

Logic, History of:  
Modern Logic:  
Since Gödel:  
Turing and Computability Theory

In the late 1930s Alan M. Turing was one of the founders of computability theory. His main contributions to this field were published in three papers that appeared in the span of a few years, and especially in his ground-breaking 1936–1937 paper, published when he was twenty-four years old.

As indicated by its title, “On Computable Numbers, with an Application to the Entscheidungsproblem,” Turing’s paper deals ostensibly with real numbers that are computable in the sense that their decimal expansion “can be written down by a machine.” As he pointed out, however, the ideas carry over easily to computable functions on the integers or to computable predicates.

The paper was based on work that Turing had carried out as a Cambridge graduate student, under the direction of Maxwell Newman (1897–1984). When Turing first saw a 1936 paper by Alonzo Church, he realized at once that the two of them were tackling the same problem—making computability precise—albeit from different points of view. Turing wrote to Church and then traveled to Princeton University to meet with him. The final form of the paper was submitted from Princeton.

In a space of thirty-six pages, the paper manages to accomplish the following six goals:

1. A formalization of what it means to be “calculable by finite means,” in terms of an idealized computing device—now known of course as a Turing machine.
2. The construction of a “universal computing machine,” which when supplied with the “standard description” of a machine  $M$  on its tape, will simulate the operation on  $M$ .
3. The proof of the unsolvability of the halting problem and proofs of the unsolvability of other problems, such as the problem of deciding, given a machine  $M$ , whether or not  $M$  will ever print the symbol 0.
4. Three kinds of arguments for “Turing’s thesis,” that is, the claim that his formulation in terms of machines is successful in capturing the idea

of “processes which can be carried out in computing” (p. 249). It should be noted that Kurt Gödel and others have found Turing’s arguments here completely convincing.

5. A proof of Church’s theorem that David Hilbert’s Entscheidungsproblem can have no solution, that is, the problem of deciding whether or not a given formula is derivable in the predicate calculus is unsolvable.
6. An outline, in an appendix, of the equivalence of computability by Turing machines to computability as formulated by Church in terms of the  $\lambda$ -calculus. (This proof was given in further detail in Turing’s 1937 paper, “Computability and  $\lambda$ -Definability.”)

Turing’s paper remains a readable introduction to his ideas. How might a diligent clerk carry out a calculation, following instructions? The clerk might organize the work in a notebook. At any given moment his or her attention is focused on a particular page. Following the instructions, he or she might alter that page, and then might turn to another page. And the notebook is large enough that he or she never comes to the last page.

The alphabet of symbols available to the clerk must be finite; if there were infinitely many symbols, then there would be two that were arbitrarily similar and so might be confused. One can then, without loss of generality, regard what can be written on one page of notebook as a single symbol. And one can envision the notebook pages as being placed side by side, forming a paper tape, consisting of squares, each square being either blank or printed with a symbol. At each stage of the work, the clerk—or the mechanical machine—can alter the square under examination, can turn attention to the next square or the previous one, and can look to the instructions to see what part of them to follow next. Turing describes the latter as a change of state of mind.

Turing wrote, “We may now construct a machine to do the work” (p. 251). Such a machine is of course now called a Turing machine, a phrase first used by Church in his review of Turing’s paper in *The Journal of Symbolic Logic*. The machine has a potentially infinite tape, marked into squares. Initially, the tape is blank. (Alternatively, if one wants to compute some function, the input word or number can be written on the tape.) The machine is capable of being in any one of finitely many states (the phrase *of mind* being inappropriate for a machine).

At each step of the calculation, depending on its state at the time, the machine can change the symbol in the square under examination at that time, can turn its attention to the square to the left or to the right, and can then change its state to another state.

The program for this Turing machine can be given by a table. Where the possible states of the machine are  $q_1, \dots, q_r$ , each line of the table is a quintuple  $\langle q_i, S_j, S_k, D, q_m \rangle$  that is to be interpreted as directing that whenever the machine is in state  $q_i$  and the square under examination contains the symbol  $S_j$ , then that symbol should be altered to  $S_k$  and the machine should shift its attention to the square on the left (if  $D = L$ ) or on the right (if  $D = R$ ), and

should change its state to  $q_m$ . For the program to be unambiguous, it should have no two different quintuples with the same first two components. One of the states, say  $q_1$ , is designated as the initial state—the state in which the machine begins its calculation. If one starts the machine running in this state, it might (or might not), after some number of steps, reach a state and a symbol for which its table lacks a quintuple having that state and symbol for its first two components. At that point the machine halts.

