to have the four m-configurations "$\mathfrak{b}$", "$\mathfrak{c}$", "$\mathfrak{f}$", "$\mathfrak{e}$" and is capable of printing "0" and "1". The behaviour of the machine is described in the following table in which "R" means "the machine moves so that it scans the square immediately on the right of the one it was scanning previously". Similarly for "L". "E" means "the scanned symbol is erased" and "P" stands for "prints". This table (and all succeeding tables of the same kind) is to be understood to mean that for a configuration described in the first two columns the operations in the third column are carried out successively, and the machine then goes over into the m-configuration described in the last column. When the second column is left blank, it is understood that the behaviour of the third and fourth columns applies for any symbol and for no symbol. The machine starts in the m-configuration $\mathfrak{b}$ with a blank tape.

Configuration		*Behaviour*	
m-config.	*symbol*	*operations*	*final m-config.*
$\mathfrak{b}$	None	$P0,\ R$	$\mathfrak{c}$
$\mathfrak{c}$	None	R	$\mathfrak{e}$
$\mathfrak{e}$	None	$P1,\ R$	$\mathfrak{f}$
$\mathfrak{f}$	None	R	$\mathfrak{b}$

If (contrary to the description in § 1) we allow the letters L, R to appear more than once in the operations column we can simplify the table considerably.

m-config.	*symbol*	*operations*	*final m-config.*
𝔟	None	$P0$	𝔟
	0	$R, R, P1$	𝔟
	1	$R, R, P0$	𝔟

II. As a slightly more difficult example we can construct a machine to compute the sequence 001011011101111011111.... The machine is to be capable of five m-configurations, viz. "𝔬", "𝔮", "𝔭", "𝔣", "𝔟" and of printing "ə", "x", "0", "1". The first three symbols on the tape will be "əə0"; the other figures follow on alternate squares. On the intermediate squares we never print anything but "x". These letters serve to "keep the place" for us and are erased when we have finished with them. We also arrange that in the sequence of figures on alternate squares there shall be no blanks.

Configuration		*Behaviour*	
m-config.	*symbol*	*operations*	*final m-config.*
𝔟		$Pə, R, Pə, R, P0, R, R, P0, L, L$	𝔬
𝔬	1	R, Px, L, L, L	𝔬
	0		𝔮
𝔮	Any (0 or 1)	R, R	𝔮
	None	$P1, L$	𝔭
𝔭	x	E, R	𝔮
	ə	R	𝔣
	None	L, L	𝔭
𝔣	Any	R, R	𝔣
	None	$P0, L, L$	𝔬

To illustrate the working of this machine a table is given below of the first few complete configurations. These complete configurations are described by writing down the sequence of symbols which are on the tape, with the m-configuration written below the scanned symbol. The successive complete configurations are separated by colons.

```
  : ə ə 0   0 : ə ə 0   0 : ə ə 0   0 : ə ə 0   0     : ə ə 0   0   1 :
𝔟       𝔬           𝔮           𝔮           𝔮              𝔭
ə ə 0   0   1 : ə ə 0   0   1 : ə ə 0   0   1 : ə ə 0   0   1 :
      𝔭           𝔭           𝔣           𝔣
ə ə 0   0   1 : ə ə 0   0   1     : ə ə 0   0   1   0 :
          𝔣           𝔣           𝔬
ə ə 0   0   1 x 0 : ....
      𝔬
```

This table could also be written in the form

𝔟 : ə ə 𝔬 0   0 : ə ə 𝔮 0   0 : ..., (C)

in which a space has been made on the left of the scanned symbol and the m-configuration written in this space. This form is less easy to follow, but we shall make use of it later for theoretical purposes.

The convention of writing the figures only on alternate squares is very useful: I shall always make use of it. I shall call the one sequence of alternate squares F-squares and the other sequence E-squares. The symbols on E-squares will be liable to erasure. The symbols on F-squares form a continuous sequence. There are no blanks until the end is reached. There is no need to have more than one E-square between each pair of F-squares: an apparent need of more E-squares can be satisfied by having a sufficiently rich variety of symbols capable of being printed on E-squares. If a symbol β is on an F-square S and a symbol α is on the E-square next on the right of S, then S and β will be said to be *marked* with α. The process of printing this α will be called marking β (or S) with α.

4. *Abbreviated tables.*

There are certain types of process used by nearly all machines, and these, in some machines, are used in many connections. These processes include copying down sequences of symbols, comparing sequences, erasing all symbols of a given form, etc. Where such processes are concerned we can abbreviate the tables for the m-configurations considerably by the use of "skeleton tables". In skeleton tables there appear capital German letters and small Greek letters. These are of the nature of "variables". By replacing each capital German letter throughout by an m-configuration and each small Greek letter by a symbol, we obtain the table for an m-configuration.

The skeleton tables are to be regarded as nothing but abbreviations: they are not essential. So long as the reader understands how to obtain the complete tables from the skeleton tables, there is no need to give any exact definitions in this connection.

Let us consider an example:

m-config.	*Symbol*	*Behaviour*	*Final m-config.*
$\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$	ə	L	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	not ə	L	$\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$
$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$	α		$\mathfrak{C}$
	not α	R	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	None	R	$\mathfrak{f}_2(\mathfrak{C}, \mathfrak{B}, \alpha)$
$\mathfrak{f}_2(\mathfrak{C}, \mathfrak{B}, \alpha)$	α		$\mathfrak{C}$
	not α	R	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	None	R	$\mathfrak{B}$

From the m-configuration $\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$ the machine finds the symbol of form α which is farthest to the left (the "first α") and the m-configuration then becomes $\mathfrak{C}$. If there is no α then the m-configuration becomes $\mathfrak{B}$.

If we were to replace $\mathfrak{C}$ throughout by $\mathfrak{q}$ (say), $\mathfrak{B}$ by $\mathfrak{r}$, and α by x, we should have a complete table for the m-configuration $\mathfrak{f}(\mathfrak{q}, \mathfrak{r}, x)$. $\mathfrak{f}$ is called an "m-configuration function" or "m-function".

The only expressions which are admissible for substitution in an m-function are the m-configurations and symbols of the machine. These have to be enumerated more or less explicitly: they may include expressions such as $\mathfrak{p}(\mathfrak{e}, x)$; indeed they must if there are any m-functions used at all. If we did not insist on this explicit enumeration, but simply stated that the machine had certain m-configurations (enumerated) and all m-configurations obtainable by substitution of m-configurations in certain m-functions, we should usually get an infinity of m-configurations; *e.g.*, we might say that the machine was to have the m-configuration $\mathfrak{q}$ and all m-configurations obtainable by substituting an m-configuration for $\mathfrak{C}$ in $\mathfrak{p}(\mathfrak{C})$. Then it would have $\mathfrak{q}$, $\mathfrak{p}(\mathfrak{q})$, $\mathfrak{p}\big(\mathfrak{p}(\mathfrak{q})\big)$, $\mathfrak{p}\Big(\mathfrak{p}\big(\mathfrak{p}(\mathfrak{q})\big)\Big)$, ... as m-configurations.

Our interpretation rule then is this. We are given the names of the m-configurations of the machine, mostly expressed in terms of m-functions. We are also given skeleton tables. All we want is the complete table for the m-configurations of the machine. This is obtained by repeated substitution in the skeleton tables.

Further examples.

(In the explanations the symbol "$\rightarrow$" is used to signify "the machine goes into the m-configuration. . . .")

$\mathfrak{e}(\mathfrak{C}, \mathfrak{B}, \alpha)$			$\mathfrak{f}\big(\mathfrak{e}_1(\mathfrak{C}, \mathfrak{B}, \alpha), \mathfrak{B}, \alpha\big)$	From $\mathfrak{e}(\mathfrak{C}, \mathfrak{B}, \alpha)$ the first α is erased and $\rightarrow \mathfrak{C}$. If there is no $\alpha \rightarrow \mathfrak{B}$.
$\mathfrak{e}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$		E	$\mathfrak{C}$	
$\mathfrak{e}(\mathfrak{B}, \alpha)$			$\mathfrak{e}\big(\mathfrak{e}(\mathfrak{B}, \alpha), \mathfrak{B}, \alpha\big)$	From $\mathfrak{e}(\mathfrak{B}, \alpha)$ all letters α are erased and $\rightarrow \mathfrak{B}$.

The last example seems somewhat more difficult to interpret than most. Let us suppose that in the list of m-configurations of some machine there appears $\mathfrak{e}(\mathfrak{b}, x)$ ($= \mathfrak{q}$, say). The table is

	$\mathfrak{e}(\mathfrak{b}, x)$	$\mathfrak{e}\big(\mathfrak{e}(\mathfrak{b}, x), \mathfrak{b}, x\big)$
or	$\mathfrak{q}$	$\mathfrak{e}(\mathfrak{q}, \mathfrak{b}, x)$.

Or, in greater detail:

$\mathfrak{q}$		$\mathfrak{e}(\mathfrak{q}, \mathfrak{b}, x)$
$\mathfrak{e}(\mathfrak{q}, \mathfrak{b}, x)$		$\mathfrak{f}\big(\mathfrak{e}_1(\mathfrak{q}, \mathfrak{b}, x), \mathfrak{b}, x\big)$
$\mathfrak{e}_1(\mathfrak{q}, \mathfrak{b}, x)$	E	$\mathfrak{q}$.

In this we could replace $\mathfrak{e}_1(\mathfrak{q}, \mathfrak{b}, x)$ by $\mathfrak{q}'$ and then give the table for $\mathfrak{f}$ (with the right substitutions) and eventually reach a table in which no m-functions appeared.

$\mathfrak{pe}(\mathfrak{C}, \beta)$			$\mathfrak{f}\big(\mathfrak{pe}_1(\mathfrak{C}, \beta), \mathfrak{C}, ə\big)$	From $\mathfrak{pe}\ (\mathfrak{C}, \beta)$ the machine prints β at the end of the sequence of symbols and $\rightarrow \mathfrak{C}$.
$\mathfrak{pe}_1(\mathfrak{C}, \beta)$	Any	R, R	$\mathfrak{pe}_1(\mathfrak{C}, \beta)$	
	None	$P\beta$	$\mathfrak{C}$	
$\mathfrak{l}(\mathfrak{C})$		L	$\mathfrak{C}$	From $\mathfrak{f}'(\mathfrak{C}, \mathfrak{B}, \alpha)$ it does the same as for $\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$ but moves to the left before $\rightarrow \mathfrak{C}$.
$\mathfrak{r}(\mathfrak{C})$		R	$\mathfrak{C}$	
$\mathfrak{f}'(\mathfrak{C}, \mathfrak{B}, \alpha)$			$\mathfrak{f}\big(\mathfrak{l}(\mathfrak{C}), \mathfrak{B}, \alpha\big)$	
$\mathfrak{f}''(\mathfrak{C}, \mathfrak{B}, \alpha)$			$\mathfrak{f}\big(\mathfrak{r}(\mathfrak{C}), \mathfrak{B}, \alpha\big)$	

$\mathfrak{c}(\mathfrak{C}, \mathfrak{B}, \alpha)$		$\mathfrak{f}'\big(\mathfrak{c}_1(\mathfrak{C}), \mathfrak{B}, \alpha\big)$	$\mathfrak{c}(\mathfrak{C}, \mathfrak{B}, \alpha)$. The machine writes at the end the first symbol marked α and $\rightarrow \mathfrak{C}$.
$\mathfrak{c}_1(\mathfrak{C})$	β	$\mathfrak{pe}(\mathfrak{C}, \beta)$	

The last line stands for the totality of lines obtainable from it by replacing β by any symbol which may occur on the tape of the machine concerned.

