5 *Turing and the computer*

B. Jack Copeland and Diane Proudfoot

The Turing machine

In his first major publication, 'On computable numbers, with an application to the Entscheidungsproblem' (1936), Turing introduced his abstract Turing machines.[1] (Turing referred to these simply as 'computing machines'—the American logician Alonzo Church dubbed them 'Turing machines'.[2]) 'On Computable Numbers' pioneered the idea essential to the modern computer—the concept of controlling a computing machine's operations by means of a program of coded instructions stored in the machine's memory. This work had a profound influence on the development in the 1940s of the electronic stored-program digital computer—an influence often neglected or denied by historians of the computer.

A Turing machine is an abstract conceptual model. It consists of a scanner and a limitless memory-tape. The tape is divided into squares, each of which may be blank or may bear a single symbol ('0' or '1', for example, or some other symbol taken from a finite alphabet). The scanner moves back and forth through the memory, examining one square at a time (the 'scanned square'). It reads the symbols on the tape and writes further symbols. The tape is both the memory and the vehicle for input and output. The tape may also contain a program of instructions. (Although the tape itself is limitless—Turing's aim was to show that there are tasks that Turing machines cannot perform, even given unlimited working memory and unlimited time—any input inscribed on the tape must consist of a finite number of symbols.)

A Turing machine has a small repertoire of basic operations: *move left one square*, *move right one square*, *print*, and *change state*. Movement is always by one square at a time. The scanner can print a symbol on the scanned square (after erasing any existing symbol). By changing its state the machine can (as Turing put it) 'remember some of the symbols which it has "seen" (scanned) previously'.[3] Turing did not specify a mechanism for changing state (Turing machines are abstractions and proposing a specific mechanism is unnecessary) but one can easily be imagined. Suppose that a device within the scanner consists of a dial with a finite number of positions, labelled '**a**', '**b**', '**c**', and so on, each position counting as a different state. Changing state consists in shifting the dial's pointer from one labelled position to another. This device functions as a simple memory; for example, a dial with three positions can be used to record whether the square that the scanner has just vacated contained '0' or '1', or was blank.

The operation of the machine is governed by (what Turing called) a table of instructions. He gave the following simple example.[4] A machine—call it M—begins work with an endless blank tape and with the scanner positioned over any square of the tape. M has four states, labelled '**a**', '**b**', '**c**', and '**d**', and is in state **a** when it starts work. In the table below, 'R' is an abbreviation of the instruction 'move right one square', 'P[0]' is an abbreviation of 'print 0 on the scanned square', and analogously 'P[1]'. The top line of the table reads: if you are in state **a** and the square you are scanning is blank, then print 0 on the scanned square, move right one square, and go into state **b**.

State	Scanned square	Operations	Next state
a	Blank	P[0], R	**b**
b	Blank	R	**c**
c	Blank	P[1], R	**d**
d	Blank	R	**a**

Acting in accordance with this table of instructions—or program—M prints alternating binary digits on the tape, 0 1 0 1 0 1 . . ., working endlessly to the right from its starting place, leaving a blank square in between each digit.

The universal Turing machine (UTM)

The UTM is universal in that it can be programmed to carry out any calculation that could in principle be performed by a 'human computer'—a clerk who works by rote and has unlimited time and an endless supply of paper and pencils. (Before the advent of the electronic computer, many thousands of human computers were employed in business, government, and research establishments.)

The universal machine has a single, fixed table of instructions built into it ('hard-wired', so to speak, into the machine). Operating in accordance with this one fixed table, the UTM can read and execute coded instructions inscribed on its tape. This ingenious idea—the concept of controlling the function of the computing machine by storing a program of instructions in the machine's memory—is fundamental to computer science. An instruction table for carrying out a desired task is placed on the machine's tape in a suitably encoded form, the first line of the table occupying the first so many squares of the tape, the second line the next so many squares, and so on. The UTM reads the instructions and carries them out on its tape. Different programs can be inscribed on the tape, enabling the UTM to carry out any task for which a Turing-machine instruction table can be written—thus a single machine of fixed structure is able to carry out every computation that can be carried out by any Turing machine whatsoever.

