Rolf Herken (ed.)

The Universal Turing Machine
A Half-Century Survey

Second Edition

1995

SPRINGER-VERLAG
Wien New York

Kreisel, G.

1958 Ordinal logics and the characterization of informal concepts of proof. In: *Proceedings of the International Congress of Mathematics at Edinburgh* (1958) 289–299.

1970 Principles of proof and ordinals implicit in given concepts. In: *Intuitionism and Proof Theory*, eds. J. Myhill, A. Kino, and R.E. Vesley. Amsterdam: North-Holland Publ. Co. (1958).

1972 Which number theoretic problems can be solved in recursive progressions on Π^1_1-paths through O? *J. Symb. Log.* **37** (1972) 311–334.

Paris, J., and L. Harrington

1977 A mathematical incompleteness in Peano arithmetic. In: *Handbook of Mathematical Logic*, ed. J. Barwise. Amsterdam: North-Holland Publ. Co. (1977).

Post, E.L.

1944 Recursively enumerable sets of positive integers and their decision problems. *B. Am. Math. Soc.* **50** (1944) 284–316.

Simpson, S.G.

1986 Unprovable theorems and fast-growing functions. In: *Logic and Combinatorics*, ed. S.G. Simpson, pp. 359–394. Providence, RI: American Mathematical Society (1986).

Takeuti, G.

1987 *Proof Theory*, 2nd. ed. Amsterdam: North-Holland Publ. Co. (1987).

Turing, A.M.

1936-37 On computable numbers, with an application to the Entscheidungsproblem. *P. Lond. Math. Soc. (2)* **42** (1936-7) 230–265; A correction, ibid. **43** (1937) 544–546 (reprinted in Davis *1965*).

1937 Computability and λ-definability. *J. Symb. Log.* **2** (1937) 153–163.

1939 Systems of logic based on ordinals. *P. Lond. Math. Soc. (2)* **45** (1939) 161–228 (reprinted in *Davis 1965*).

1969 Intelligent machinery. In: *Machine Intelligence 5*, eds. B. Meltzer and D. Michie. Edinburgh: Edinburgh University Press (1969).

Visser, A.

1981 An incompleteness result for paths through or within *O*. *P. Kon. Ned. A* **84(2)** (1981) 237–243; also pub. in *Ind. Math.* **43.**

Weyl, H.

1918 *Das Kontinuum. Kritische Untersuchungen über die Grundlagen der Analysis.* Leipzig: Veit (1918).

Mathematical Logic and the Origin of Modern Computers*

Martin Davis

The very word *computer* immediately suggests one of the main uses of these remarkable devices: an instrument of calculation. But it is a matter of widespread experience that modern computers can be used for many purposes having no evident connection with numerical computation. The main thesis of this article is that the source of this generalized conception of the scope of computers is to be found in the vision of a computer as an engine of logic implicit in the abstract theory of computation developed by mathematical logicians.

The connection between logic and computing is apparent even from the everyday use of language: the English word "reckon" means both to calculate and to conclude. Without trying to understand this connection in any very profound manner, we can certainly see that computation is a (very restricted) form of reasoning. To see the connection in the opposite direction, imagine our seeking to demonstrate to a skeptic that some conclusion follows logically from certain assumptions. We present a "proof" that our claim is correct, only to be faced by the demand that we demonstrate that our proof is correct. If we then attempt a proof that our previous "proof" was correct, we clearly are faced with an infinite regress. The way out that has been found is to insist on a purely algorithmic criterion for logical correctness – a proof is correct if it proceeds according to rules whose correct application can be verified in a purely computational manner.

There are many examples of important concepts and methods first introduced by logicians which later proved to be important in computer science. Tracing the paths along which some of these ideas found their way from theory to practice is a fascinating (and often frustrating) task for the historian of ideas. The subject of this paper is Alan Turing's discovery of the universal (or all-purpose) digital computer as a mathematical abstraction. This concept was introduced by Turing as part of

* This paper was first published in *Studies in the History of Mathematics*, ©1987 by The Mathematical Association of America, and is reprinted here with their permission.

his solution to a problem that David Hilbert had called the "principal problem of mathematical logic" (Hilbert and Ackermann *1928*). We will try to show how this very abstract work helped to lead Turing and John von Neumann to the modern concept of the electronic computer.

But first, before discussing the work of Alan Turing, we will see how some of the underlying themes of computer science had already appeared in the seventeenth century in the work of G.W. Leibniz (1646–1716).

1. Leibniz' Dream

It is striking to note the many different ways in which Gottfried Leibniz anticipated what later came to be central concerns in computer science. He made an important invention, the so-called *Leibniz wheel*, which he used as early as the 1670's to build a mechanical calculating machine that could add, subtract, multiply, and divide. He showed keen awareness of the great advantages to be expected from the mechanization of computation. Thus, Leibniz said of his calculator:

> And now that we may give final praise to the machine we may say that it will be desirable to all who are engaged in computations . . . managers of financial affairs, merchants, surveyors, geographers, navigators, astronomers . . . But limiting ourselves to scientific uses, the old geometric and astronomic tables could be corrected and new ones constructed. . . . it will pay to extend as far as possible the major Pythagorean tables; the table of squares, cubes, and other powers; and the tables of combination, variations, and progressions of all kinds. . . . Also the astronomers surely will not have to continue to exercise the patience which is required for computation. . . . For it is unworthy of excellent men to lose hours like slaves in the labor of computation. (Smith *1929*, pp. 180–181)

Leibniz was one of the first (Ceruzzi *1983*, p. 40, footnote 11) to work out the properties of the binary number system, which of course has turned out to be fundamental for computer science. He proposed the development of a *calculus of reason* or *calculus ratiocinator* and actually proceeded to develop what amounts to a fragment of Boolean algebra (Parkinson *1966*, pp. 132–133; Davis *1983*, pp. 2–3). Finally, there was Leibniz' amazing program calling for the development of a universal language – a *lingua characteristica* – which would not only incorporate the calculus ratiocinator, but would also be suitable for communication and would include scientific and mathematical knowledge. Leibniz hoped to mechanize much of thought, saying that the mind "will be freed from having to think directly of things themselves, and yet everything will come out correctly" (Parkinson *1966*, p. xvii). Leibniz imagined problems in human affairs being handled by a learned committee sitting around a table and saying (Kneale and Kneale *1962*, p. 328): "*Calculemus*" i.e., "Let us calculate!"

The importance with which Leibniz regarded these projects is clear from his assessment:

> For if praise is given to the men who have determined the number of regular solids – which is of no use, except insofar as it is pleasant to contemplate – and if it is thought to be an exercise worthy of a mathematical genius to have brought to light the more elegant properties of a conchoid or cissoid, or some other figure which rarely has any use, how much better will it be to bring under mathematical laws human reasoning, which is the most excellent and useful thing we have. (Parkinson *1966*, p. 105)

It is at times amusing to imagine some great person from a past age reacting to one of the marvels of the contemporary world. Confronted by a modern computer, Leibniz would surely have been awestruck by the wonders of twentieth century technology. But perhaps he would have been better equipped than any other seventeenth century person to comprehend the scope and potential of these amazing machines.