$\mathfrak{ce}(\mathfrak{C}, \mathfrak{B}, \alpha)$		$\mathfrak{c}\big(\mathfrak{e}(\mathfrak{C}, \mathfrak{B}, \alpha), \mathfrak{B}, \alpha\big)$
$\mathfrak{ce}(\mathfrak{B}, \alpha)$		$\mathfrak{ce}\big(\mathfrak{ce}(\mathfrak{B}, \alpha), \mathfrak{B}, \alpha\big)$

$\mathfrak{ce}(\mathfrak{B}, \alpha)$. The machine copies down in order at the end all symbols marked α and erases the letters α; $\rightarrow \mathfrak{B}$.

$\mathfrak{re}(\mathfrak{C}, \mathfrak{B}, \alpha, \beta)$		$\mathfrak{f}\big(\mathfrak{re}_1(\mathfrak{C}, \mathfrak{B}, \alpha, \beta), \mathfrak{B}, \alpha\big)$
$\mathfrak{re}_1(\mathfrak{C}, \mathfrak{B}, \alpha, \beta)$	$E, P\beta$	$\mathfrak{C}$
$\mathfrak{re}(\mathfrak{B}, \alpha, \beta)$		$\mathfrak{re}\big(\mathfrak{re}(\mathfrak{B}, \alpha, \beta), \mathfrak{B}, \alpha, \beta\big)$

$\mathfrak{re}(\mathfrak{C}, \mathfrak{B}, \alpha, \beta)$. The machine replaces the first α by β and $\rightarrow \mathfrak{C} \rightarrow \mathfrak{B}$ if there is no α. $\mathfrak{re}(\mathfrak{B}, \alpha, \beta)$. The machine replaces all letters α by β; $\rightarrow \mathfrak{B}$.

$\mathfrak{cr}(\mathfrak{C}, \mathfrak{B}, \alpha)$		$\mathfrak{c}\big(\mathfrak{re}(\mathfrak{C}, \mathfrak{B}, \alpha, a), \mathfrak{B}, \alpha\big)$
$\mathfrak{cr}(\mathfrak{B}, \alpha)$		$\mathfrak{cr}\big(\mathfrak{cr}(\mathfrak{B}, \alpha), \mathfrak{re}(\mathfrak{B}, a, \alpha), \alpha\big)$

$\mathfrak{cr}(\mathfrak{B}, \alpha)$ differs from $\mathfrak{ce}(\mathfrak{B}, \alpha)$ only in that the letters α are not erased. The m-configuration $\mathfrak{cr}(\mathfrak{B}, \alpha)$ is taken up when no letters "a" are on the tape.

$\mathfrak{cp}(\mathfrak{C}, \mathfrak{A}, \mathfrak{E}, \alpha, \beta)$		$\mathfrak{f}'\big(\mathfrak{cp}_1(\mathfrak{C}_1 \mathfrak{A}, \beta), \mathfrak{f}(\mathfrak{A}, \mathfrak{E}, \beta), \alpha\big)$
$\mathfrak{cp}_1(\mathfrak{C}, \mathfrak{A}, \beta)$	γ	$\mathfrak{f}'\big(\mathfrak{cp}_2(\mathfrak{C}, \mathfrak{A}, \gamma), \mathfrak{A}, \beta\big)$
$\mathfrak{cp}_2(\mathfrak{C}, \mathfrak{A}, \gamma)$	γ	$\mathfrak{C}$
	not γ	$\mathfrak{A}$.

The first symbol marked α and the first marked β are compared. If there is neither α nor β, $\rightarrow \mathfrak{E}$. If there are both and the symbols are alike, $\rightarrow \mathfrak{C}$. Otherwise $\rightarrow \mathfrak{A}$.

$$\mathfrak{cpe}(\mathfrak{C}, \mathfrak{A}, \mathfrak{E}, \alpha, \beta) \qquad \mathfrak{cp}\Big(\mathfrak{e}\big(\mathfrak{e}(\mathfrak{C}, \mathfrak{C}, \beta), \mathfrak{C}, \alpha\big), \mathfrak{A}, \mathfrak{E}, \alpha, \beta\Big)$$

$\mathfrak{cpe}(\mathfrak{C}, \mathfrak{A}, \mathfrak{E}, \alpha, \beta)$ differs from $\mathfrak{cp}(\mathfrak{C}, \mathfrak{A}, \mathfrak{E}, \alpha, \beta)$ in that in the case when there is similarity the first α and β are erased.

$$\mathfrak{cpe}(\mathfrak{A}, \mathfrak{E}, \alpha, \beta) \qquad \mathfrak{cpe}\big(\mathfrak{cpe}(\mathfrak{A}, \mathfrak{E}, \alpha, \beta), \mathfrak{A}, \mathfrak{E}, \alpha, \beta\big).$$

$\mathfrak{cpe}(\mathfrak{A}, \mathfrak{E}, \alpha, \beta)$. The sequence of symbols marked α is compared with the sequence marked β. $\rightarrow \mathfrak{E}$ if they are similar. Otherwise $\rightarrow \mathfrak{A}$. Some of the symbols α and β are erased.

$\mathfrak{q}(\mathfrak{C})$	Any	R	$\mathfrak{q}(\mathfrak{C})$	$\mathfrak{q}(\mathfrak{C}, \alpha)$. The machine finds the last symbol of form α. $\rightarrow \mathfrak{C}$.
	None	R	$\mathfrak{q}_1(\mathfrak{C})$	
$\mathfrak{q}_1(\mathfrak{C})$	Any	R	$\mathfrak{q}(\mathfrak{C})$	
	None		$\mathfrak{C}$	
$\mathfrak{q}(\mathfrak{C}, \alpha)$			$\mathfrak{q}\big(\mathfrak{q}_1(\mathfrak{C}, \alpha)\big)$	
$\mathfrak{q}_1(\mathfrak{C}, \alpha)$	α		$\mathfrak{C}$	
	not α	L	$\mathfrak{q}_1(\mathfrak{C}, \alpha)$	
$\mathfrak{pe}_2(\mathfrak{C}, \alpha, \beta)$			$\mathfrak{pe}\big(\mathfrak{pe}(\mathfrak{C}, \beta), \alpha\big)$	$\mathfrak{pe}_2(\mathfrak{C}, \alpha, \beta)$. The machine prints $\alpha\ \beta$ at the end.
$\mathfrak{ce}_2(\mathfrak{B}, \alpha, \beta)$			$\mathfrak{ce}\big(\mathfrak{ce}(\mathfrak{B}, \beta), \alpha\big)$	$\mathfrak{ce}_3(\mathfrak{B}, \alpha, \beta, \gamma)$. The machine copies down at the end first the symbols marked α, then those marked β, and finally those marked γ; it erases the symbols α, β, γ.
$\mathfrak{ce}_3(\mathfrak{B}, \alpha, \beta, \gamma)$			$\mathfrak{ce}\big(\mathfrak{ce}_2(\mathfrak{B}, \beta, \gamma), \alpha\big)$	
$\mathfrak{e}(\mathfrak{C})$	ə	R	$\mathfrak{e}_1(\mathfrak{C})$	From $\mathfrak{e}(\mathfrak{C})$ the marks are erased from all marked symbols. $\rightarrow \mathfrak{C}$.
	Not ə	L	$\mathfrak{e}(\mathfrak{C})$	
$\mathfrak{e}_1(\mathfrak{C})$	Any	R, E, R	$\mathfrak{e}_1(\mathfrak{C})$	
	None		$\mathfrak{C}$	

5. *Enumeration of computable sequences.*

A computable sequence γ is determined by a description of a machine which computes γ. Thus the sequence 001011011101111... is determined by the table on p. 234, and, in fact, any computable sequence is capable of being described in terms of such a table.

It will be useful to put these tables into a kind of standard form. In the first place let us suppose that the table is given in the same form as the first table, for example, I on p. 233. That is to say, that the entry in the operations column is always of one of the forms $E : E$, $R : E$, $L : Pa : Pa$, $R : Pa$, $L : R : L :$ or no entry at all. The table can always be put into this form by introducing more m-configurations. Now let us give numbers to the m-configurations, calling them $q_1, \ldots, q_R$, as in § 1. The initial m-configuration is always to be called q_1. We also give numbers to the symbols $S_1, \ldots, S_m$

and, in particular, blank $= S_0$, $0 = S_1$, $1 = S_2$. The lines of the table are now of form

m-config.	*Symbol*	*Operations*	*Final m-config.*	
q_i	S_j	PS_k, L	q_m	(N_1)
q_i	S_j	PS_k, R	q_m	(N_2)
q_i	S_j	PS_k	q_m	(N_3)

Lines such as

q_i	S_j	E, R	q_m

are to be written as

q_i	S_j	PS_0, R	q_m

and lines such as

q_i	S_j	R	q_m

to be written as

q_i	S_j	PS_j, R	q_m

In this way we reduce each line of the table to a line of one of the forms (N_1), (N_2), (N_3).

From each line of form (N_1) let us form an expression $q_i S_j S_k L q_m$; from each line of form (N_2) we form an expression $q_i S_j S_k R q_m$; and from each line of form (N_3) we form an expression $q_i S_j S_k N q_m$.

Let us write down all expressions so formed from the table for the machine and separate them by semi-colons. In this way we obtain a complete description of the machine. In this description we shall replace q_i by the letter "*D*" followed by the letter "*A*" repeated i times, and S_j by "*D*" followed by "*C*" repeated j times. This new description of the machine may be called the *standard description* (S.D). It is made up entirely from the letters "*A*", "*C*", "*D*", "*L*", "*R*", "*N*", and from ";".

If finally we replace "*A*" by "1", "*C*" by "2", "*D*" by "3", "*L*" by "4", "*R*" by "5", "*N*" by "6", and ";" by "7" we shall have a description of the machine in the form of an arabic numeral. The integer represented by this numeral may be called a *description number* (D.N) of the machine. The D.N determine the S.D and the structure of the machine uniquely. The machine whose D.N is n may be described as $\mathcal{M}(n)$.

To each computable sequence there corresponds at least one description number, while to no description number does there correspond more than one computable sequence. The computable sequences and numbers are therefore enumerable.

Let us find a description number for the machine I of § 3. When we rename the m-configurations its table becomes:

q_1	S_0	PS_1, R	q_2
q_2	S_0	PS_0, R	q_3
q_3	S_0	PS_2, R	q_4
q_4	S_0	PS_0, R	q_1

Other tables could be obtained by adding irrelevant lines such as

q_1	S_1	PS_1, R	q_2

Our first standard form would be

$$q_1 S_0 S_1 R q_2;\ q_2 S_0 S_0 R q_3;\ q_3 S_0 S_2 R q_4;\ q_4 S_0 S_0 R q_1;.$$

The standard description is

DADDCRDAA ;*DAADDRDAAA* ;

DAAADDCCRDAAAA ;*DAAAADDRDA* ;

A description number is

31332531173113353111731113322531111731111335317

and so is

3133253117311335311173111332253111173111133531731323253117

A number which is a description number of a circle-free machine will be called a *satisfactory* number. In § 8 it is shown that there can be no general process for determining whether a given number is satisfactory or not.