In particular, where 0 and 1 are among the symbols in the alphabet, the machine might run on and on, sometimes printing a 0 or 1 on the tape (besides printing whatever other markers are needed for the computation). In this way, the machine might generate an infinite binary string. One can interpret this binary string as giving the binary expansion of a real number in the unit interval. Say that a real number is computable if it differs by an integer from a number in the unit interval whose binary expansion can be generated by some Turing machine.

Alternatively, if one wants the machine to compute a function, one can, after starting the machine with the input word or number on the tape, wait for it to halt and then look at the tape (starting with the square then under examination) to see what the output word or number is.

Turing shows how to construct a “universal computing machine” that, when supplied with the “standard description” of a machine  $M$  on its tape, will simulate the operation on  $M$ . This allows him to apply a diagonal argument to show that there can be no computable way to determine whether or not a given machine will continue to print 0’s and 1’s forever. In effect, he shows the unsolvability of the halting problem.

Turing argues that his formulation of the computability concept includes all sequences that would informally be considered to be computable. That is, he argues for what is now called Turing’s thesis, the Church–Turing thesis, or Church’s thesis, depending on the context. He gives three kinds of arguments: First, he shows how his machines can capture the informal idea of a step-by-step process, as indicated briefly earlier. Secondly, he shows that certain changes to his definition of a machine would have no effect at all on what sequences would be computable. And thirdly, he gives examples of large classes of numbers that are computable: the real algebraic numbers,  $e$ ,  $\pi$ , the real zeros of the Bessel functions, and so forth. Of course, as he emphasizes, only countably many real numbers can be computable.

Turing’s 1939 paper, “Systems of logic based on ordinals,” is based on his PhD dissertation, written under Church’s supervision during Turing’s two-year stay at Princeton.

Gödel’s incompleteness theorems had shown that any sufficiently strong formal system was incomplete, and in particular could not prove its own consistency. One can then add to this formal system the sentence expressing its consistency, thereby obtaining a stronger system. And this process can be iterated. The iteration can be transfinite, making use of ordinal notations for the constructive ordinals. This topic, which was later taken up by Solomon Feferman (1928–) in the 1950s, does not directly pertain to computability theory.

In the process, however, Turing introduced the important concept of computability relative to an oracle. He gave the basic definitions, and indicated how his work on computability could be adapted to incorporate the idea of calculations that, at any stage, could utilize a hypothetical fixed body of information. This idea later led to work on the classification (of problems or of sets or of functions) according to degree of unsolvability. Moreover, the degrees of unsolvability are partially ordered, under what is now called Turing reducibility.

After 1939, Turing's work on computability stopped while Turing, now back in England, threw himself into wartime cryptographic work. There was an urgent need to break the German battlefield Enigma code. The success of Turing and the British cryptographic team was of enormous military importance throughout World War II. But nothing was known publicly about this work until it was declassified, several years after Turing's suicide in 1954.

After the war, Turing turned to computation topics, both practical and theoretical, outside the field of computability theory. On the practical side, he was involved in hardware and software design for early digital computers. On the theoretical side, he published important work on artificial intelligence.

*See also* Computability Theory.

### Bibliography

- Alonzo Church. "An Unsolvable Problem of Elementary Number Theory." *American Journal of Mathematics* 58 (1936): 345–363.
- A. M. Turing. "On Computable Numbers, with an Application to the Entscheidungsproblem." *Proceedings of the London Mathematical Society* 42 (1936–1937): 230–265. A correction, 43 (1937): 544–546.
- A. M. Turing. "Computability and  $\lambda$ -Definability." *The Journal of Symbolic Logic* 2 (1937): 153–163.
- A. M. Turing. "Systems of Logic based on Ordinals." *Proceedings of the London Mathematical Society* 45 (3) (1939): 161–228.

Herbert B. Enderton