2. Alan Turing's Analysis of the Concept of Computation

A century and a half ago, Charles Babbage already had conceived of an all-purpose automatic calculating machine, his proposed but never constructed *analytical engine*. Babbage's device was intended to carry out numerical computations of the most varied kind that arise in algebra and mathematical analysis. To emphasize the power and scope of his engine, Babbage remarked facetiously that "it could do everything but compose country dances" (Huskey and Huskey *1980*, p. 300). A contemporary computer expert seeking a figure of speech to bring home to a popular audience the widespread applicability of computers would select a different example. For we know that today's computers can perfectly well be programmed to compose country dances (although presumably not of the finest quality). While for Babbage it was self-evident that calculating machines could not be expected to compose dances, it does not strike us today as being at all out of the question. Clearly, our very concept of what constitutes "computation" has been altered drastically. We shall see how the modern view of computation developed out of the work in mathematical logic of Alan Turing.

Babbage never succeeded in constructing his engine, in large part because of the limitations of nineteenth century technology. In fact, it was only with some of the electro-mechanical calculators that began to be built during the 1930's (for example, by Howard Aiken at Harvard University) that Babbage's vision was fully realized. But during the 1930's and 1940's no one involved with this work suggested the possibility of designing an automatic computer that not only could do everything that Babbage had envisioned, but also could be used for commercial purposes, or for that matter, to "compose country dances". Even as late as 1956, Howard Aiken, himself a pioneer of modern computing, could write:

> If it should turn out that the basic logics of a machine designed for the numerical solution of differential equations coincide with the logics of a machine intended to make bills for a department store, I would regard this as the most amazing coincidence that I have ever encountered. (Ceruzzi *1983*, p.43)

If Aiken had grasped the significance of a paper by Alan Turing that had been published two decades earlier (Turing *1936-7*), he would never have found himself in the position of making such a statement only a few years before machines that performed quite well at both of the tasks he listed were readily available.

Alan Mathison Turing was born on June 23, 1912 in London. His father was a civil servant in India, and Turing spent most of his childhood away from his parents.[1] After five years at Sherborne, a traditional English public school he was awarded a fellowship to King's College at Cambridge University. Turing arrived at Cambridge in 1931. This was just after the young logician Kurt Gödel had startled the mathematical world by demonstrating that for any formal system adequate for elementary number theory, arithmetic assertions could be found that were not decidable within that formal system. In fact Gödel had even shown that among these "undecidable propositions" was the very assertion that the given formal system itself is consistent. This last result was devastating to Hilbert's program in the foundations of mathematics, which called for proving the consistency of more and more powerful formal systems using only very restricted proof methods, methods that Hilbert called *finitistic*. John von Neumann was probably the most brilliant of the young people who had been striving to carry out Hilbert's program. In addition to his contributions to Hilbert's program, von Neumann's intense interest in logic and foundations is also evidenced by his early papers on axiomatic set theory (von Neumann *1961*). However, after Gödel's discovery, von Neumann stopped working in this field. In the spring of 1935, Turing attended a course of lectures by the topologist M.H.A. Newman on *Foundations of Mathematics* in which Hilbert's program and Gödel's work were among the topics discussed. In particular, Newman called the attention of his audience to Hilbert's *Entscheidungsproblem*, a problem which Hilbert had called the "principal problem of mathematical logic".

In 1928, a little textbook of logic by Hilbert and Wilhelm Ackermann, entitled *Grundzüge der theoretischen Logik*, had been published. The book emphasized first order logic, the logic of *and, or, not, if ... then, for all,* and *there exists*, which the authors called the *engere Funktionenkalkül*. The authors showed how the various parts of mathematics could be formalized within first order logic, and a simple set of

1 In his authoritative biography, Andrew Hodges (*1983*, p. 132) quotes Turing as having, on at least one occasion, attributed his homosexuality to his childhood in boarding schools in England far from his parents in India. (Hodges himself makes it clear that he does not accept this explanation.)

rules of proof was given for making logical inferences. They noted that any inference that can be carried out according to their rules of proof is also *valid*, in the sense that in any mathematical structure in which all the premises are true, the conclusion is also true. Hilbert and Ackermann then raised the problem of *completeness:* if an inference is valid (in the sense just explained), would it always be possible, using their rules of proof, to obtain the conclusion from the premises? This question was answered affirmatively two years later by Gödel in his doctoral dissertation at the University of Vienna. Another problem raised in the *Grundzüge* by Hilbert and Ackermann was the Entscheidungsproblem, the problem of finding an algorithm to determine whether a given proposed inference is valid. By the completeness theorem from Gödel's dissertation, this problem is equivalent to seeking an algorithm for determining whether a particular conclusion may be derived from certain premises using the Hilbert-Ackermann rules of proof. The Entscheidungsproblem was called the "principal problem of mathematical logic", because an algorithm for the Entscheidungsproblem could, in principle, be used to answer any mathematical question: it would suffice to employ a formalization in first-order logic of the branch of mathematics relevant to the question under consideration. Alan Turing's attention was drawn to the Entscheidungsproblem by Newman's lectures, and he soon saw how to settle the problem negatively. That is, Turing showed that no algorithm exists for solving the Entscheidungsproblem. The tools that Turing developed for this purpose have turned out to be absolutely fundamental for computer science.

If a positive solution of the Entscheidungsproblem would lead to algorithms for settling all mathematical questions, then it must follow that if there is even one problem that has no algorithmic solution, then the Entscheidungsproblem itself must have no algorithmic solution. Now, the intuitive notion of *algorithm* serves perfectly well when what we only need to verify is that some proposed procedure does indeed constitute a positive solution to a given problem. However, remaining at this intuitive level, we could not hope to prove that some problem has no algorithmic solution. In order to be certain that no algorithm will work, it would appear necessary to somehow survey the class of all possible algorithms. This is the task that Turing set himself.

Turing began with a human being who would carry out the successive steps called for by some algorithm; that is, Turing proposed to consider the behavior of a "computer". Here the word *computer* refers to a person carrying out a computation; this was how Turing (and everyone else) used the word in 1935. Turing then proceeded (Turing *1936-7*), by a sequence of simplifications, each of which could be seen to make no essential difference, to obtain his characterization of computability.

Turing's first simplification was to assume that "the computation is carried out on one-dimensional paper, i.e., on a tape divided into squares" since "it will be agreed that the two-dimensional character of paper is no essential of computation". Turing continued:

> I shall also assume that the number of symbols . . . used . . . is finite. . . . this restriction . . . is not very serious. It is always possible to use sequences of symbols in the place of single symbols. . . . The behavior 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 . . . squares which the computer can observe at one moment. We will also suppose that the number of states of mind which need be taken into account is finite.[2]

Turing next argues that the entire computation can be thought of as consisting of atomic "simple" steps, each of which consists of:

1. a change of one symbol in one of the B "observed" squares (changes of more than one symbol can always be reduced to successive changes of a single symbol);
2. changes from the squares currently being "observed" to other squares no more than a distance of L squares away, where L is some constant; and
3. a change in the "state of mind" of the "computer".