6. *The universal computing machine.*

It is possible to invent a single machine which can be used to compute any computable sequence. If this machine $\mathcal{U}$ is supplied with a tape on the beginning of which is written the S.D of some computing machine $\mathcal{M}$, then $\mathcal{U}$ will compute the same sequence as $\mathcal{M}$. In this section I explain in outline the behaviour of the machine. The next section is devoted to giving the complete table for $\mathcal{U}$.

Let us first suppose that we have a machine $\mathcal{M}'$ which will write down on the F-squares the successive complete configurations of $\mathcal{M}$. These might be expressed in the same form as on p. 235, using the second description, (C), with all symbols on one line. Or, better, we could transform this description (as in § 5) by replacing each m-configuration by "D" followed by "A" repeated the appropriate number of times, and by replacing each symbol by "D" followed by "C" repeated the appropriate number of times. The numbers of letters "A" and "C" are to agree with the numbers chosen in § 5, so that, in particular, "0" is replaced by "DC", "1" by "DCC", and the blanks by "D". These substitutions are to be made after the complete configurations have been put together, as in (C). Difficulties arise if we do the substitution first. In each complete configuration the blanks would all have to be replaced by "D", so that the complete configuration would not be expressed as a finite sequence of symbols.

If in the description of the machine II of § 3 we replace "$\mathfrak{o}$" by "DAA", "ə" by "$DCCC$", "$\mathfrak{q}$" by "$DAAA$", then the sequence (C) becomes:

$$DA:DCCCDCCCDAADCDDC:DCCCDCCCDAAADCDDC:\ldots \quad (C_1)$$

(This is the sequence of symbols on F-squares.)

It is not difficult to see that if $\mathcal{M}$ can be constructed, then so can $\mathcal{M}'$. The manner of operation of $\mathcal{M}'$ could be made to depend on having the rules of operation (*i.e.*, the S.D) of $\mathcal{M}$ written somewhere within itself (*i.e.* within $\mathcal{M}'$); each step could be carried out by referring to these rules. We have only to regard the rules as being capable of being taken out and exchanged for others and we have something very akin to the universal machine.

One thing is lacking: at present the machine $\mathcal{M}'$ prints no figures. We may correct this by printing between each successive pair of complete configurations the figures which appear in the new configuration but not in the old. Then (C_1) becomes

$$DDA:0:0:DCCCDCCCDAADCDDC:DCCC.\ldots \quad (C_2)$$

It is not altogether obvious that the E-squares leave enough room for the necessary "rough work", but this is, in fact, the case.

The sequences of letters between the colons in expressions such as (C_1) may be used as standard descriptions of the complete configurations. When the letters are replaced by figures, as in § 5, we shall have a numerical description of the complete configuration, which may be called its description number.

7. *Detailed description of the universal machine.*

A table is given below of the behaviour of this universal machine. The m-configurations of which the machine is capable are all those occurring in the first and last columns of the table, together with all those which occur when we write out the unabbreviated tables of those which appear in the table in the form of m-functions. *E.g.*, $\mathfrak{e}(\mathfrak{anf})$ appears in the table and is an m-function. Its unabbreviated table is (see p. 239)

$\mathfrak{e}(\mathfrak{anf})$	ə	R	$\mathfrak{e}_1(\mathfrak{anf})$
	not ə	L	$\mathfrak{e}(\mathfrak{anf})$
$\mathfrak{e}_1(\mathfrak{anf})$	Any	R, E, R	$\mathfrak{e}_1(\mathfrak{anf})$
	None		$\mathfrak{anf}$

Consequently $\mathfrak{e}_1(\mathfrak{anf})$ is an m-configuration of $\mathfrak{U}$.

When $\mathfrak{U}$ is ready to start work the tape running through it bears on it the symbol ə on an F-square and again ə on the next E-square; after this, on F-squares only, comes the S.D of the machine followed by a double colon "::" (a single symbol, on an F-square). The S.D consists of a number of instructions, separated by semi-colons.

Each instruction consists of five consecutive parts

(i) "D" followed by a sequence of letters "A". This describes the relevant m-configuration.

(ii) "D" followed by a sequence of letters "C". This describes the scanned symbol.

(iii) "D" followed by another sequence of letters "C". This describes the symbol into which the scanned symbol is to be changed.

(iv) "L", "R", or "N", describing whether the machine is to move to left, right, or not at all.

(v) "D" followed by a sequence of letters "A". This describes the final m-configuration.

The machine $\mathfrak{U}$ is to be capable of printing "A", "C", "D", "0", "1", "u", "v", "w", "x", "y", "z". The S.D is formed from ";", "A", "C", "D", "L", "R", "N".

Subsidiary skeleton table.

$\mathfrak{con}(\mathfrak{C}, \alpha)$	Not A	R, R	$\mathfrak{con}(\mathfrak{C}, \alpha)$
	A	L, Pa, R	$\mathfrak{con}_1(\mathfrak{C}, \alpha)$
$\mathfrak{con}_1(\mathfrak{C}, \alpha)$	A	R, Pa, R	$\mathfrak{con}_1(\mathfrak{C}, \alpha)$
	D	R, Pa, R	$\mathfrak{con}_2(\mathfrak{C}, \alpha)$
$\mathfrak{con}_2(\mathfrak{C}, \alpha)$	C	R, Pa, R	$\mathfrak{con}_2(\mathfrak{C}, \alpha)$
	Not C	R, R	$\mathfrak{C}$

$\mathfrak{con}(\mathfrak{C}, \alpha)$. Starting from an F-square, S say, the sequence C of symbols describing a configuration closest on the right of S is marked out with letters α. $\rightarrow \mathfrak{C}$.

$\mathfrak{con}(\mathfrak{C}, \ )$. In the final configuration the machine is scanning the square which is four squares to the right of the last square of C. C is left unmarked.

The table for $\mathfrak{U}$.

$\mathfrak{b}$		$\mathfrak{f}(\mathfrak{b}_1, \mathfrak{b}_1, ::)$
$\mathfrak{b}_1$	$R, R, P:, R, R, PD, R, R, PA$	$\mathfrak{anf}$

$\mathfrak{b}$. The machine prints $:DA$ on the F-squares after $::$ $\rightarrow \mathfrak{anf}$.

$\mathfrak{anf}$	$\mathfrak{g}(\mathfrak{anf}_1, :)$
$\mathfrak{anf}_1$	$\mathfrak{con}(\mathfrak{fom}, y)$

$\mathfrak{anf}$. The machine marks the configuration in the last complete configuration with y. $\rightarrow \mathfrak{fom}$.

$\mathfrak{fom}$	;	R, Pz, L	$\mathfrak{con}(\mathfrak{fmp}, x)$
	z	L, L	$\mathfrak{fom}$
	not z nor ;	L	$\mathfrak{fom}$

$\mathfrak{fom}$. The machine finds the last semi-colon not marked with z. It marks this semi-colon with z and the configuration following it with x.

$\mathfrak{fmp}$	$\mathfrak{cpe}\big(\mathfrak{e}(\mathfrak{fom}, x, y), \mathfrak{sim}, x, y\big)$

$\mathfrak{fmp}$. The machine compares the sequences marked x and y. It erases all letters x and y. $\rightarrow \mathfrak{sim}$ if they are alike. Otherwise $\rightarrow \mathfrak{fom}$.

$\mathfrak{anf}$. Taking the long view, the last instruction relevant to the last configuration is found. It can be recognised afterwards as the instruction following the last semi-colon marked z. $\rightarrow \mathfrak{sim}$.

$\mathfrak{sim}$			$\mathfrak{f}'(\mathfrak{sim}_1, \mathfrak{sim}_1, z)$
$\mathfrak{sim}_1$			$\mathfrak{con}(\mathfrak{sim}_2, \ )$
$\mathfrak{sim}_2$	A		$\mathfrak{sim}_3$
	not A	R, Pu, R, R, R	$\mathfrak{sim}_2$
$\mathfrak{sim}_3$	not A	L, Py	$\mathfrak{e}(\mathfrak{mk}, z)$
	A	L, Py, R, R, R	$\mathfrak{sim}_3$

$\mathfrak{sim}$. The machine marks out the instructions. That part of the instructions which refers to operations to be carried out is marked with u, and the final m-configuration with y. The letters z are erased.

$\mathfrak{mk}$			$\mathfrak{g}(\mathfrak{mk}, :)$
$\mathfrak{mk}_1$	not A	R, R	$\mathfrak{mk}_1$
	A	L, L, L, L	$\mathfrak{mk}_2$
$\mathfrak{mk}_2$	C	R, Px, L, L, L	$\mathfrak{mk}_2$
	:		$\mathfrak{mk}_4$
	D	R, Px, L, L, L	$\mathfrak{mk}_3$
$\mathfrak{mk}_3$	not :	R, Pv, L, L, L	$\mathfrak{mk}_3$
	:		$\mathfrak{mk}_4$
$\mathfrak{mk}_4$			$\mathfrak{con}\Big(\mathfrak{l}\big(\mathfrak{l}(\mathfrak{mk}_5)\big), \ \Big)$
$\mathfrak{mk}_5$	Any	R, Pw, R	$\mathfrak{mk}_5$
	None	$P:$	$\mathfrak{sh}$

$\mathfrak{mk}$. The last complete configuration is marked out into four sections. The configuration is left unmarked. The symbol directly preceding it is marked with x. The remainder of the complete configuration is divided into two parts, of which the first is marked with v and the last with w. A colon is printed after the whole. $\rightarrow \mathfrak{sh}$.

$\mathfrak{sh}$			$\mathfrak{f}(\mathfrak{sh}_1, \mathfrak{inst}, u)$
$\mathfrak{sh}_1$		L, L, L	$\mathfrak{sh}_2$
$\mathfrak{sh}_2$	D	R, R, R, R	$\mathfrak{sh}_2$
	not D		$\mathfrak{inst}$
$\mathfrak{sh}_3$	C	R, R	$\mathfrak{sh}_4$
	not C		$\mathfrak{inst}$
$\mathfrak{sh}_4$	C	R, R	$\mathfrak{sh}_5$
	not C		$\mathfrak{pe}_2(\mathfrak{inst}, 0, :)$
$\mathfrak{sh}_5$	C		$\mathfrak{inst}$
	not C		$\mathfrak{pe}_2(\mathfrak{inst}, 1, :)$

$\mathfrak{sh}$. The instructions (marked u) are examined. If it is found that they involve "Print 0" or "Print 1", then 0: or 1: is printed at the end.

$\mathfrak{inst}$			$\mathfrak{g}\left(\mathfrak{l}(\mathfrak{inst}_1), u\right)$	$\mathfrak{inst}$. The next complete configuration is written down, carrying out the marked instructions. The letters u, v, w, x, y are erased. $\rightarrow \mathfrak{anf}$.
$\mathfrak{inst}_1$	α	R, E	$\mathfrak{inst}_1(\alpha)$	
$\mathfrak{inst}_1(L)$			$\mathfrak{ce}_5(\mathfrak{ov}, v, y, x, u, w)$	
$\mathfrak{inst}_1(R)$			$\mathfrak{ce}_5(\mathfrak{ov}, v, x, u, y, w)$	
$\mathfrak{inst}_1(N)$			$\mathfrak{ec}_5(\mathfrak{ov}, v, x, y, u, w)$	
$\mathfrak{ov}$			$\mathfrak{e}(\mathfrak{anf})$	

8. *Application of the diagonal process.*

It may be thought that arguments which prove that the real numbers are not enumerable would also prove that the computable numbers and sequences cannot be enumerable*. It might, for instance, be thought that the limit of a sequence of computable numbers must be computable. This is clearly only true if the sequence of computable numbers is defined by some rule.