In 1936 the UTM existed only as an idea. But right from the start Turing was interested in the possibility of actually building such a machine.[5] His wartime acquaintance with electronics was the key link between his earlier theoretical work and his 1945 design for an electronic stored-program digital computer.

The Church–Turing thesis

The Church–Turing thesis (also known simply as 'Church's thesis' and 'Turing's thesis') played a pivotal role in Turing's argument (in 'On Computable Numbers') that there are well-defined mathematical tasks that cannot be carried out by a rote method or algorithm.[6] One of Turing's most accessible formulations of the Church–Turing thesis is found in a report written in 1948, 'Intelligent Machinery':

> LCMs [Turing machines] can do anything that could be described as 'rule of thumb' or 'purely mechanical'.[7]

Turing remarked:

> This is sufficiently well established that it is now agreed amongst logicians that 'calculable by means of an LCM' is the correct accurate rendering of such phrases.[8]

In 'On Computable Numbers', having proved that there are tasks the universal Turing machine cannot carry out (even given unlimited time and tape), Turing appealed to the Church–Turing thesis in moving to the conclusion that there are tasks that cannot be accomplished by means of any rote method.

The Church–Turing thesis is sometimes said to state the 'limits of machines'.[9] For example, Newell asserted:

> That there exists a most general formulation of machine and that it leads to a unique set of input–output functions has come to be called *Church's thesis*.[10]

However, this goes well beyond anything that Turing and Church said themselves. Turing's concern was not to give 'a most general formulation of machine', but to state the limits of what can be accomplished by a human being working by rote. Turing, like Church, aimed to show that some well-defined tasks cannot be carried out by a human being working in this way; it does not follow from this (and nor did they claim that it does) that there are *no machines* able to carry out these tasks.[11]

Cryptanalytic machines

Turing completed the logical design of the famous Bombe—built to break German Enigma messages—in the last months of 1939.[12] His designs were handed over to Keen at the British Tabulating Machine Company in Letchworth, where the engineering development was carried out.[13] The first Bombe, named 'Victory', was installed at the Government Code and Cypher School (GC & CS) at Bletchley Park early in 1940.[14] (An improved model—'Agnus', short for 'Agnus Dei', but later corrupted to 'Agnes' and 'Aggie'—which contained Welchman's ingenious diagonal board was installed some months later.[15]) The Bombe was a 'computing machine'—a term for any machine able to do work that could be done by a human computer—but one with a very narrow and specialized purpose, namely searching through the wheel-positions of the Enigma machine, at superhuman speed, in order to find the positions at which a German message had been encrypted. The Bombe produced likely candidates, which were tested by hand on an Enigma machine (or a replica of one)—if German emerged (even a few words followed by nonsense), the candidate settings were the right ones. The Bombe was based on the electromagnetic relay, although some later versions were electronic (i.e. valve-based) and in consequence faster. Relays are small switches consisting of a moving metal rod—which opens and closes an electrical circuit—and an electrical coil, the magnetic field of which moves the rod. Electronic valves, called 'vacuum tubes' in the United States, operate very many times faster than relays, as the valve's only moving part is a beam of electrons.

During the attack on Enigma, Bletchley Park approached the Post Office Research Station at Dollis Hill in London to build a relay-based machine for use in conjunction with the Bombe. Once the Bombe had uncovered the Enigma settings for a given day, these settings were to be transferred to the proposed machine, which would then decipher messages and print out the original German text.[16] Dollis Hill sent electronic engineer Thomas Flowers to Bletchley Park. In the end, the machine Flowers built was not used, but he was soon to become one of the great figures of Second World War codebreaking. Thanks to his pre-war research, Flowers was (as he himself remarked) possibly the only person in Britain who realized that valves could be used on a large scale for high-speed digital computing.[17]