The final step in Turing's analysis is the crucial remark that the outcome will in no way be altered if the human computer is replaced by a machine capable of a finite number of distinct states, or "*m*-configurations" as Turing called them, corresponding to the different states of mind the human computer possesses in the course of the computation. Each *m*-configuration is then completely characterized by a table which specifies the changes of types 1, 2, and 3 above, corresponding to each possible set of observed squares and symbols appearing in those squares. Hence, any algorithmic process can be carried out by machine, and, moreover, by a machine that conceptually is quite simple. As an addendum to his analysis, Turing pointed out that his machines could be simplified even further, permitting only one square at a time to be observed or "scanned" and allowing only a change of the observed square to the square immediately to the left or immediately to the right. Unlike the previous simplifications, this one can be justified rigorously: one can *prove* that nothing is lost by the restriction to the simpler machines.

We have arrived at the famous notion of a *Turing machine*: a Turing machine can be specified by a finite set of quintuples, each having one of the three forms:

$$p\alpha\beta Rq \quad \text{or} \quad p\alpha\beta Lq \quad \text{or} \quad p\alpha\beta Nq$$

Such a quintuple signifies that if the machine is in *m*-configuration p and the symbol α appears in the scanned square, then the machine will replace α by β, enter *m*-configuration q, and move one square to the right, move one square to the left, or not move at all, depending on whether the third symbol in the quintuple is R, L, or N, respectively. The quintuples which constitute a particular Turing machine

2 For discussion of the significance of this "finite mental states" hypothesis, see Wang *1974*, p. 93, and Webb *1980*, pp. 8 and 221–222.

determine the course of a computation. Turing's machines were to be *deterministic* in the sense that no two of its quintuples are allowed to begin with the same pair p, α, so that, at any particular time, no more than one action could be called for. If a machine were to be in *m*-configuration p scanning a symbol α, such that *no* quintuple of the machine begins with the pair $p\alpha$, then the computation would be at an end – the machine would "halt". At any stage of a computation by a Turing machine, there will be only a finite number of nonblank squares on the tape. But it is crucial that there be no a priori bound on this number. Thus, the tape is infinite, in the sense that additional (possibly blank) squares are always available to the right of the scanned square.

If Turing's analysis is accepted, then it may be concluded that any algorithmic process whatever, can be carried out by one of these Turing machines. In particular, if the Entscheidungsproblem has an algorithmic solution, then it can be solved by a Turing machine. Moreover, for the very reason that made it appropriate to call the Entscheidungsproblem "the principal problem of mathematical logic", if any problem at all can be shown to be unsolvable by Turing machines, the unsolvability of the Entscheidungsproblem should readily follow.

In obtaining his undecidable arithmetic propositions, Gödel had made use of a diagonal method that was formally similar to Cantor's proof that the set of real numbers is not countable. Therefore, it was quite natural that Turing should think of applying diagonalization. He called a real number[3] in the interval (0,1) *computable* if there exists a Turing machine that, beginning with a blank tape, successively prints the infinite sequences of 0's and 1's constituting the number's binary expansion. (Symbols other than 0 and 1 printed by the machine are just ignored for this purpose.) A machine that computes a real number in this sense was called *circle-free*; one that does not (because it never prints more than a finite number of 0's and 1's) was called *circular*. Since there are only countably many distinct finite sets of quintuples, the set of computable real numbers is likewise countable. Yet superficially, it appears that one could argue the reverse. As Turing expressed it, speaking in terms of the *sequences* if 0's and 1's corresponding to computable real numbers:

> 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. (Turing *1936-7*, p. 246; reprinted in Davis *1965*, p. 132)

Turing continued (the words in brackets are substituted for Turing's for expository reasons):

3 To include numbers outside the interval (0,1), Turing simply declared any number of the form $n \pm x$ computable if n is an integer and x is a computable number in the interval (0,1).

> 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 [finite set of quintuples determines] 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.

In other words if there were such a "general process", it could be used to delete non-circle-free machines from an enumeration of all possible finite sets of quintuples, thereby producing an enumeration of "the computable sequences by finite means". But then β, as defined above, would be computable, and we would be led to a contradiction. The only way out is to conclude that "there cannot be any such general process".[4] The problem of determining whether the Turing machine defined by a given finite set of quintuples is circle-free has no algorithmic solution! Of course, in this form the result depends on accepting Turing's analysis of the computation process. However, it is possible to state the result in the form of a rigorously proved theorem about Turing machines. For this purpose, let us imagine the very quintuples constituting a Turing machine themselves placed on a Turing machine tape. We could then seek to construct a Turing machine M which, begun with a set of quintuples defining any particular Turing machine N on its tape, will eventually halt with an affirmative or a negative message on its tape, according as N is or is not circle-free. Turing's argument can then be used to prove that there cannot be such a Turing machine M. (To make this entirely precise, it is necessary to be explicit as to how the quintuples, as well as the affirmative and negative output messages, are to be coded on the tape in terms of some finite alphabet. But this causes no difficulty.)

Turing next showed that there is no algorithm for the:

Blank Tape Printing Problem: *To determine whether a given Turing machine, starting with a blank tape, will ever print some particular symbol, say* |.

Turing's proof of this result proceeds by showing that if there were such an algorithm, then there must also be an algorithm for determining whether a given Turing machine is circle-free. This argument is a bit complicated,[5] and we outline a simpler proof that uses another diagonalization. First we show that there is no algorithm for the:

4 Turing recognized that although this proof is "perfectly sound", it "may leave the reader with a feeling that 'there must be something wrong' ", and he therefore supplied another proof that does not so closely approach paradox.

5 A remark for the knowledgeable: the problem of determining whether a given Turing machine is circle-free is complete of degree **0**'', and therefore can not be reduced to either of the printing problems.

General Printing Problem: *To determine whether a given Turing machine, starting with a given string of symbols on its tape, will ever print* |.

Suppose there were an algorithm for this problem. Then, in particular, there would be an algorithm to determine whether a given Turing machine, *starting with its own set of quintuples on its tape*, will ever print |. So, it would follow from Turing's analysis that a Turing machine M could be constructed that would respond to a set of quintuples on its tape by printing | if and only if the machine defined by that set of quintuples *never* prints | when started with its own set of quintuples on its tape. Now, what happens when M is started with *its own set of quintuples* on its tape? It eventually prints | if and only if it never prints | ! This contradiction shows that there can be no algorithm for the general printing problem. Finally, if there were an algorithm for the blank tape printing problem, it could also be used to solve the general printing problem. Namely, given the quintuples constituting a Turing machine M together with the string of symbols σ on its tape, we can construct a machine N that, beginning with a blank tape, first prints the string σ, and then behaves exactly like M. So N will eventually print | beginning with a blank tape if and only if M will eventually print | beginning with the string σ on its tape.