Or we might apply the diagonal process. "If the computable sequences are enumerable, let α_n be the n-th computable sequence, and let $\phi_n(m)$ be the m-th figure in α_n. Let β be the sequence with $1-\phi_n(n)$ as its n-th figure. Since β is computable, there exists a number K such that $1-\phi_n(n) = \phi_K(n)$ all n. Putting $n = K$, we have $1 = 2\phi_K(K)$, *i.e.* 1 is even. This is impossible. The computable sequences are therefore not enumerable".

The fallacy in this argument lies in the assumption that β is computable. It would be true if we could enumerate the computable sequences by finite means, but the problem of enumerating computable sequences is equivalent to the problem of finding out whether a given number is the D.N of a circle-free machine, and we have no general process for doing this in a finite number of steps. In fact, by applying the diagonal process argument correctly, we can show that there cannot be any such general process.

The simplest and most direct proof of this is by showing that, if this general process exists, then there is a machine which computes β. This proof, although perfectly sound, has the disadvantage that it may leave the reader with a feeling that "there must be something wrong". The proof which I shall give has not this disadvantage, and gives a certain insight into the significance of the idea "circle-free". It depends not on constructing β, but on constructing β', whose n-th figure is $\phi_n(n)$.

* *Cf.* Hobson, *Theory of functions of a real variable* (2nd ed., 1921), 87, 88.

Let us suppose that there is such a process; that is to say, that we can invent a machine $\mathfrak{D}$ which, when supplied with the S.D. of any computing machine $\mathcal{M}$ will test this S.D and if $\mathcal{M}$ is circular will mark the S.D with the symbol "u" and if it is circle-free will mark it with "s". By combining the machines $\mathfrak{D}$ and $\mathfrak{U}$ we could construct a machine $\mathfrak{H}$ to compute the sequence β'. The machine $\mathfrak{D}$ may require a tape. We may suppose that it uses the E-squares beyond all symbols on F-squares, and that when it has reached its verdict all the rough work done by $\mathfrak{D}$ is erased.

The machine $\mathfrak{H}$ has its motion divided into sections. In the first $N-1$ sections, among other things, the integers $1, 2, \ldots, N-1$ have been written down and tested by the machine $\mathfrak{D}$. A certain number, say $R(N-1)$, of them have been found to be the D.N's of circle-free machines. In the N-th section the machine $\mathfrak{D}$ tests the number N. If N is satisfactory, *i.e.*, if it is the D.N of a circle-free machine, then $R(N) = 1+R(N-1)$ and the first $R(N)$ figures of the sequence of which a D.N is N are calculated. The $R(N)$-th figure of this sequence is written down as one of the figures of the sequence β' computed by $\mathfrak{H}$. If N is not satisfactory, then $R(N) = R(N-1)$ and the machine goes on to the $(N+1)$-th section of its motion.

From the construction of $\mathfrak{H}$ we can see that $\mathfrak{H}$ is circle-free. Each section of the motion of $\mathfrak{H}$ comes to an end after a finite number of steps. For, by our assumption about $\mathfrak{D}$, the decision as to whether N is satisfactory is reached in a finite number of steps. If N is not satisfactory, then the N-th section is finished. If N is satisfactory, this means that the machine $\mathcal{M}(N)$ whose D.N is N is circle-free, and therefore its $R(N)$-th figure can be calculated in a finite number of steps. When this figure has been calculated and written down as the $R(N)$-th figure of β', the N-th section is finished. Hence $\mathfrak{H}$ is circle-free.

Now let K be the D.N of $\mathfrak{H}$. What does $\mathfrak{H}$ do in the K-th section of its motion? It must test whether K is satisfactory, giving a verdict "s" or "u". Since K is the D.N of $\mathfrak{H}$ and since $\mathfrak{H}$ is circle-free, the verdict cannot be "u". On the other hand the verdict cannot be "s". For if it were, then in the K-th section of its motion $\mathfrak{H}$ would be bound to compute the first $R(K-1)+1 = R(K)$ figures of the sequence computed by the machine with K as its D.N and to write down the $R(K)$-th as a figure of the sequence computed by $\mathfrak{H}$. The computation of the first $R(K)-1$ figures would be carried out all right, but the instructions for calculating the $R(K)$-th would amount to "calculate the first $R(K)$ figures computed by H and write down the $R(K)$-th". This $R(K)$-th figure would never be found. *I.e.*, $\mathfrak{H}$ is circular, contrary both to what we have found in the last paragraph and to the verdict "s". Thus both verdicts are impossible and we conclude that there can be no machine $\mathfrak{D}$.

We can show further that *there can be no machine $\mathfrak{E}$ 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*).

We will first show that, if there is a machine $\mathfrak{E}$, then there is a general process for determining whether a given machine $\mathfrak{M}$ prints 0 infinitely often. Let $\mathfrak{M}_1$ be a machine which prints the same sequence as $\mathfrak{M}$, except that in the position where the first 0 printed by $\mathfrak{M}$ stands, $\mathfrak{M}_1$ prints $\bar{0}$. $\mathfrak{M}_2$ is to have the first two symbols 0 replaced by $\bar{0}$, and so on. Thus, if $\mathfrak{M}$ were to print

$$A\,B\,A\,0\,1\,A\,A\,B\,0\,0\,1\,0\,A\,B\ldots,$$

then $\mathfrak{M}_1$ would print

$$A\,B\,A\,\bar{0}\,1\,A\,A\,B\,0\,0\,1\,0\,A\,B\ldots$$

and $\mathfrak{M}_2$ would print

$$A\,B\,A\,\bar{0}\,1\,A\,A\,B\,\bar{0}\,0\,1\,0\,A\,B\ldots.$$

Now let $\mathfrak{F}$ be a machine which, when supplied with the S.D of $\mathfrak{M}$, will write down successively the S.D of $\mathfrak{M}$, of $\mathfrak{M}_1$, of $\mathfrak{M}_2$, ... (there is such a machine). We combine $\mathfrak{F}$ with $\mathfrak{E}$ and obtain a new machine, $\mathfrak{G}$. In the motion of $\mathfrak{G}$ first $\mathfrak{F}$ is used to write down the S.D of $\mathfrak{M}$, and then $\mathfrak{E}$ tests it, : 0 : is written if it is found that $\mathfrak{M}$ never prints 0; then $\mathfrak{F}$ writes the S.D of $\mathfrak{M}_1$, and this is tested, : 0 : being printed if and only if $\mathfrak{M}_1$ never prints 0, and so on. Now let us test $\mathfrak{G}$ with $\mathfrak{E}$. If it is found that $\mathfrak{G}$ never prints 0, then $\mathfrak{M}$ prints 0 infinitely often; if $\mathfrak{G}$ prints 0 sometimes, then $\mathfrak{M}$ does not print 0 infinitely often.

Similarly there is a general process for determining whether $\mathfrak{M}$ prints 1 infinitely often. By a combination of these processes we have a process for determining whether $\mathfrak{M}$ prints an infinity of figures, *i.e.* we have a process for determining whether $\mathfrak{M}$ is circle-free. There can therefore be no machine $\mathfrak{E}$.

The expression "there is a general process for determining ..." has been used throughout this section as equivalent to "there is a machine which will determine ...". This usage can be justified if and only if we can justify our definition of "computable". For each of these "general process" problems can be expressed as a problem concerning a general process for determining whether a given integer n has a property $G(n)$ [*e.g.* $G(n)$ might mean "n is satisfactory" or "n is the Gödel representation of a provable formula"], and this is equivalent to computing a number whose n-th figure is 1 if $G(n)$ is true and 0 if it is false.

9. *The extent of the computable numbers.*

No attempt has yet been made to show that the "computable" numbers include all numbers which would naturally be regarded as computable. All arguments which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically. The real question at issue is "What are the possible processes which can be carried out in computing a number?"

The arguments which I shall use are of three kinds.

(*a*) A direct appeal to intuition.

(*b*) A proof of the equivalence of two definitions (in case the new definition has a greater intuitive appeal).

(*c*) Giving examples of large classes of numbers which are computable.

Once it is granted that computable numbers are all "computable", several other propositions of the same character follow. In particular, it follows that, if there is a general process for determining whether a formula of the Hilbert function calculus is provable, then the determination can be carried out by a machine.

I. [Type (*a*)]. This argument is only an elaboration of the ideas of § 1.

Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, *i.e.* on a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite. If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent†. The effect of this restriction of the number of symbols is not very serious. It is always possible to use sequences of symbols in the place of single symbols. Thus an Arabic numeral such as

† If we regard a symbol as literally printed on a square we may suppose that the square is $0 \leqslant x \leqslant 1$, $0 \leqslant y \leqslant 1$. The symbol is defined as a set of points in this square, viz. the set occupied by printer's ink. If these sets are restricted to be measurable, we can define the "distance" between two symbols as the cost of transforming one symbol into the other if the cost of moving unit area of printer's ink unit distance is unity, and there is an infinite supply of ink at $x = 2$, $y = 0$. With this topology the symbols form a conditionally compact space.

17 or 999999999999999 is normally treated as a single symbol. Similarly in any European language words are treated as single symbols (Chinese, however, attempts to have an enumerable infinity of symbols). The differences from our point of view between the single and compound symbols is that the compound symbols, if they are too lengthy, cannot be observed at one glance. This is in accordance with experience. We cannot tell at a glance whether 9999999999999999 and 999999999999999 are the same.

The behaviour of the computer at any moment is determined by the symbols which he is observing, and his "state of mind" at that moment. We may suppose that there is a bound B to the number of symbols or squares which the computer can observe at one moment. If he wishes to observe more, he must use successive observations. We will also suppose that the number of states of mind which need be taken into account is finite. The reasons for this are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be "arbitrarily close" and will be confused. Again, the restriction is not one which seriously affects computation, since the use of more complicated states of mind can be avoided by writing more symbols on the tape.

Let us imagine the operations performed by the computer to be split up into "simple operations" which are so elementary that it is not easy to imagine them further divided. Every such operation consists of some change of the physical system consisting of the computer and his tape. We know the state of the system if we know the sequence of symbols on the tape, which of these are observed by the computer (possibly with a special order), and the state of mind of the computer. We may suppose that in a simple operation not more than one symbol is altered. Any other changes can be split up into simple changes of this kind. The situation in regard to the squares whose symbols may be altered in this way is the same as in regard to the observed squares. We may, therefore, without loss of generality, assume that the squares whose symbols are changed are always "observed" squares.

Besides these changes of symbols, the simple operations must include changes of distribution of observed squares. The new observed squares must be immediately recognisable by the computer. I think it is reasonable to suppose that they can only be squares whose distance from the closest of the immediately previously observed squares does not exceed a certain fixed amount. Let us say that each of the new observed squares is within L squares of an immediately previously observed square.