The world's first large-scale electronic digital computer, Colossus, was designed and built during 1943 by Flowers and his team at Dollis Hill, in consultation with the Cambridge mathematician Max Newman, head of the section at Bletchley Park known simply as the 'Newmanry'. (Turing attended Newman's lectures on mathematical logic at Cambridge before the war; these lectures launched Turing on the research that led to his 'On Computable Numbers'.[18]) Colossus first worked in December 1943[19] (exactly two years before the first comparable US machine, the ENIAC, was operational[20]). Colossus was used against the Lorenz cipher machine, more advanced than Enigma and introduced in 1941.[21] The British government kept Colossus secret: before the 1970s few had any idea that electronic computation had been used successfully during the Second World War, and it was not until 2000 that Britain and the United States finally declassified the complete account of Colossus' wartime role.[22] So it was that, in the decades following the war, John von Neumann and others told the world that the ENIAC was 'the first electronic computing machine'.[23]

Although Colossus possessed a certain amount of flexibility, it was very far from universal. Nor did it store instructions internally. As with the ENIAC, in order to set Colossus up for a new job it was necessary to modify by hand some of the machine's wiring, by means of switches and plugs. (During the construction of Colossus, Newman showed Flowers Turing's 'On Computable Numbers', with its key idea of storing coded instructions in memory, but Flowers, not being a mathematical logician, 'didn't really understand much of it'.[24]) Nevertheless, Flowers established decisively and for the first time that large-scale electronic computing machinery was practicable.

Flowers said that, once Turing saw Colossus in operation, it was just a matter of Turing's waiting to see what opportunity might arise to put the idea of his universal computing machine into practice.[25] There is little doubt that by 1944 Newman too had firmly in mind the possibility of building a universal machine using electronic technology. In February 1946, a few months after his appointment as Professor of Mathematics at the University of Manchester, Newman wrote to von Neumann in the United States:

> I am ... hoping to embark on a computing machine section here, having got very interested in electronic devices of this kind during the last two or three years. By about eighteen months ago I had decided to try my hand at starting up a machine unit when I got out. ... I am of course in close touch with Turing.[26]

Turing's own opportunity came when Womersley appeared out of the blue to recruit him to the National Physical Laboratory (see Chapter 3). By then Turing had educated himself in electronic engineering (during the later part of the war he gave a series of evening lectures 'on valve theory').[27]

The ACE and the EDVAC

In the years immediately following the Second World War, the Hungarian American logician and mathematician John von Neumann, through writings and charismatic public addresses, made the concept of the stored-program digital computer widely known. Von Neumann wrote 'First Draft of a Report on the EDVAC' (see the Introduction) and subsequently directed the computer project at the Princeton Institute for Advanced Study. The ensuing machine, the IAS computer, although not the first to run in the United States (it began work in the summer of 1951[28]), was the most influential of the early US computers and the precursor to the IBM 701, the company's first mass-produced stored-program electronic computer.

Von Neumann's 'First Draft of a Report on the EDVAC' was widely read and was used as a blueprint by, among others, Wilkes, whose EDSAC at the University of Cambridge was the second stored-program electronic computer to function (in 1949). Turing certainly expected his readers to be familiar with the 'First Draft'. At the end of the first section of 'Proposed Electronic Calculator' he said:

> The present report gives a fairly complete account of the proposed calculator. It is recommended however that it be read in conjunction with J. von Neumann's '[First Draft of a] Report on the EDVAC'.

To what extent was the content of 'Proposed Electronic Calculator' influenced by the 'First Draft'? The former follows von Neumann's terminology and notation to some extent. This decision was a sensible one in the circumstances, making it likely that Turing's report would be more readily understood. In order to depict the EDVAC's logic gates, von Neumann had used a modified version of a diagrammatic notation introduced by McCulloch and Pitts in connection with neural nets.[29] Turing adopted this modified notation and in fact considerably extended it.[30] There is no doubt that Turing simply borrowed some of the more elementary material from the 'First Draft'. (For example, his diagram of an adder—figure 10 of 'Proposed Electronic Calculator'—is essentially the same as von Neumann's figure 3.[31] A newspaper report of 1946 stated that Turing 'gives credit for the donkey work on the A.C.E. to Americans'.[32]) However, Turing's logic diagrams set out detailed designs for the logical control and the arithmetic part of the calculator and go far beyond anything to be found in the 'First Draft'. The similarities between 'Proposed Electronic Calculator' and the 'First Draft' are relatively minor in comparison to the striking differences in the designs that they contain (see the Introduction and Chapter 8 'Computer Architecture and the ACE Computers'). Moreover, von Neumann's minor influence on 'Proposed Electronic Calculator' should not be allowed to mask the extent to which Turing's universal machine of 1936 was itself a fundamental influence upon von Neumann (see below).