Turing used the fact that there is no algorithm for the blank tape printing problem to show that Hilbert's Entscheidungsproblem is likewise unsolvable. With each Turing machine M, he associated a formula $\alpha(M)$ of first-order logic which, roughly speaking, describes the behavior of M starting with a blank tape. He constructed a second formula β which has the interpretation that the symbol | eventually appears on the tape. It was then not difficult to see that β follows from $\alpha(M)$ by the Hilbert-Ackermann rules if and only if the Turing machine M eventually prints |. Thus, an algorithm for the Entscheidungsproblem would lead to an algorithm for the blank tape printing problem.

The notion of Turing machine was developed in order to solve Hilbert's Entscheidungsproblem. But it also enabled Turing to realize that it was possible to conceive of a single machine that was capable of performing all possible computations. As Turing expressed it: "It is possible to invent a single machine which can be used to compute any computable sequence." Turing called such a machine *universal*. A Turing machine U was to be called universal if, when started with a (suitably coded) finite set of quintuples defining a Turing machine M on its tape, U would proceed to compute the very same sequence of 0's and 1's that M would compute (beginning with an empty tape). Now, intuitively speaking, there clearly exists an algorithm that does what is required of the universal machine U; the algorithm just amounts to carrying out the instructions expressed by M's quintuples. Thus, the existence of a universal Turing machine is a consequence of Turing's analysis of the concept of computation. On the other hand, it is a rather implausible consequence. Why should we expect a single mechanism to be able to carry out algorithms for "the numerical solution of differential equations" as well as those needed to "make bills for a department store"? However, Turing did not simply depend on the validity of his analysis. He proceeded to produce in detail the actual quintuples needed to define a universal machine. Thus, in the light of the apparent implausibility of the existence of such a machine, Turing was entitled to regard his success in constructing one as a significant vindication of his analysis. The universal machine U actually given by Turing can be thought of as being specified by what is nowadays called an *interpretative* program. U operates by scanning the coded instructions (that is quintuples) on its tape and then proceeding to carry them out. Of course, many interpretative programs have been constructed in recent years to make it possible to run programs written in such languages as BASIC, LISP, SNOBOL, and PROLOG, but Turing's was the first.

Turing's analysis provided a new and profound insight into the ancient craft of computing. The notion of computation was seen as embracing far more than arithmetic and algebraic calculations. And at the same time, there emerged the vision of universal machines that "in principle" could compute everything that is computable. Turing's examples of specific machines were already instances of the art of programming; the universal machine in particular was the first example of an interpretative program. The universal machine also provided a model of a "stored program" computer in which the coded quintuples on the tape play the role of a stored program, and in which the machine makes no fundamental distinction between "program" and "data". Finally, the universal machine showed how "hardware" in the form of a set of quintuples thought of as a description of the functioning of a mechanism can be replaced by equivalent "software" in the form of those same quintuples in coded form "stored" on the tape of a universal machine.

While working out his proof that there is no algorithmic solution to the Entscheidungsproblem, Turing did not suspect that similar conclusions were being reached on the other side of the Atlantic. In fact, Newman had already received the first draft of Turing's paper, when an issue of the *American Journal of Mathematics* arrived in Cambridge containing an article by Alonzo Church of Princeton University, entitled "An Unsolvable Problem of Elementary Number Theory". In this paper, Church had already shown that there were algorithmically unsolvable problems. His paper did not mention machines, but it did point to two concepts, each of which had been proposed as explications of the intuitive notion of computability or, as Church put it, "effective calculability". The two concepts were *λ-definability*, developed by Church and his student Stephen Kleene, and *general recursiveness*, proposed by Gödel (in lectures at the Institute for Advanced Study in Princeton during the spring of 1934) as a modification of an idea of J. Herbrand. The two notions had been proved to be equivalent, and Church's unsolvable problem was in fact unsolvable with respect to either equivalent notion. Although in this paper Church had not drawn the conclusion that Hilbert's Entscheidungsproblem was itself unsolvable with respect to these notions, volume 1 (1936), number 1 of the new quarterly *Journal of Symbolic Logic* contained a brief note by Church in which he did exactly that. A later issue of volume 1 of the same journal contained an article by Emil Post, taking cognizance of Church's work, but proposing a formulation of computability very much like Turing's. Turing quickly showed that his notion of computability was equivalent to λ-definability, and he decided to attempt to spend some time in Princeton.

Thus, much of what Turing had accomplished amounted to a rediscovery of what had already been done in the United States. But his analysis of the notion of computation and his discovery of the universal computing machine were entirely novel, going beyond anything that had been done in Princeton. In particular, although Gödel had remained unconvinced by the evidence available in Princeton, that Church's proposal to identify effective calculability with the two equivalent proposed notions was correct, Turing's analysis finally convinced him.[6]

Turing was at Princeton for two academic years beginning in the summer of 1936. Formally, he was a graduate student, and indeed he did complete the requirements for a doctorate with Alonzo Church as his thesis adviser. His doctoral dissertation was his deep and important paper Turing *1939*, in which he studied the effect on Gödel undecidability of transfinite sequences of formal systems of increasing strength. This paper also introduced the key notion of an *oracle*, which made it possible to classify unsolvable problems,[7] and which is playing a very important role in current research in theoretical computer science.[8] Some writers have been confused about the circumstances under which Turing was a graduate student at Princeton, and have assumed that Turing's earlier work on computability had been done at Princeton under Church's supervision. A circumstance that may have helped lead to this confusion is that the published account (Turing *1936-7*) of the work on computability concludes with an appendix (in which a proof is outlined of the equivalence of Turing's concept of computability with Church's λ-definability) dated August 28, 1936 at "The Graduate College, Princeton University, New Jersey, U.S.A".[9]

6 For a discussion of some of the historical issues involved in these developments, as well as references, see Davis *1982*.

7 Thus suppose that we could somehow come to possess an oracle or "black box" which can tell us for a given set of quintuples whether the Turing machine defined by that set of quintuples is circle-free. Then it is not difficult to show that it is possible to construct a Turing machine which can solve either of the two unsolvable printing problems if only the machine is permitted to ask the oracle questions and make use of the answers. However, this will not work in the other direction. This is expressed by saying that the circle-free problem is of a *higher degree of unsolvability* than the printing problems.

8 Many important open questions in computer science ask whether certain inclusions between classes of sets of strings are proper. (The famous $P = NP$ problem is of this character.) In many cases (including the $P = NP$ problem), although the original question remains unresolved, it has proved possible to obtain answers when the problem is modified to permit access to suitable oracles.

9 Actually even this appendix must have been completed before Turing left England. Turing's departure was on September 23.