In connection with "immediate recognisability", it may be thought that there are other kinds of square which are immediately recognisable. In particular, squares marked by special symbols might be taken as immediately recognisable. Now if these squares are marked only by single symbols there can be only a finite number of them, and we should not upset our theory by adjoining these marked squares to the observed squares. If, on the other hand, they are marked by a sequence of symbols, we cannot regard the process of recognition as a simple process. This is a fundamental point and should be illustrated. In most mathematical papers the equations and theorems are numbered. Normally the numbers do not go beyond (say) 1000. It is, therefore, possible to recognise a theorem at a glance by its number. But if the paper was very long, we might reach Theorem 157767733443477; then, further on in the paper, we might find "... hence (applying Theorem 157767733443477) we have ...". In order to make sure which was the relevant theorem we should have to compare the two numbers figure by figure, possibly ticking the figures off in pencil to make sure of their not being counted twice. If in spite of this it is still thought that there are other "immediately recognisable" squares, it does not upset my contention so long as these squares can be found by some process of which my type of machine is capable. This idea is developed in III below.

The simple operations must therefore include:

(*a*) Changes of the symbol on one of the observed squares.

(*b*) Changes of one of the squares observed to another square within L squares of one of the previously observed squares.

It may be that some of these changes necessarily involve a change of state of mind. The most general single operation must therefore be taken to be one of the following:

(A) A possible change (*a*) of symbol together with a possible change of state of mind.

(B) A possible change (*b*) of observed squares, together with a possible change of state of mind.

The operation actually performed is determined, as has been suggested on p. 250, by the state of mind of the computer and the observed symbols. In particular, they determine the state of mind of the computer after the operation is carried out.

We may now construct a machine to do the work of this computer. To each state of mind of the computer corresponds an "m-configuration" of the machine. The machine scans B squares corresponding to the B squares observed by the computer. In any move the machine can change a symbol on a scanned square or can change any one of the scanned squares to another square distant not more than L squares from one of the other scanned squares. The move which is done, and the succeeding configuration, are determined by the scanned symbol and the m-configuration. The machines just described do not differ very essentially from computing machines as defined in § 2, and corresponding to any machine of this type a computing machine can be constructed to compute the same sequence, that is to say the sequence computed by the computer.

II. [Type (b)].

If the notation of the Hilbert functional calculus† is modified so as to be systematic, and so as to involve only a finite number of symbols, it becomes possible to construct an automatic‡ machine $\mathcal{K}$, which will find all the provable formulae of the calculus§.

Now let α be a sequence, and let us denote by $G_\alpha(x)$ the proposition "The x-th figure of α is 1", so that‖ $-G_\alpha(x)$ means "The x-th figure of α is 0". Suppose further that we can find a set of properties which define the sequence α and which can be expressed in terms of $G_\alpha(x)$ and of the propositional functions $N(x)$ meaning "x is a non-negative integer" and $F(x, y)$ meaning "$y = x+1$". When we join all these formulae together conjunctively, we shall have a formula, $\mathfrak{A}$ say, which defines α. The terms of $\mathfrak{A}$ must include the necessary parts of the Peano axioms, viz.,

$$(\exists u) N(u) \ \& \ (x) \big(N(x) \to (\exists y) F(x, y)\big) \ \& \ \big(F(x, y) \to N(y)\big),$$

which we will abbreviate to P.

When we say "$\mathfrak{A}$ defines α", we mean that $-\mathfrak{A}$ is not a provable formula, and also that, for each n, one of the following formulae (A_n) or (B_n) is provable.

$$\mathfrak{A} \ \& \ F^{(n)} \to G_\alpha(u^{(n)}), \qquad (A_n)¶$$

$$\mathfrak{A} \ \& \ F^{(n)} \to \big(-G_\alpha(u^{(n)})\big), \qquad (B_n),$$

where $F^{(n)}$ stands for $F(u, u') \ \& \ F(u', u'') \ \& \ldots F(u^{(n-1)}, u^{(n)})$.

† The expression "the functional calculus" is used throughout to mean the *restricted* Hilbert functional calculus.

‡ It is most natural to construct first a choice machine (§ 2) to do this. But it is then easy to construct the required automatic machine. We can suppose that the choices are always choices between two possibilities 0 and 1. Each proof will then be determined by a sequence of choices $i_1, i_2, \ldots, i_n$ ($i_1 = 0$ or 1, $i_2 = 0$ or $1, \ldots, i_n = 0$ or 1), and hence the number $2^n + i_1 2^{n-1} + i_2 2^{n-2} + \ldots + i_n$ completely determines the proof. The automatic machine carries out successively proof 1, proof 2, proof 3, ...

§ The author has found a description of such a machine.

‖ The negation sign is written before an expression and not over it.

¶ A sequence of r primes is denoted by $^{(r)}$.

I say that α is then a computable sequence: a machine $\mathcal{K}_\alpha$ to compute α can be obtained by a fairly simple modification of $\mathcal{K}$.

We divide the motion of $\mathcal{K}_\alpha$ into sections. The n-th section is devoted to finding the n-th figure of α. After the $(n-1)$-th section is finished a double colon :: is printed after all the symbols, and the succeeding work is done wholly on the squares to the right of this double colon. The first step is to write the letter "A" followed by the formula (A_n) and then "B" followed by (B_n). The machine $\mathcal{K}_\alpha$ then starts to do the work of $\mathcal{K}$, but whenever a provable formula is found, this formula is compared with (A_n) and with (B_n). If it is the same formula as (A_n), then the figure "1" is printed, and the n-th section is finished. If it is (B_n), then "0" is printed and the section is finished. If it is different from both, then the work of $\mathcal{K}$ is continued from the point at which it had been abandoned. Sooner or later one of the formulae (A_n) or (B_n) is reached; this follows from our hypotheses about α and $\mathfrak{A}$, and the known nature of $\mathcal{K}$. Hence the n-th section will eventually be finished. $\mathcal{K}_\alpha$ is circle-free; α is computable.

It can also be shown that the numbers α definable in this way by the use of axioms include all the computable numbers. This is done by describing computing machines in terms of the function calculus.

It must be remembered that we have attached rather a special meaning to the phrase "$\mathfrak{A}$ defines α". The computable numbers do not include all (in the ordinary sense) definable numbers. Let δ be a sequence whose n-th figure is 1 or 0 according as n is or is not satisfactory. It is an immediate consequence of the theorem of § 8 that δ is not computable. It is (so far as we know at present) possible that any assigned number of figures of δ can be calculated, but not by a uniform process. When sufficiently many figures of δ have been calculated, an essentially new method is necessary in order to obtain more figures.

III. This may be regarded as a modification of I or as a corollary of II.

We suppose, as in I, that the computation is carried out on a tape; but we avoid introducing the "state of mind" by considering a more physical and definite counterpart of it. It is always possible for the computer to break off from his work, to go away and forget all about it, and later to come back and go on with it. If he does this he must leave a note of instructions (written in some standard form) explaining how the work is to be continued. This note is the counterpart of the "state of mind". We will suppose that the computer works in such a desultory manner that he never does more than one step at a sitting. The note of instructions must enable him to carry out one step and write the next note. Thus the state of progress of the computation at any stage is completely determined by the note of instructions and the symbols on the tape. That is, the state of the system may be described by a single expression (sequence of symbols), consisting of the symbols on the tape followed by Δ (which we suppose not to appear elsewhere) and then by the note of instructions. This expression may be called the "state formula". We know that the state formula at any given stage is determined by the state formula before the last step was made, and we assume that the relation of these two formulae is expressible in the functional calculus. In other words, we assume that there is an axiom $\mathfrak{A}$ which expresses the rules governing the behaviour of the computer, in terms of the relation of the state formula at any stage to the state formula at the preceding stage. If this is so, we can construct a machine to write down the successive state formulae, and hence to compute the required number.

10. *Examples of large classes of numbers which are computable.*

It will be useful to begin with definitions of a computable function of an integral variable and of a computable variable, etc. There are many equivalent ways of defining a computable function of an integral variable. The simplest is, possibly, as follows. If γ is a computable sequence in which 0 appears infinitely† often, and n is an integer, then let us define $\xi(\gamma, n)$ to be the number of figures 1 between the n-th and the $(n+1)$-th figure 0 in γ. Then $\phi(n)$ is computable if, for all n and some γ, $\phi(n) = \xi(\gamma, n)$. An equivalent definition is this. Let $H(x, y)$ mean $\phi(x) = y$. Then, if we can find a contradiction-free axiom $\mathfrak{A}_\phi$, such that $\mathfrak{A}_\phi \rightarrow P$, and if for each integer n there exists an integer N, such that

$$\mathfrak{A}_\phi \ \& \ F^{(N)} \rightarrow H(u^{(n)}, u^{(\phi(n))}),$$

and such that, if $m \neq \phi(n)$, then, for some N',

$$\mathfrak{A}_\phi \ \& \ F^{(N')} \rightarrow \big(-H(u^{(n)}, u^{(m)}\big),$$

then ϕ may be said to be a computable function.

We cannot define general computable functions of a real variable, since there is no general method of describing a real number, but we can define a computable function of a computable variable. If n is satisfactory, let γ_n be the number computed by $\mathcal{M}(n)$, and let

$$a_n = \tan\big(\pi(\gamma_n - \tfrac{1}{2})\big),$$

† If $\mathcal{M}$ computes γ, then the problem whether $\mathcal{M}$ prints 0 infinitely often is of the same character as the problem whether $\mathcal{M}$ is circle-free.

unless $\gamma_n = 0$ or $\gamma_n = 1$, in either of which cases $\alpha_n = 0$. Then, as n runs through the satisfactory numbers, α_n runs through the computable numbers†. Now let $\phi(n)$ be a computable function which can be shown to be such that for any satisfactory argument its value is satisfactory‡. Then the function f, defined by $f(\alpha_n) = \alpha_{\phi(n)}$, is a computable function and all computable functions of a computable variable are expressible in this form.

Similar definitions may be given of computable functions of several variables, computable-valued functions of an integral variable, etc.

I shall enunciate a number of theorems about computability, but I shall prove only (ii) and a theorem similar to (iii).

(i) A computable function of a computable function of an integral or computable variable is computable.

(ii) Any function of an integral variable defined recursively in terms of computable functions is computable. *I.e.* if $\phi(m, n)$ is computable, and r is some integer, then $\eta(n)$ is computable, where

$$\eta(0) = r,$$

$$\eta(n) = \phi\big(n, \eta(n-1)\big).$$

(iii) If $\phi\ (m, n)$ is a computable function of two integral variables, then $\phi(n, n)$ is a computable function of n.

(iv) If $\phi(n)$ is a computable function whose value is always 0 or 1, then the sequence whose n-th figure is $\phi(n)$ is computable.

Dedekind's theorem does not hold in the ordinary form if we replace "real" throughout by "computable". But it holds in the following form:

(v) If $G(\alpha)$ is a propositional function of the computable numbers and

$$(a) \quad (\exists\alpha)(\exists\beta)\big\{G(\alpha)\ \&\ \big(-G(\beta)\big)\big\},$$

$$(b) \quad G(\alpha)\ \&\ \big(-G(\beta)\big) \rightarrow (\alpha < \beta),$$

and there is a general process for determining the truth value of $G(\alpha)$, then

† A function α_n may be defined in many other ways so as to run through the computable numbers.