Notoriously, the universal machine of 1936 received no explicit mention in 'Proposed Electronic Calculator'. In his chapter 'The ACE and the Shaping of British Computing' Campbell-Kelly raises the question whether the universal machine was a 'direct ancestor' of the ACE at all, emphasizing that the ACE's 'addressable memory of fixed-length binary numbers had no equivalent in the Turing Machine'. However, Turing's previously unpublished notes on memory (in Part V) cast light on this issue. The notes are fragments of a draft of 'Proposed Electronic Calculator'; in them Turing related the ACE to the universal Turing machine, explaining why the memory arrangement described in 'On Computable Numbers' could not 'be taken over as it stood to give a practical form of machine'.

Turing regarded the ACE as a 'practical version' of the UTM:

> Some years ago I was researching on what might now be described as an investigation of the theoretical possibilities and limitations of digital computing machines. I considered a type of machine which had a central mechanism, and an infinite memory which was contained on an infinite tape . . . It can be shown that a single special machine of that type can be made to do the work of all. . . . The special machine may be called the universal machine; it works in the following quite simple manner. When we have decided what machine we wish to imitate we punch a description of it on the tape of the universal machine. . . . The universal machine has only to keep looking at this description in order to find out what it should do at each stage. Thus the complexity of the machine to be imitated is concentrated in the tape and does not appear in the universal machine proper in any way. . . . [D]igital computing machines such as the ACE . . . are in fact practical versions of the universal machine. There is a certain central pool of electronic equipment, and a large memory. When any particular problem has to be handled the appropriate instructions for the computing process involved are stored in the memory of the ACE . . .[33]

A letter from Turing to the cyberneticist W. Ross Ashby again highlights the fundamental point of similarity between the ACE and the UTM:

> The ACE is in fact, analogous to the 'universal machine' described in my paper on conputable [sic] numbers . . . [W]ithout altering the design of the machine itself, it can, in theory at any rate, be used as a model of any other machine, by making it remember a suitable set of instructions.[34]

Turing's influence on von Neumann

In the secondary literature, von Neumann is often said to have invented the stored-program computer, but he repeatedly emphasized that the fundamental conception was Turing's. Von Neumann became familiar with ideas in 'On Computable Numbers' during Turing's time at Princeton (1936–8) and was to become intrigued by Turing's concept of a universal computing machine.[35] It was von Neumann who placed Turing's concept into the hands of American engineers. Stanley Frankel (the Los Alamos physicist responsible, with von Neumann and others, for mechanizing the large-scale calculations involved in the design of the atomic and hydrogen bombs) recorded von Neumann's view of the importance of 'On Computable Numbers':

> I know that in or about 1943 or '44 von Neumann was well aware of the fundamental importance of Turing's paper of 1936 'On computable numbers ...', which describes in principle the 'Universal Computer' of which every modern computer (perhaps not ENIAC as first completed but certainly all later ones) is a realization. Von Neumann introduced me to that paper and at his urging I studied it with care. Many people have acclaimed von Neumann as the 'father of the computer' (in a modern sense of the term) but I am sure that he would never have made that mistake himself. He might well be called the midwife, perhaps, but he firmly emphasized to me, and to others I am sure, that the fundamental conception is owing to Turing—insofar as not anticipated by Babbage, Lovelace, and others. In my view von Neumann's essential role was in making the world aware of these fundamental concepts introduced by Turing and of the development work carried out in the Moore school and elsewhere.[36]