Fine Hall[10] in 1936 housed not only the mathematics faculty of Princeton University, but also the mathematicians who were part of the recently established Institute for Advanced Study. The great influx to the United States of scientists fleeing the Nazi regime had begun. The concentration of mathematical talent at Princeton during the 1930's came to rival and then surpass that at Göttingen, where David Hilbert held sway. Among those to be seen in the corridors of Fine Hall were Solomon Lefschetz, Hermann Weyl, Albert Einstein, and ... John von Neumann.[11] During Turing's second year at Princeton, he held the prestigious Procter Fellowship. Among the letters of recommendation written in support of his application was the following:

> June 1, 1937
>
> Sir,
>
> Mr. A.M. Turing has informed me that he is applying for a Proctor [sic] Visiting Fellowship to Princeton University from Cambridge for the academic year 1937–1938. I should like to support his application and to inform you that I know Mr. Turing very well from previous years: during the last term of 1935, when I was a visiting professor in Cambridge, and during 1936–1937, which year Mr Turing has spent in Princeton, I had opportunity to observe his scientific work. *He has done good work in branches of mathematics in which I am interested, namely: theory of almost periodic functions, and theory of continuous groups.* [emphasis added]
>
> I think that he is a most deserving candidate for the Proctor [sic] Fellowship, and I should be very glad if you should find it possible to award one to him.
>
> I am, Respectfully, John von Neumann (Hodges *1983*, p. 131)

Thus, as late as June 1937, either von Neumann was unaware of Turing's work on computability, or he did not think it appropriate to mention it in a letter of recommendation. There have been tantalizing rumors of important discussions between the two mathematicians about computing machinery, during the Princeton years, or later, during the Second World War. But there does not appear to be any real evidence that such discussions ever took place.[12] However, von Neumann's friend and collaborator Stanislaw Ulam, in a letter to Andrew Hodges (Hodges *1983*, p. 145),

10 Fine Hall in 1936 (and indeed through the 1950's) was a low-level attractive red brick building. The building where Princeton's mathematics department is housed today is also called Fine Hall; it is visible as a concrete tower from Highway US 1, a mile away.

11 Kurt Gödel, who had lectured at the Institute for Advanced Study during the spring of 1934, was unfortunately not in Princeton during Turing's stay. Gödel left Princeton in the fall of 1935 and did not return until after the Second World War had begun.

12 In the doctoral dissertation Aspray *1980*, (pp. 147–148) there is a reference to discussions between Turing and von Neumann at this time, on the question of whether "computing machines could be built which would adequately model any mental feature of the human brain". Aspray based his account on an interview with J.B. Rosser. However, in a conversation with the present author, Aspray explained that Rosser had not claimed to have himself overheard such discussions, and that Rosser had been unable to remember his source. Aspray indicated that he no longer believes that such a conversation actually occurred. In a recent letter, Alonzo Church indicates that he neither recalled nor could find any record of such "consultations". See also Randell *1972*.

mentioned a game that von Neumann had proposed during the summer of 1938 when he and Ulam were traveling together in Europe; the game involved "writing down on a piece of paper as big a number as we could, defining it by a method which indeed has something to do with some schemata of Turing's".[13] Ulam's letter also stated that "von Neumann mentioned to me Turing's name several times in 1939 in conversations, concerning mechanical ways to develop formal mathematical systems." On the basis of Ulam's letter it seems safe to conclude that, by the outbreak of the Second World War in September 1939, von Neumann was well aware of Turing's work on computability and regarded it as important.

When did Turing begin to think about the possibility of constructing a physical device that would be, in some appropriate sense, an embodiment of his universal machine? According to Turing's teacher, M.H.A. Newman, this was in Turing's mind from the very first. In an obituary article in *The Times*, Newman wrote:

> The description that he then gave of a "universal" computing machine was entirely theoretical in purpose, but Turing's strong interest in all kinds of practical experiment made him even then interested in the possibility of actually constructing a machine on these lines. (quoted in Hodges *1983*, p. 545)

In Princeton, Turing's "practical" interests included a developing concern with cryptanalysis. Possibly in this connection, he designed an electro-mechanical binary multiplier, and gaining access to the Physics Department graduate student machine shop,[14] he constructed various parts of the device, building the necessary relays himself. Another of Turing's interests during this period – an interest which combined the theoretical with practical computation – was the famous Riemann Hypothesis concerning the distribution of the zeros of the Riemann ζ-function. Shortly after Turing returned to England in the summer of 1938, he applied for and was granted $40 to build a special purpose analogue computer for computing Riemann's ζ-function,[15] which Turing hoped to use to test the Riemann hypothesis numerically (Hodges *1983*, pp. 138–140, 155–158). But even as Turing was beginning serious work on this machine, the Second World War intervened and moved him in quite another direction. The ζ-function machine was never completed.

13 This sounds very much as though von Neumann had anticipated the important Chaitin-Kolmogoroff notion of descriptive complexity.

14 The Palmer Physics laboratory was located next door to Fine Hall – there was even a convenient passageway joining the two buildings.

15 The design of this computer was based on that of a machine in Liverpool which was used to predict the tides.

3. *To Build a Brain*

Turing spent the war years at Bletchley Park, a country mansion that housed Britain's brilliant group of cryptanalysts. The Germans had developed improved versions of a commercial encrypting machine, the *Enigma*. The task of breaking the enigma code fell to the group at Bletchley Park. They were given a head start in their task by having access to the work of a group of Polish mathematicians who had succeeded with an earlier and considerably simpler version of the Enigma. Building on this work, Turing and Gordon Welchman (an algebraic geometer from Cambridge) progressed to the point where machines, called Bombes (the name first used by the Poles for their much more primitive device) could be built to decode everyday German military communications. Naval communications were Turing's special province, and by the summer of 1941, the information derived from the Bombes enabled the British Admiralty to defeat the German submarine offensive against Atlantic shipping that had been threatening to strangle a beleaguered Britain (Welchman *1982*; Hodges *1983*, pp. 160–210).[16] But this great success was a precarious one. It was clear that if the Germans introduced more complexity into their procedures, the Bombes would be overwhelmed. And so, more ambitious machines (which indeed turned out to be necessary) were constructed: first the Heath Robinson series, and later the Colossus. The latter, constructed in 1943 under the direct supervision of M.H.A. Newman, used vacuum tube circuits to carry out complex Boolean computations very rapidly. The Colossus contained 1500 tubes and was built in the face of skepticism on the part of the engineers that so many vacuum tubes could work together without a failure, long enough to get useful work done.

Thus, when the war ended, Turing had a solid basic knowledge of electronics, and was aware that large scale computing machines could be constructed using electronic circuits. The significance for Turing of this practical knowledge can not be fully grasped without taking into account the new conceptual framework for thinking about computing to which his work on computability had led him. For Turing had been led to conclude that computation was simply carrying out the steps in some "rule of thumb" process (as Turing expressed it in an address to the London Mathematical Society (Turing *1947*, p. 107)). A "rule of thumb" process is to be understood as one which can be carried out simply by following a list of unambiguous instructions referring to finite discrete configurations of whatever kind. Turing's work had also shown that, without loss of generality, one could restrict oneself to instructions of an extremely simple kind. Finally, it was possible to construct a single mechanical device capable, in principle, of carrying out any computation whatever. "It can be shown that a single machine ... can be made to do the work of all" (Turing *1947*, p. 112). As exciting as this prospect must have appeared, it was only part of Turing's remarkable vision. Turing dared to imagine not only that computation encompassed far more than mere calculation, but that it actually included the human mental processes that we call "thought". He was interested in much more than a machine capable of very rapid computation; Alan Turing wanted to build a brain. This vision had been the subject of much discussion at Bletchley Park, where Turing focused on chess as an example of human "thought" that should be capable of mechanization. The full scope of Turing's thought was only exposed to the public later, in Turing *1947* and in his now classical essay Turing *1950*. In 1947 Turing was already speaking of circumstances in which "one is obliged to regard the machine as showing intelligence. As soon as one can provide a reasonably large memory capacity it should be possible to experiment along these lines" (Turing *1947*, p. 123). As we shall see, Turing was eager to help build such machines. But for a second time, Turing's professional life was profoundly affected by developments in the Western Hemisphere.