‡ Although it is not possible to find a general process for determining whether a given number is satisfactory, it is often possible to show that certain classes of numbers are satisfactory.

there is a computable number ξ such that

$$G(a) \to a \leqslant \xi,$$
$$-G(a) \to a \geqslant \xi.$$

In other words, the theorem holds for any section of the computables such that there is a general process for determining to which class a given number belongs.

Owing to this restriction of Dedekind's theorem, we cannot say that a computable bounded increasing sequence of computable numbers has a computable limit. This may possibly be understood by considering a sequence such as

$$-1,\ -\tfrac{1}{2},\ -\tfrac{1}{4},\ -\tfrac{1}{8},\ -\tfrac{1}{16},\ \tfrac{1}{2},\ \ldots.$$

On the other hand, (v) enables us to prove

(vi) If a and β are computable and $a < \beta$ and $\phi(a) < 0 < \phi(\beta)$, where $\phi(a)$ is a computable increasing continuous function, then there is a unique computable number γ, satisfying $a < \gamma < \beta$ and $\phi(\gamma) = 0$.

Computable convergence.

We shall say that a sequence β_n of computable numbers *converges computably* if there is a computable integral valued function $N(\epsilon)$ of the computable variable ϵ, such that we can show that, if $\epsilon > 0$ and $n > N(\epsilon)$ and $m > N(\epsilon)$, then $|\beta_n - \beta_m| < \epsilon$.

We can then show that

(vii) A power series whose coefficients form a computable sequence of computable numbers is computably convergent at all computable points in the interior of its interval of convergence.

(viii) The limit of a computably convergent sequence is computable.

And with the obvious definition of "uniformly computably convergent":

(ix) The limit of a uniformly computably convergent computable sequence of computable functions is a computable function. Hence

(x) The sum of a power series whose coefficients form a computable sequence is a computable function in the interior of its interval of convergence.

From (viii) and $\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \ldots)$ we deduce that π is computable.

From $e = 1 + 1 + \dfrac{1}{2\,!} + \dfrac{1}{3\,!} + \ldots$ we deduce that e is computable.

From (vi) we deduce that all real algebraic numbers are computable.

From (vi) and (x) we deduce that the real zeros of the Bessel functions are computable.

Proof of (ii).

Let $H(x, y)$ mean "$\eta(x) = y$", and let $K(x, y, z)$ mean "$\phi(x, y) = z$". $\mathfrak{A}_\phi$ is the axiom for $\phi(x, y)$. We take $\mathfrak{A}_\eta$ to be

$$\mathfrak{A}_\phi \mathbin{\&} P \mathbin{\&} \big(F(x, y) \rightarrow G(x, y)\big) \mathbin{\&} \big(G(x, y) \mathbin{\&} G(y, z) \rightarrow G(x, z)\big)$$
$$\mathbin{\&} \big(F^{(r)} \rightarrow H(u, u^{(r)})\big) \mathbin{\&} \big(F(v, w) \mathbin{\&} H(v, x) \mathbin{\&} K(w, x, z) \rightarrow H(w, z)\big)$$
$$\mathbin{\&} \Big[H(w, z) \mathbin{\&} G(z, t) \vee G(t, z) \rightarrow \big(-H(w, t)\big)\Big].$$

I shall not give the proof of consistency of $\mathfrak{A}_\eta$. Such a proof may be constructed by the methods used in Hilbert and Bernays, *Grundlagen der Mathematik* (Berlin, 1934), p. 209 *et seq.* The consistency is also clear from the meaning.

Suppose that, for some n, N, we have shown

$$\mathfrak{A}_\eta \mathbin{\&} F^{(N)} \rightarrow H(u^{(n-1)}, u^{(\eta(n-1))}),$$

then, for some M,

$$\mathfrak{A}_\phi \mathbin{\&} F^{(M)} \rightarrow K(u^{(n)}, u^{(\eta(n-1))}, u^{(\eta(n))}),$$
$$\mathfrak{A}_\eta \mathbin{\&} F^{(M)} \rightarrow F(u^{(n-1)}, u^{(n)}) \mathbin{\&} H(u^{(n-1)}, u^{(\eta(n-1))})$$
$$\mathbin{\&} K(u^{(n)}, u^{(\eta(n-1))}, u^{(\eta(n))}),$$

and

$$\mathfrak{A}_\eta \mathbin{\&} F^{(M)} \rightarrow [F(u^{(n-1)}, u^{(n)}) \mathbin{\&} H(u^{(n-1)}, u^{(\eta(n-1))})$$
$$\mathbin{\&} K(u^{(n)}, u^{(\eta(n-1))}, u^{(\eta(n))}) \rightarrow H(u^{(n)}, u^{(\eta(n))})].$$

Hence $$\mathfrak{A}_\eta \mathbin{\&} F^{(M)} \rightarrow H(u^{(n)}, u^{(\eta(n))}).$$

Also $$\mathfrak{A}_\eta \mathbin{\&} F^{(r)} \rightarrow H(u, u^{(\eta(0))}).$$

Hence for each n some formula of the form

$$\mathfrak{A}_\eta \mathbin{\&} F^{(M)} \rightarrow H(u^{(n)}, u^{(\eta(n))})$$

is provable. Also, if $M' \geqslant M$ and $M' \geqslant m$ and $m \neq \eta(u)$, then

$$\mathfrak{A}_\eta \mathbin{\&} F^{(M')} \rightarrow G(u^{\eta((n))}, u^{(m)}) \vee G(u^{(m)}, u^{(\eta(n))})$$

and

$$\mathfrak{A}_\eta \,\&\, F^{(M')} \rightarrow \Big[\{ G(u^{(\eta(n))}, u^{(m)}) \vee G(u^{(m)}, u^{(\eta(n))}) \;\&\; H(u^{(n)}, u^{(\eta(n))}) \} \rightarrow \big(-H(u^{(n)}, u^{(m)}) \big) \Big].$$

Hence $$\mathfrak{A}_\eta \,\&\, F^{(M')} \rightarrow \big(-H(u^{(n)}, u^{(m)}) \big).$$

The conditions of our second definition of a computable function are therefore satisfied. Consequently η is a computable function.

Proof of a modified form of (iii).

Suppose that we are given a machine $\mathcal{N}$, which, starting with a tape bearing on it əə followed by a sequence of any number of letters "F" on F-squares and in the m-configuration $\mathfrak{b}$, will compute a sequence γ_n depending on the number n of letters "F". If $\phi_n(m)$ is the m-th figure of γ_n, then the sequence β whose n-th figure is $\phi_n(n)$ is computable.

We suppose that the table for $\mathcal{N}$ has been written out in such a way that in each line only one operation appears in the operations column. We also suppose that Ξ, Θ, $\bar{0}$, and $\bar{1}$ do not occur in the table, and we replace ə throughout by Θ, 0 by $\bar{0}$, and 1 by $\bar{1}$. Further substitutions are then made. Any line of form

$\mathfrak{A}$	α	$P\bar{0}$	$\mathfrak{B}$

we replace by

$\mathfrak{A}$	α	$P\bar{0}$	$\mathfrak{re}(\mathfrak{B}, \mathfrak{u}, h, k)$

and any line of the form

$\mathfrak{A}$	α	$P\bar{1}$	$\mathfrak{B}$

by

$\mathfrak{A}$	α	$P\bar{1}$	$\mathfrak{re}(\mathfrak{B}, \mathfrak{v}, h, k)$

and we add to the table the following lines:

$\mathfrak{u}$		$\mathfrak{pe}(\mathfrak{u}_1, 0)$
$\mathfrak{u}_1$	$R, Pk, R, P\Theta, R, P\Theta$	$\mathfrak{u}_2$
$\mathfrak{u}_2$		$\mathfrak{re}(\mathfrak{u}_3, \mathfrak{u}_3, k, h)$
$\mathfrak{u}_3$		$\mathfrak{pe}(\mathfrak{u}_2, F)$

and similar lines with $\mathfrak{v}$ for $\mathfrak{u}$ and 1 for 0 together with the following line

$\mathfrak{c}$	$R, P\Xi, R, Ph$	$\mathfrak{b}$.

We then have the table for the machine $\mathcal{N}'$ which computes β. The initial m-configuration is $\mathfrak{c}$, and the initial scanned symbol is the second ə.

11. *Application to the Entscheidungsproblem.*

The results of § 8 have some important applications. In particular, they can be used to show that the Hilbert Entscheidungsproblem can have no solution. For the present I shall confine myself to proving this particular theorem. For the formulation of this problem I must refer the reader to Hilbert and Ackermann's *Grundzüge der Theoretischen Logik* (Berlin, 1931), chapter 3.

I propose, therefore, to show that there can be no general process for determining whether a given formula $\mathfrak{A}$ of the functional calculus **K** is provable, *i.e.* that there can be no machine which, supplied with any one $\mathfrak{A}$ of these formulae, will eventually say whether $\mathfrak{A}$ is provable.

It should perhaps be remarked that what I shall prove is quite different from the well-known results of Gödel†. Gödel has shown that (in the formalism of Principia Mathematica) there are propositions $\mathfrak{A}$ such that neither $\mathfrak{A}$ nor $-\mathfrak{A}$ is provable. As a consequence of this, it is shown that no proof of consistency of Principia Mathematica (or of **K**) can be given within that formalism. On the other hand, I shall show that there is no general method which tells whether a given formula $\mathfrak{A}$ is provable in **K**, or, what comes to the same, whether the system consisting of **K** with $-\mathfrak{A}$ adjoined as an extra axiom is consistent.

If the negation of what Gödel has shown had been proved, *i.e.* if, for each $\mathfrak{A}$, either $\mathfrak{A}$ or $-\mathfrak{A}$ is provable, then we should have an immediate solution of the Entscheidungsproblem. For we can invent a machine $\mathcal{K}$ which will prove consecutively all provable formulae. Sooner or later $\mathcal{K}$ will reach either $\mathfrak{A}$ or $-\mathfrak{A}$. If it reaches $\mathfrak{A}$, then we know that $\mathfrak{A}$ is provable. If it reaches $-\mathfrak{A}$, then, since **K** is consistent (Hilbert and Ackermann, p. 65), we know that $\mathfrak{A}$ is not provable.

Owing to the absence of integers in **K** the proofs appear somewhat lengthy. The underlying ideas are quite straightforward.

Corresponding to each computing machine $\mathcal{M}$ we construct a formula Un $(\mathcal{M})$ and we show that, if there is a general method for determining whether Un $(\mathcal{M})$ is provable, then there is a general method for determining whether $\mathcal{M}$ ever prints 0.

The interpretations of the propositional functions involved are as follows :

$R_{S_l}(x, y)$ is to be interpreted as "in the complete configuration x (of $\mathcal{M}$) the symbol on the square y is S".

† *Loc. cit.*

$I(x, y)$ is to be interpreted as "in the complete configuration x the square y is scanned".

$K_{q_m}(x)$ is to be interpreted as "in the complete configuration x the m-configuration is q_m.

$F(x, y)$ is to be interpreted as "y is the immediate successor of x".