In 1944, von Neumann joined the Eckert–Mauchly ENIAC group at the Moore School of Electrical Engineering at the University of Pennsylvania. (At that time he was involved in the Manhattan Project at Los Alamos, where roomfuls of clerks armed with desk calculating machines were struggling to carry out the massive calculations required by the physicists.) ENIAC—which had been under construction since 1943—was, as mentioned above, not a stored-program computer: programming consisted of re-routing cables and setting switches. Moreover, the ENIAC was far from universal, having been designed with only one very specific task in mind, the calculation of trajectories of artillery shells. Von Neumann brought his knowledge of 'On Computable Numbers' to the practical arena of the Moore School. Thanks to Turing's abstract logical work, von Neumann knew that, by making use of coded instructions stored in memory, a single machine of fixed structure can in principle carry out any task for which a program can be written. When Eckert explained his idea of using the mercury delay line as a high-speed recirculating memory, von Neumann saw that this was the means to make concrete the abstract universal computing machine of 'On Computable Numbers'.[37]

When, in 1946, von Neumann established his own project to build a stored-program computer at the Princeton Institute for Advanced Study, he gave his engineers ‘On Computable Numbers’ to read.[38] Bigelow, von Neumann’s chief engineer and largely responsible for the engineering design of the computer built at the Institute, said:

> The person who really ... pushed the whole field ahead was von Neumann, because he understood logically what [the stored-program concept] meant in a deeper way than anybody else. ... The reason he understood it is because, among other things, he understood a good deal of the mathematical logic which was implied by the idea, due to the work of A. M. Turing ... in 1936–1937 .... Turing’s [universal] machine does not sound much like a modern computer today, but nevertheless it was. It was the germinal idea. ... So ... [von Neumann] saw ... that [the ENIAC] was just the first step, and that great improvement would come.[39]

Von Neumann repeatedly emphasized the fundamental importance of ‘On Computable Numbers’ in lectures and in correspondence. In 1946 he wrote to the mathematician Norbert Wiener of ‘the great positive contribution of Turing’—Turing’s mathematical demonstration that ‘one, definite mechanism can be “universal” ’.[40] In 1948, in a lecture entitled ‘The General and Logical Theory of Automata’, von Neumann said:

> The English logician, Turing, about twelve years ago attacked the following problem. He wanted to give a general definition of what is meant by a computing automaton. ... Turing carried out a careful analysis of what mathematical processes can be effected by automata of this type. ... He ... also introduce[d] and analyse[d] the concept of a ‘universal automaton’ ... An automaton is ‘universal’ if any sequence that can be produced by any automaton at all can also be solved by this particular automaton. It will, of course, require in general a different instruction for this purpose. *The Main Result of the Turing Theory*. We might expect a priori that this is impossible. How can there be an automaton which is at least as effective as any conceivable automaton, including, for example, one of twice its size and complexity? Turing, nevertheless, proved that this is possible.[41]

The following year, in a lecture entitled ‘Rigorous Theories of Control and Information’, von Neumann said:

> The importance of Turing’s research is just this: that if you construct an automaton right, then any additional requirements about the automaton can be handled by sufficiently elaborate instructions.

> This is only true if [the automaton] is sufficiently complicated, if it has reached a certain minimal level of complexity. In other words . . . there is a very definite finite point where an automaton of this complexity can, when given suitable instructions, do anything that can be done by automata at all.[42]

Many books on the history of computing in the United States make no mention of Turing. No doubt this is in part explained by the absence of any explicit reference to Turing's work in the series of technical reports in which von Neumann, with various co-authors, set out a logical design for an electronic stored-program digital computer.[43] Nevertheless there is evidence in these documents of von Neumann's knowledge of 'On Computable Numbers'. For example, in the report entitled 'Preliminary Discussion of the Logical Design of an Electronic Computing Instrument' (1946), von Neumann and his co-authors, Burks and Goldstine—both former members of the ENIAC group, who had joined von Neumann at the Institute for Advanced Study—wrote the following:

> First Remarks on the Control and Code: 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 more of a 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 of those problems.[44]

Burks has confirmed that the first sentence of this passage is a reference to the UTM.[45] (The report was not intended for formal publication and no attempt was made to indicate those places where reference was being made to the work of others.)