16 Another of Turing's contributions to this effort was the invention of new statistical methods for dealing with the vast quantities of data contained in the Enigma "traffic". These methods were later rediscovered independently by the American statistician A. Wald who gave them the name *sequential analysis*. Surely there cannot be many ~~Britains~~ Britons whose contribution to the ultimate victory approached Turing's. A little over a decade after the turning of the tide in the "Battle of the Atlantic", being duly convicted of performing acts "of gross indecency", Turing was sentenced in a British court of law to a one year probation term, during which time he was required to submit to a course of hormone treatments that amounted to a temporary chemical castration. Such was the hero's reward!

4. *Von Neumann and the Moore School*

As has already been noted, by the summer of 1938 von Neumann was very much aware of Turing's work on computability. There is also evidence that, early on, he perceived that Turing's work had implications for the practice of computation. A wartime colleague of von Neumann recalled that "in about 1943 or 44 von Neumann was well aware of the fundamental importance of Turing's paper of 1936 ... and at his urging I studied it ... he emphasized ... that the fundamental conception is owing to Turing ..." (Hodges *1983*, pp. 145, 304). Herman Goldstine (who was von Neumann's close collaborator) said, "There is no doubt that von Neumann was thoroughly aware of Turing's work ..." (Goldstine *1972*, p. 174).

As with Turing, von Neumann's wartime work involved large-scale computation. But, where the cryptoanalytic work at Bletchley Park emphasized the discrete combinatorial side of computation, so in tune with Turing's earlier work, it was old-fashioned, heavy, number-crunching that von Neumann needed. Although he had tried to inform himself about new developments in computational equipment, von Neumann learned of the ENIAC project quite fortuitously on meeting the young mathematician Herman Goldstine at a railway station during the summer of 1944.

Von Neumann quickly became a participant in discussions with the ENIAC group at the Moore School in Philadelphia.

The Colossus with its 1500 vacuum tubes was an engineering marvel. The ENIAC with 18,000 tubes was simply astonishing. The conventional wisdom of the time was that no such assemblage could do reliable work; it was held that the mean free path between vacuum tube failures would be a matter of seconds. It was the chief engineer on the ENIAC project, John Prosper Eckert, Jr., who was largely responsible for the project's success. Eckert insisted on extremely high standards of component reliability. Tubes were operated at extremely conservative power levels, and the failure rate was kept to three tubes per week. The ENIAC was an enormous machine, occupying a large room. It was a decimal machine and was programmed by connecting cables to a plugboard (Burks and Burks *1981*), rather like an old-fashioned telephone switchboard.

By the time that von Neumann began meeting with the Moore School group, it was clear that there were no important obstacles to the successful completion of the ENIAC, and attention was focused on the next computer to be built, tentatively called the EDVAC. Von Neumann immediately involved himself with the problems of the *logical* organization of the new machine. As Goldstine (*1972*, p. 186) recalls, "Eckert was delighted that von Neumann was so keenly interested in the logical problems surrounding the new idea, and these meetings were scenes of greatest intellectual activity." Goldstine comments:

> This work on the logical plan for the new machine was exactly to von Neumann's liking and precisely where his previous work on formal logics [sic] came to play a decisive role. Prior to his appearance on the scene, the group at the Moore School concentrated primarily on the *technological* problems, which were very great; after his arrival he took over leadership on the *logical* problems. (Goldstine *1972*, p. 188)

A key idea emphasized in the meetings was that any significant advance over the ENIAC would require a substantial capacity for the internal storage of information. This was because communication with the exterior would be at speeds far slower than the internal electronic speeds at which the computer could function, and therefore constituted a potential bottleneck. Once again, John Eckert played a crucial role. He had previously shown how to modify a device called a delay line (originally developed by the physicist W.B. Shockley, who later invented the transistor) so as to be a working component of radar systems. These delay lines (which stored information in the form of a vibrating tube of mercury) were just what was needed.

The communication bottleneck just mentioned would be evident in the case of any computation involving the manipulation of large quantities of data. But it was even more crucial for the *instructions* that the computer would carry out. Indeed, it would make little sense for a computer to produce the results of a calculation rapidly,

only to wait idly for the next instruction. The solution was to store the instructions internally with the data: what has come to be called the "stored program concept".

The computers of the postwar period differed from previous calculating devices in having provision for internal storage of programs as well as data. But they were different in another more fundamental way. They were conceived, designed, and constructed, not as mere automatic calculators, but as *engines of logic*, incorporating the general notion of what it means to be computable and embodying a physical model of Turing's universal machine. Whereas there has been a great deal of discussion concerning the introduction of the "stored program concept", the significance of this other great, but rather subtle, advance has not been fully appreciated. In fact, the tendency has been to use the single term "stored-program concept" to include all of the innovations introduced with the EDVAC design. This terminological confusion may well be responsible, at least in part, for the fact that there has been so much acrimony about who deserves credit for the revolutionary advances in computing which took place at this time (see for example the report Aspray *1982*).

The key document in which this new conception of computer first appeared was the draft report von Neumann *1945* which quickly became known as the EDVAC Report. This report never advanced beyond the draft stage and is quite evidently incomplete in a number of ways. Yet it was widely circulated almost at once and was very influential. In fact the conception of computing machine it embodies has come to be known as the "von Neumann architecture". One element of controversy, which will probably never be fully resolved, is the question of how much of the EDVAC report represented von Neumann's personal contribution. Although Eckert and his consultant J.W. Mauchly later denied that von Neumann had contributed very much, shortly after the report appeared they wrote as follows:

> During the latter part of 1944, and continuing to the present time, Dr. John von Neumann ... has fortunately been available for consultation. He has contributed to many discussions on the logical controls of the EDVAC, has prepared certain instruction codes, and has tested these proposed systems by writing out the coded instructions for specific problems. Dr. von Neumann has also written a preliminary report in which most of the results of earlier discussions are summarized. ... In his report, the physical structures and devices ... are replaced by idealized elements to avoid raising engineering problems which might distract attention from the logical considerations under discussion. (Goldstine *1972*, p. 191; Metropolis and Worlton *1980*, p. 55)

Goldstine (apparently unaware of Turing's claim to be mentioned in this connection) comments:

> Von Neumann was the first person, as far as I am concerned, who understood explicitly that a computer essentially performed logical functions, ... he also made a precise and detailed study of the functions and mutual interactions of the various parts of a computer. Today this sounds so trite as to be almost unworthy of mention. Yet in 1944 it was a major advance in thinking. (Goldstine *1972*, pp. 191–192)

One way in which the EDVAC report betrays its unfinished state is by the large number of spaces clearly intended for references, but not filled in. Almost every page contains the abbreviation "cf." followed by a space. All the more significant is the one reference that von Neumann did supply: the reference, supplied in full, was to the paper McCulloch and Pitts *1943* in which a mathematical theory of idealized neurons had been developed. Von Neumann suggested that basic vacuum tube circuits could be thought of as physical embodiments of these neurons. Here there are two connections with Turing's ideas. The first, more obvious one, is that, like Turing, von Neumann was thinking of a computer as being like a brain (or at least a nervous system). In Ulam's letter to Hodges quoted above, Ulam alluded to this confluence, writing in a postscript: "Another coincidence of ideas: both Turing and von Neumann wrote of 'organisms' beyond mere computing machines."[17] But a more explicit connection with Turing's work becomes evident on further study. McCulloch (see von Neumann *1963*, p. 319) later stated that the paper which von Neumann did reference had been directly inspired by Turing *1936-7*. In fact, the paper itself cites the fact that a universal Turing machine can be modeled in a suitable version of the neural net formalism as the principal reason for believing in the adequacy of the formalism.

There is other evidence that von Neumann was concerned with universality in Turing's sense. Thus he spoke (Randell *1982*, p. 384) of the "logical control" of a computer as being crucial for its being "as nearly as possible *all purpose*". In order to test the general applicability of the EDVAC, von Neumann wrote his first serious program, not for numerical computation of the kind for which the machine's order code was mainly developed, but rather to carry out a computational task of a logical-combinatorial nature, namely the efficient sorting of data.[18] The success of this program helped to convince von Neumann that "it is legitimate to conclude already on the basis of the now available evidence, that the EDVAC is very nearly an 'all purpose' machine, and that the present principles for the logical controls are sound" (Goldstine *1972*, p. 209). Articles written within a year of the EDVAC report confirm von Neumann's awareness of the basis in logic for the principles underlying the design of electronic computers. The introduction to one such article states:

> In this article we attempt to discuss [large scale computing] machines from the viewpoint not only of the mathematician but also of the engineer and the logician, i.e. of the ... person or group of persons really fitted to plan scientific tools. (Goldstine and von Neumann *1946*)

17 I am grateful to Andrew Hodges for making a copy of this letter available to me.

18 The sorting algorithm that von Neumann implemented belongs to the family of so-called "merge" sorts. For a very interesting discussion of this program and of the proposed EDVAC order code, see Knuth *1970*.

Another article (Burks, Goldstine, and von Neumann *1946*) clearly alludes to Turing's work, even as it indicates that purely logical considerations are not enough:

> It is easy to see by formal-logical methods that there exist codes that are in abstracto adequate to control and cause the execution of any sequence of operations which are individually available in the machine and which are, in their entirety, conceivable by the problem planner. The really decisive considerations from the present point of view, in selecting a code, are of more practical nature: simplicity of the equipment demanded by the code, and the clarity of its application to the actually important problems together with the speed of its handling those problems. It would take us much too far afield to discuss these questions at all generally or from first principles.

There has been much acrimony over the question of just what von Neumann had contributed; indeed, this question even became the subject of extensive litigation. Much of the controversy concerns the relative significance of the contributions of von Neumann on the one hand, and of Eckert and Mauchly on the other. In particular, some recent studies challenge the belief that von Neumann's technical contributions were of much importance. (See the semi-popular history Shurkin *1984* and the meticulously researched Stern *1981*.) It is not difficult to understand why this should be. The Turing–von Neumann view of computers is conceptually so simple and has become so much a part of our intellectual climate that it is difficult to understand how radically new it was. It is far easier to appreciate the importance of a new invention, like the mercury delay line, than of a new and abstract idea.

5. *The ACE*

Meanwhile, what of Turing? His mother (quoted in Hodges *1983*, p. 294) reports him saying "round about 1944" that he had plans "for the construction of a universal computer". During the war, he had been telling colleagues that he wanted to build a "brain". He proposed to construct an electronic device that would be a physical realization of his universal machine. Early in 1945, while on a trip to the United States, J.R. Womersley, Superintendent of the Mathematics Division of the National Physical Laboratory of Great Britain, was introduced to the ENIAC and to the EDVAC report. As early as 1938, Womersley had considered the possibility of constructing a "Turing machine using automatic telephone equipment". His reaction to what he had learned in the United States was to hire Alan Turing (Hodges *1983*, pp. 306–307). By the end of 1945, Turing had produced his remarkable ACE report (Turing *1945*). The excellent article Carpenter and Doran *1977* contains an analysis of the ACE report, comparing it in some detail with von Neumann's EDVAC report. They note that, whereas the EDVAC report "is a draft and is unfinished ... more important ... is incomplete ...", the ACE report "is a complete description of a computer, right down to the logical circuit diagrams" and even including "a cost estimate of \$ 11,200". Not surprisingly, Turing showed that he understood the scope of universality. He suggested that his ACE might be able to play chess and to solve jigsaw puzzles. The ACE report contains explicit mention of features such as an instruction address register and truly random access to memory locations, neither of which is dealt with in the EDVAC report, although both are already to be found in Burks, Goldstine, and von Neumann *1946*. Although it is known (Goldstine *1972*, p. 218) that the ACE report quickly made its way across the Atlantic, it seems impossible to determine whether the ACE report influenced the American developments.

It should also be mentioned that the ACE report showed an understanding of numerous issues in computer science well ahead of its time. Of these perhaps the most interesting are microprogramming, and the use of a stack for a hierarchy of subroutine calls. We have already mentioned Turing's address to the London Mathematical Society in February 1947, in which he unveiled the scope of his vision regarding the ACE and its successors. In this talk he explained that one of the central conclusions of his earlier work on computability was that "the idea of a 'rule of thumb' process and a 'machine process' were synonymous ... Machines such as the ACE may be regarded as practical versions of ... the type of machine I was considering ... There is at least a very close analogy ... digital computing machines such as the ACE ... are in fact practical versions of the universal machine." Turing went on to raise the question of "how far it is in principle possible for a computing machine to simulate human activities". This led him to propose the possibility of a computing machine programmed to learn and permitted to make mistakes. "There are several theorems which say almost exactly that ... if a machine is expected to be infallible, it cannot also be intelligent ... But these theorems say nothing about how much intelligence may be displayed if a machine makes no pretence at infallibility."

Turing was much better at communicating in this visionary manner than he was in dealing with the bureaucrats who actually allocate resources, and he had considerable difficulty in getting his ideas put into practice. However, a machine embodying much of the design in the ACE report was eventually constructed, the Pilot ACE, and a successful commercial machine, the DEUCE, followed.