Inst $\{q_i S_j S_k L q_l\}$ is to be an abbreviation for

$$(x, y, x', y') \Big\{ \big(R_{S_j}(x, y) \mathbin{\&} I(x, y) \mathbin{\&} K_{q_i}(x) \mathbin{\&} F(x, x') \mathbin{\&} F(y', y) \big)$$
$$\rightarrow \big(I(x', y') \mathbin{\&} R_{S_k}(x', y) \mathbin{\&} K_{q_l}(x')$$
$$\mathbin{\&} (z) \big[F(y', z) \vee \big(R_{S_j}(x, z) \rightarrow R_{S_k}(x', z) \big) \big] \big) \Big\}.$$

Inst $\{q_i S_j S_k R q_l\}$ and Inst $\{q_i S_j S_k N q_l\}$

are to be abbreviations for other similarly constructed expressions.

Let us put the description of $\mathcal{M}$ into the first standard form of § 6. This description consists of a number of expressions such as "$q_i\, S_j\, S_k\, L q_l$" (or with R or N substituted for L). Let us form all the corresponding expressions such as Inst $\{q_i S_j S_k L q_l\}$ and take their logical sum. This we call Des $(\mathcal{M})$.

The formula Un $(\mathcal{M})$ is to be

$$(\exists u) \big[N(u) \mathbin{\&} (x) \big(N(x) \rightarrow (\exists x') F(x, x') \big)$$
$$\mathbin{\&} (y, z) \big(F(y, z) \rightarrow N(y) \mathbin{\&} N(z) \big) \mathbin{\&} (y) R_{S_0}(u, y)$$
$$\mathbin{\&} I(u, u) \mathbin{\&} K_{q_1}(u) \mathbin{\&} \mathrm{Des}(\mathcal{M}) \big]$$
$$\rightarrow (\exists s)(\exists t) [N(s) \mathbin{\&} N(t) \mathbin{\&} R_{S_1}(s, t)].$$

$[N(u) \mathbin{\&} \ldots \mathbin{\&} \mathrm{Des}(\mathcal{M})]$ may be abbreviated to $A(\mathcal{M})$.

When we substitute the meanings suggested on p. 259–60 we find that Un $(\mathcal{M})$ has the interpretation "in some complete configuration of $\mathcal{M}$, S_1 (*i.e.* 0) appears on the tape". Corresponding to this I prove that

(*a*) If S_1 appears on the tape in some complete configuration of $\mathcal{M}$, then Un $(\mathcal{M})$ is provable.

(*b*) If Un $(\mathcal{M})$ is provable, then S_1 appears on the tape in some complete configuration of $\mathcal{M}$.

When this has been done, the remainder of the theorem is trivial.

LEMMA 1. *If S_1 appears on the tape in some complete configuration of $\mathcal{M}$, then* Un($\mathcal{M}$) *is provable.*

We have to show how to prove Un ($\mathcal{M}$). Let us suppose that in the n-th complete configuration the sequence of symbols on the tape is $S_{r(n,0)}, S_{r(n,1)}, \ldots, S_{r(n,n)}$, followed by nothing but blanks, and that the scanned symbol is the $i(n)$-th, and that the m-configuration is $q_{k(n)}$. Then we may form the proposition

$$\begin{aligned} &R_{S_{r(n,0)}}(u^{(n)}, u) \;\&\; R_{S_{r(n,1)}}(u^{(n)}, u') \;\&\; \ldots \;\&\; R_{S_{r(n,n)}}(u^{(n)}, u^{(n)}) \\ &\qquad \& \; I(u^{(n)}, u^{(i(n))}) \;\&\; K_{q_{k(n)}}(u^{(n)}) \\ &\qquad \& \; (y) F\Big((y, u') \vee F(u, y) \vee F(u', y) \vee \ldots \vee F(u^{(n-1)}, y) \vee R_{S_0}(u^{(n)}, y)\Big), \end{aligned}$$

which we may abbreviate to CC_n.

As before, $F(u, u') \;\&\; F(u', u'') \;\&\; \ldots \;\&\; F(u^{(r-1)}, u^{(r)})$ is abbreviated to $F^{(r)}$.

I shall show that all formulae of the form $A(\mathcal{M}) \;\&\; F^{(n)} \to CC_n$ (abbreviated to CF_n) are provable. The meaning of CF_n is "The n-th complete configuration of $\mathcal{M}$ is so and so", where "so and so" stands for the actual n-th complete configuration of $\mathcal{M}$. That CF_n should be provable is therefore to be expected.

CF_0 is certainly provable, for in the complete configuration the symbols are all blanks, the m-configuration is q_1, and the scanned square is u, *i.e.* CC_0 is

$$(y) R_{S_0}(u, y) \;\&\; I(u, u) \;\&\; K_{q_1}(u).$$

$A(\mathcal{M}) \to CC_0$ is then trivial.

We next show that $CF_n \to CF_{n+1}$ is provable for each n. There are three cases to consider, according as in the move from the n-th to the $(n+1)$-th configuration the machine moves to left or to right or remains stationary. We suppose that the first case applies, *i.e.* the machine moves to the left. A similar argument applies in the other cases. If $r\big(n, i(n)\big) = a$, $r\big(n+1, i(n+1)\big) = c$, $k\big(i(n)\big) = b$, and $k\big(i(n+1)\big) = d$, then Des ($\mathcal{M}$) must include Inst $\{q_a S_b S_d L q_c\}$ as one of its terms, *i.e.*

$$\text{Des}(\mathcal{M}) \to \text{Inst}\{q_a S_b S_d L q_c\}.$$

Hence $\qquad A(\mathcal{M}) \;\&\; F^{(n+1)} \to \text{Inst}\{q_a S_b S_d L q_c\} \;\&\; F^{(n+1)}$.

But $\qquad \text{Inst}\{q_a S_b S_d L q_c\} \;\&\; F^{(n+1)} \to (CC_n \to CC_{n+1})$

is provable, and so therefore is

$$A(\mathcal{M}) \;\&\; F^{(n+1)} \to (CC_n \to CC_{n+1})$$

and $$\left(A(\mathcal{M}) \;\&\; F^{(n)} \rightarrow CC_n\right) \rightarrow \left(A(\mathcal{M}) \;\&\; F^{(n+1)} \rightarrow CC_{n+1}\right),$$

i.e. $$CF_n \rightarrow CF_{n+1}.$$

CF_n is provable for each n. Now it is the assumption of this lemma that S_1 appears somewhere, in some complete configuration, in the sequence of symbols printed by $\mathcal{M}$; that is, for some integers N, K, CC_N has $R_{S_1}(u^{(N)}, u^{(K)})$ as one of its terms, and therefore $CC_N \rightarrow R_{S_1}(u^{(N)}, u^{(K)})$ is provable. We have then

$$CC_N \rightarrow R_{S_1}(u^{(N)}, u^{(K)})$$

and $$A(\mathcal{M}) \;\&\; F^{(N)} \rightarrow CC^N.$$

We also have

$$(\exists u) A(\mathcal{M}) \rightarrow (\exists u)(\exists u') \dots (\exists u^{(N')})\left(A(\mathcal{M}) \;\&\; F^{(N)}\right),$$

where $N' = \max(N, K)$. And so

$$(\exists u) A(\mathcal{M}) \rightarrow (\exists u)(\exists u') \dots (\exists u^{(N')}) R_{S_1}(u^{(N)}, u^{(K)}),$$

$$(\exists u) A(\mathcal{M}) \rightarrow (\exists u^{(N)})(\exists u^{(K)}) R_{S_1}(u^{(N)}, u^{(K)}),$$

$$(\exists u) A(\mathcal{M}) \rightarrow (\exists s)(\exists t) R_{S_1}(s, t),$$

i.e. $\mathrm{Un}(\mathcal{M})$ is provable.

This completes the proof of Lemma 1.

Lemma 2. *If* $\mathrm{Un}(\mathcal{M})$ *is provable, then* S_1 *appears on the tape in some complete configuration of* $\mathcal{M}$.

If we substitute any propositional functions for function variables in a provable formula, we obtain a true proposition. In particular, if we substitute the meanings tabulated on pp. 259–260 in $\mathrm{Un}(\mathcal{M})$, we obtain a true proposition with the meaning "S_1 appears somewhere on the tape in some complete configuration of $\mathcal{M}$".

We are now in a position to show that the Entscheidungsproblem cannot be solved. Let us suppose the contrary. Then there is a general (mechanical) process for determining whether $\mathrm{Un}(\mathcal{M})$ is provable. By Lemmas 1 and 2, this implies that there is a process for determining whether $\mathcal{M}$ ever prints 0, and this is impossible, by § 8. Hence the Entscheidungsproblem cannot be solved.

In view of the large number of particular cases of solutions of the Entscheidungsproblem for formulae with restricted systems of quantors, it is interesting to express Un($\mathcal{M}$) in a form in which all quantors are at the beginning. Un($\mathcal{M}$) is, in fact, expressible in the form

$$(u)(\exists x)(w)(\exists u_1)\ldots(\exists u_n)\mathfrak{B}, \tag{I}$$

where $\mathfrak{B}$ contains no quantors, and $n = 6$. By unimportant modifications we can obtain a formula, with all essential properties of Un($\mathcal{M}$), which is of form (I) with $n = 5$.

Added 28 *August*, 1936.

APPENDIX.

Computability and effective calculability

The theorem that all effectively calculable (λ-definable) sequences are computable and its converse are proved below in outline. It is assumed that the terms "well-formed formula" (W.F.F.) and "conversion" as used by Church and Kleene are understood. In the second of these proofs the existence of several formulae is assumed without proof; these formulae may be constructed straightforwardly with the help of, *e.g.*, the results of Kleene in "A theory of positive integers in formal logic", *American Journal of Math.*, 57 (1935), 153-173, 219-244.

The W.F.F. representing an integer n will be denoted by N_n. We shall say that a sequence γ whose n-th figure is $\phi_\gamma(n)$ is λ-definable or effectively calculable if $1+\phi_\gamma(n)$ is a λ-definable function of n, *i.e.* if there is a W.F.F. M_γ such that, for all integers n,

$$\{M_\gamma\}(N_n) \text{ conv } N_{\phi_\gamma(n)+1},$$

i.e. $\{M_\gamma\}(N_n)$ is convertible into $\lambda xy.x\big(x(y)\big)$ or into $\lambda xy.x(y)$ according as the n-th figure of λ is 1 or 0.

To show that every λ-definable sequence γ is computable, we have to show how to construct a machine to compute γ. For use with machines it is convenient to make a trivial modification in the calculus of conversion. This alteration consists in using $x, x', x'', \ldots$ as variables instead of $a, b, c, \ldots$. We now construct a machine $\mathcal{L}$ which, when supplied with the formula M_γ, writes down the sequence γ. The construction of $\mathcal{L}$ is somewhat similar to that of the machine $\mathcal{K}$ which proves all provable formulae of the functional calculus. We first construct a choice machine $\mathcal{L}_1$, which, if supplied with a W.F.F., M say, and suitably manipulated, obtains any formula into which M is convertible. $\mathcal{L}_1$ can then be modified so as to yield an automatic machine $\mathcal{L}_2$ which obtains successively all the formulae into which M is convertible (cf. foot-note p. 252). The machine $\mathcal{L}$ includes $\mathcal{L}_2$ as a part. The motion of the machine $\mathcal{L}$ when supplied with the formula M_γ is divided into sections of which the n-th is devoted to finding the n-th figure of γ. The first stage in this n-th section is the formation of $\{M_\gamma\}(N_n)$. This formula is then supplied to the machine $\mathcal{L}_2$, which converts it successively into various other formulae. Each formula into which it is convertible eventually appears, and each, as it is found, is compared with

$$\lambda x\Big[\lambda x'\Big[\{x\}\big(\{x\}(x')\big)\Big]\Big], \quad i.e.\ N_2,$$

and with
$$\lambda x\Big[\lambda x'[\{x\}(x')]\Big], \quad i.e.\ N_1.$$

If it is identical with the first of these, then the machine prints the figure 1 and the n-th section is finished. If it is identical with the second, then 0 is printed and the section is finished. If it is different from both, then the work of $\mathcal{L}_2$ is resumed. By hypothesis, $\{M_\gamma\}(N_n)$ is convertible into one of the formulae N_2 or N_1; consequently the n-th section will eventually be finished, *i.e.* the n-th figure of γ will eventually be written down.