6. *Logic and the Future*

Since 1945, we have witnessed a number of breath-taking technological developments that have completely altered the physical form and the computational power of computers. I sit writing this essay using a text-editing program running on my personal microcomputer with a memory capacity of 5 million bits. The "johniac" computers on which I learned to program in the 1950's had a total memory of 41,000 bits and were only affordable by institutions. But the connection between logic and

computing continues to be a vital one, and the lesson of universality, of the possibility of replacing the construction of diverse pieces of hardware by the programming of a single all-purpose device continues to be relevant. In fact the very existence of personal microcomputers is the result of this lesson being learned anew in the case of integrated circuit technology. In 1971, faced with the requirement for ever more complex and diverse "chips" by his employer, the Intel Corporation, Marcian Hoff found the solution: a single all-purpose programmable chip, and the microprocessor was born. We can foresee that this will happen again and again as technology continues its march towards faster and smaller components.

Acknowledgement. The research for this article was done while I was a John Simon Guggenheim Fellow (1983–84). The article is part of a comprehensive investigation of the relation between logic and computation. I am very grateful to the Guggenheim Foundation for its support. I also wish to thank Professors Harold Edwards and Esther Phillips who read an earlier draft and made many helpful suggestions.

References

Aspray, W.F.

1980 *From Mathematical Constructivity to Computer Science: Alan Turing, John von Neumann and the Origins of Computer Science in Mathematical Logic.* Doctoral dissertation, University of Wisconsin.

1982 History of the stored-program concept. Report on a session held on this subject as part of Pioneer Day at the National Computer Conference, June 1982. *Ann. Hist. C.* **4** (1982) 358–361.

Burks, A.W., H.H. Goldstine, and J. von Neumann,

1946 Preliminary discussion of the logical design of an electronic computing instrument, Institute for Advanced Study; reprinted in von Neumann *1963*, pp. 34–79.

Burks, A.W., and A.R. Burks

1981 The ENIAC: First general-purpose electronic computer. *Ann. Hist. C.* **3** (1981) 310–399.

Carpenter, B.E. and R.W. Doran

1977 The other Turing machine. *Comp. J.* **20** (1977) 269–279.

Carpenter, B.E. and R.W. Doran (eds.)

1986 *A.M. Turing's ACE Report of 1946 and Other Papers.* Cambridge, MA: MIT Press (1986).

Ceruzzi, P.E.

1983 *Reckoners, the Prehistory of the Digital Computer from Relays to the Stored Program Concept, 1933–1945.* Westport, CN: Greenwood Press (1983).

Davis, M.

1965 *The Undecidable*. New York: Raven Press (1965).

1982 Why Gödel didn't have Church's thesis. *Inf. Contr.* **54** (1965) 3–24.

1983 The prehistory and early history of automated deduction. In: *Automation of Reasoning*, vol. 1, eds. J. Siekmann and G. Wrightson. Berlin: Springer-Verlag (1983).

Goldstine, H.H.

1972 *The Computer from Pascal to von Neumann*. Princeton, NJ: Princeton University Press (1972).

Hilbert, D. and W. Ackermann

1928 *Grundzüge der Theoretischen Logik*. Berlin: Julius Springer (1928).

Hodges, A.

1983 *Alan Turing: The Enigma*. New York: Simon and Schuster (1983).

Huskey, V.R. and H.D. Huskey

1980 Lady Lovelace and Charles Babbage. *Ann. Hist. C.* **2** (1980) 299–329.

Kneale, W. and M. Kneale

1962 *The Development of Logic*. Oxford: Oxford University Press (1962).

Knuth, D.E.

1970 Von Neumann's first computer program. *Comp. Sur.* **2** (1970) 247–260.

McCulloch, W.S. and W. Pitts

1943 A logical calculus of the ideas immanent in nervous activity. *B. Math. Biophy.* **5** (1943) 115–133; reprinted in: McCulloch, W.S., *Embodiments of Mind*, pp. 19–39. Cambridge, MA: MIT Press (1965).

Metropolis, N., J. Howlett, and G.-C. Rota (eds.)

1980 *A History of Computing in the Twentieth Century*. New York: Academic Press (1980).

Metropolis, N. and J. Warlton

1980 A trilogy of errors in the history of computing. *Ann. Hist. Comp.* **2** (1980) 49–59.

Parkinson, G.H.R.

1966 *Leibniz – Logical Papers*. Oxford: Oxford University Press (1966).

Post, E.

1936 Finite combinatory processes. Formulation I. *J. Symb. Log.* **1** (1936) 103–105; reprinted in Davis *1965*.

Randell, B.

1972 On Alan Turing and the origins of digital computers. *Mach. Intell.* **7** (1972) 3–20.

Randell, Brian (ed.)

1977 Colossus: Godfather of the computer. *New Sci.* **73** (1977) 346–348; reprinted in Randell *1982*, pp. 349–354.

1982 *The Origins of Digital Computers, Selected Papers*, (third edition). Berlin: Springer-Verlag (1982).

Shurkin, J.

1984 *Engines of the Mind: A History of the Computer.* W.W. Norton & Company (1984).

Smith, D.E.

1929 *A Source Book in Mathematics*. McGraw-Hill (1929).

Stern, N.

1980 John von Neumann's influence on electronic digital computing, 1944–1946. *Ann. Hist. C.* **2** (1980) 349–362.

1981 *From Eniac to Univac: An Appraisal of the Eckert-Mauchly Machines*. Digital Press (1981).

Turing, A.M.

1936-7 On computable numbers with an application to the Entscheidungsproblem. *P. Lond. Math. Soc. (2)* **42** (1936-7) 230–267; A correction, ibid. **43** (1937) 544–546; reprinted in Davis *1965*, pp. 155–222.

1945 *Proposals for Development in the Mathematics Division of an Automatic Computing Engine (ACE)*, 1945, Report e882 National Physical Laboratory of Great Britain; reprinted 1972 with a forward by D.W. Davies as National Physical Laboratory Report, *Com. Sci.* **45**, and in Carpenter and Doran *1986*, pp. 20–105.

1947 Lecture to the London Mathematical Society, first published in Carpenter and Doran *1986*, pp. 106–124.

1950 Computing machinery and intelligence. *Mind*, **LIX** No. 236 (1950).

von Neumann, J.

1945 First draft of a report on the EDVAC, Moore School of Electrical Engineering, University of Pennsylvania, unpublished; reprinted in Stern *1981*, pp. 177–246.

1961 *Collected Works*, vol. 1, ed. A.H. Taub. Pergamon Press (1961).

1963 *Collected Works*, vol. 5, ed. A.H. Taub. Pergamon Press (1963).

Wang, H.

1974 *From Mathematics to Philosophy*. Humanities Press (1974).

Webb, J.C.

1980 *Mechanism, Mentalism, and Metamathematics: An Essay on Finitism*. Doordrecht: D. Reidel (1980).

Welchman, G.
1982 *The Hut Six Story*. McGraw-Hill (1982).

Woodger, M.
1958 The history and present use of digital computers in the National Physical Laboratory. *Proc. Cont. Aut.* (November 1958) 437–442; reprinted with a correction in Carpenter and Doran *1986*, pp. 125–140.

Part II