To prove that every computable sequence γ is λ-definable, we must show how to find a formula M_γ such that, for all integers n,

$$\{M_\gamma\}(N_n) \text{ conv } N_{1+\phi_\gamma(n)}.$$

Let $\mathcal{M}$ be a machine which computes γ and let us take some description of the complete configurations of $\mathcal{M}$ by means of numbers, *e.g.* we may take the D.N of the complete configuration as described in §6. Let $\xi(n)$ be the D.N of the n-th complete configuration of $\mathcal{M}$. The table for the machine $\mathcal{M}$ gives us a relation between $\xi(n+1)$ and $\xi(n)$ of the form

$$\xi(n+1) = \rho_\gamma\big(\xi(n)\big),$$

where ρ_γ is a function of very restricted, although not usually very simple, form: it is determined by the table for $\mathcal{M}$. ρ_γ is λ-definable (I omit the proof of this), *i.e.* there is a W.F.F. A_γ such that, for all integers n,

$$\{A_\gamma\}(N_{\xi(n)}) \text{ conv } N_{\xi(n+1)}.$$

Let U stand for
$$\lambda u\Big[\big\{\{u\}(A_\gamma)\big\}(N_r)\Big],$$

where $r = \xi(0)$; then, for all integers n,

$$\{U_\gamma\}(N_n) \text{ conv } N_{\xi(n)}.$$

It may be proved that there is a formula V such that

$$\Big\{\{V\}(N_{\xi(n+1)})\Big\}(N_{\xi(n)}) \begin{cases} \text{conv } N_1 & \text{if, in going from the } n\text{-th to the } (n+1)\text{-th complete configuration, the figure 0 is printed.} \\ \text{conv } N_2 & \text{if the figure 1 is printed.} \\ \text{conv } N_3 & \text{otherwise.} \end{cases}$$

Let W_γ stand for

$$\lambda u\Big[\Big\{\{V\}\Big(\{A_\gamma\}\big(\{U_\gamma\}(u)\big)\Big)\Big\}\Big(\{U_\gamma\}(u)\Big)\Big],$$

so that, for each integer n,

$$\Big\{\{V\}(N_{\xi(n+1)})\Big\}(N_{\xi(n)}) \text{ conv } \{W_\gamma\}(N_n),$$

and let Q be a formula such that

$$\Big\{\{Q\}(W_\gamma)\Big\}(N_s) \text{ conv } N_{r(s)},$$

where $r(s)$ is the s-th integer q for which $\{W_\gamma\}(N_q)$ is convertible into either N_1 or N_2. Then, if M_γ stands for

$$\lambda w\Big[\{W_\gamma\}\Big(\Big\{\{Q\}(W_\gamma)\Big\}(w)\Big)\Big],$$

it will have the required property†.

The Graduate College,
Princeton University,
New Jersey, U.S.A.

† In a complete proof of the λ-definability of computable sequences it would be best to modify this method by replacing the numerical description of the complete configurations by a description which can be handled more easily with our apparatus. Let us choose certain integers to represent the symbols and the m-configurations of the machine. Suppose that in a certain complete configuration the numbers representing the successive symbols on the tape are $s_1 s_2 \ldots s_n$, that the m-th symbol is scanned, and that the m-configuration has the number t; then we may represent this complete configuration by the formula

$$\Big[[N_{s_1}, N_{s_2}, \ldots, N_{s_{m-1}}],\ [N_t, N_{s_m}],\ [N_{s_{m+1}}, \ldots, N_{s_n}]\Big],$$

where $[a, b]$ stands for $\lambda u\Big[\Big\{\{u\}(a)\Big\}(b)\Big]$,

$[a, b, c]$ stands for $\lambda u\Big[\Big\{\Big\{\{u\}(a)\Big\}(b)\Big\}(c)\Big]$,

etc.

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM. A CORRECTION

By A. M. TURING.

In a paper entitled "On computable numbers, with an application to the Entscheidungsproblem"* the author gave a proof of the insolubility of the Entscheidungsproblem of the "engere Funktionenkalkül". This proof contained some formal errors† which will be corrected here: there are also some other statements in the same paper which should be modified, although they are not actually false as they stand.

The expression for Inst $\{q_i S_j S_k L q_l\}$ on p. 260 of the paper quoted should read

$$(x, y, x', y')\Big\{\Big(R_{S_j}(x, y) \,\&\, I(x, y) \,\&\, K_{q_i}(x) \,\&\, F(x, x') \,\&\, F(y', y)\Big)$$

$$\rightarrow \Big(I(x', y') \,\&\, R_{S_k}(x', y) \,\&\, K_{q_l}(x') \,\&\, F(y', z) \vee \Big[\Big(R_{S_0}(x, z) \rightarrow R_{S_0}(x', z)\Big)$$

$$\& \Big(R_{S_1}(x, z) \rightarrow R_{S_1}(x', z)\Big) \,\&\, \ldots \,\&\, \Big(R_{S_M}(x, z) \rightarrow R_{S_M}(x', z)\Big)\Big]\Big)\Big\},$$

$S_0, S_1, \ldots, S_M$ being the symbols which $\mathcal{M}$ can print. The statement on p. 261, line 33, viz.

$$\text{“Inst}\,\{q_a S_b S_d L q_c\} \,\&\, F^{(n+1)} \rightarrow (CC_n \rightarrow CC_{n+1})$$

is provable" is false (even with the new expression for Inst $\{q_a S_b S_d L q_c\}$): we are unable for example to deduce $F^{(n+1)} \rightarrow \Big(-F(u, u'')\Big)$ and therefore can never use the term

$$F(y', z) \vee \Big[\Big(R_{S_0}(x, z) \rightarrow R_{S_0}(x', z)\Big) \,\&\, \ldots \,\&\, \Big(R_{S_M}(x, z) \rightarrow R_{S_M}(x', z)\Big)\Big]$$

* *Proc. London Math. Soc.* (2), 42 (1936–7), 230–265.

† The author is indebted to P. Bernays for pointing out these errors.

in Inst $\{q_a S_b S_d L q_c\}$. To correct this we introduce a new functional variable G [$G(x, y)$ to have the interpretation "x precedes y"]. Then, if Q is an abbreviation for

$$(x)(\exists w)(y, z)\Big\{F(x, w) \,\&\, \big(F(x, y) \rightarrow G(x, y)\big) \,\&\, \big(F(x, z) \,\&\, G(z, y) \rightarrow G(x, y)\big)$$
$$\&\Big[G(z, x) \vee \big(G(x, y) \,\&\, F(y, z)\big) \vee \big(F(x, y) \,\&\, F(z, y)\big) \rightarrow \big(-F(x, z)\big)\Big]\Big\}$$

the corrected formula $\mathrm{Un}(\mathfrak{M})$ is to be

$$(\exists u)\, A(\mathfrak{M}) \rightarrow (\exists s)(\exists t)\, R_{S_1}(s, t),$$

where $A(\mathfrak{M})$ is an abbreviation for

$$Q \,\&\, (y)\, R_{S_0}(u, y) \,\&\, I(u, u) \,\&\, K_{q_1}(u) \,\&\, \mathrm{Des}\,(\mathfrak{M}).$$

The statement on page 261 (line 33) must then read

$$\mathrm{Inst}\,\{q_a S_b S_d L q_c\} \,\&\, Q \,\&\, F^{(n+1)} \rightarrow (CC_n \rightarrow CC_{n+1}),$$

and line 29 should read

$$r\big(n, i(n)\big) = b, \quad r\big(n+1,\, i(n)\big) = d, \quad k(n) = a, \quad k(n+1) = c.$$

For the words "logical sum" on p. 260, line 15, read "conjunction With these modifications the proof is correct. $\mathrm{Un}(\mathfrak{M})$ may be put in the form (I) (p. 263) with $n = 4$.

Some difficulty arises from the particular manner in which "computable number" was defined (p. 233). If the computable numbers are to satisfy intuitive requirements we should have:

If we can give a rule which associates with each positive integer n two rationals a_n, b_n satisfying $a_n \leqslant a_{n+1} < b_{n+1} \leqslant b_n$, $b_n - a_n < 2^{-n}$, then there is a computable number α for which $a_n \leqslant \alpha \leqslant b_n$ each n. (A)

A proof of this may be given, valid by ordinary mathematical standards, but involving an application of the principle of excluded middle. On the other hand the following is false:

There is a rule whereby, given the rule of formation of the sequences a_n, b_n in (A) *we can obtain a* D.N. *for a machine to compute α.* (B)

That (B) is false, at least if we adopt the convention that the decimals of numbers of the form $m/2^n$ shall always terminate with zeros, can be seen in this way. Let $\mathfrak{N}$ be some machine, and define c_n as follows: $c_n = \frac{1}{2}$ if $\mathfrak{N}$ has not printed a figure 0 by the time the n-th complete configuration is reached $c_n = \frac{1}{2} - 2^{-m-3}$ if 0 had first been printed at the m-th complete configuration $(m \leqslant n)$. Put $a_n = c_n - 2^{-n-2}$, $b_n = c_n + 2^{-n-2}$. Then the inequalities of (A) are satisfied, and the first figure of α is 0 if $\mathfrak{N}$ ever prints 0 and is 1 otherwise. If (B) were true we should have a means of finding the first figure of α given the D.N. of $\mathfrak{N}$: *i.e.* we should be able to determine whether $\mathfrak{N}$ ever prints 0, contrary to the results of § 8 of the paper quoted. Thus although (A) shows that there must be machines which compute the Euler constant (for example) we cannot at present describe any such machine, for we do not yet know whether the Euler constant is of the form $m/2^n$.

This disagreeable situation can be avoided by modifying the manner in which computable numbers are associated with computable sequences, the totality of computable numbers being left unaltered. It may be done in many ways* of which this is an example. Suppose that the first figure of a computable sequence γ is i and that this is followed by 1 repeated n times, then by 0 and finally by the sequence whose r-th figure is c_r; then the sequence γ is to correspond to the real number

$$(2i-1)n + \sum_{r=1}^{\infty} (2c_r - 1)(\tfrac{2}{3})^r.$$

If the machine which computes γ is regarded as computing also this real number then (B) holds. The uniqueness of representation of real numbers by sequences of figures is now lost, but this is of little theoretical importance, since the D.N.'s are not unique in any case.

The Graduate College,
Princeton, N.J., U.S.A.

* This use of overlapping intervals for the definition of real numbers is due originally to Brouwer.