The passage just quoted is an excellent summary of the situation at that time. In 'On Computable Numbers' Turing had shown in abstracto that, by means of instructions expressed in the programming code of his 'standard descriptions', a single machine of fixed structure is able to carry out any task that a 'problem planner' is able to analyse into effective steps. By 1945, considerations in abstracto had given way to the practical problem of devising an equivalent programming code that could be implemented efficiently by means of electronic circuits. Von Neumann's embryonic code appeared in the ‘First Draft’. ‘Proposed Electronic Calculator’ set out Turing’s own very different and much more fully developed code.

The Manchester computer

The first stored-program electronic computer, the Manchester ‘Baby’, came to life in June 1948 in Newman’s Computing Machine Laboratory at the University of Manchester.[46]

At the time of the Baby and its successor, the Manchester Mark I, the electronic engineers Williams and Kilburn, who had translated the logico-mathematical idea of the stored-program computer into hardware, were given too little credit by the mathematicians at Manchester—Williams and Kilburn were regarded as excellent engineers but not as ‘ideas men’.[47] Nowadays the tables have turned too far and the triumph at Manchester is usually credited to Williams and Kilburn alone. Fortunately the words of the late Freddie Williams survive to set the record straight:

> Now let’s be clear before we go any further that neither Tom Kilburn nor I knew the first thing about computers when we arrived in Manchester University . . . Newman explained the whole business of how a computer works to us.[48]

> Tom Kilburn and I knew nothing about computers . . . Professor Newman and Mr A. M. Turing . . . knew a lot about computers . . . They took us by the hand and explained how numbers could live in houses with addresses . . .[49]

In an address to the Royal Society on 4 March 1948, Newman presented this very explanation:

> In modern times the idea of a universal calculating machine was independently [of Babbage] introduced by Turing . . . There is provision for storing numbers, say in the scale of 2, so that each number appears as a row of, say, forty 0’s and 1’s in certain places or ‘houses’ in the machine. . . . Certain of these numbers, or ‘words’ are read, one after another, as orders. In one possible type of machine an order consists of four numbers, for example 11, 13, 27, 4. The number 4 signifies ‘add’, and when control shifts to this word the ‘houses’ $H11$ and $H13$ will be connected to the adder as inputs, and $H27$ as output. The numbers stored in $H11$ and $H13$ pass through the adder, are added, and the sum is passed on to $H27$. The control then shifts to

> the next order. In most real machines the process just described would be done by three separate orders, the first bringing ⟨*H*11⟩ (=content of *H*11) to a central accumulator, the second adding ⟨*H*13⟩ into the accumulator, and the third sending the result to *H*27; thus only one address would be required in each order. ... A machine with storage, with this automatic-telephone-exchange arrangement and with the necessary adders, subtractors and so on, is, in a sense, already a universal machine.[50]

Following this explanation of Turing's three-address concept (source 1, source 2, destination, function) Newman went on to describe program storage ('the orders shall be in a series of houses *X*1, *X*2, ...') and conditional branching. He then summed up:

> From this highly simplified account it emerges that the essential internal parts of the machine are, first, a storage for numbers (which may also be orders). ... Secondly, adders, multipliers, etc. Thirdly, an 'automatic telephone exchange' for selecting 'houses', connecting them to the arithmetic organ, and writing the answers in other prescribed houses. Finally, means of moving control at any stage to any chosen order, if a certain condition is satisfied, otherwise passing to the next order in the normal sequence. Besides these there must be ways of setting up the machine at the outset, and extracting the final answer in useable form.[51]

In a letter written in 1972 Williams described in some detail what he and Kilburn were told by Newman:

> About the middle of the year [1946] the possibility of an appointment at Manchester University arose and I had a talk with Professor Newman who was already interested in the possibility of developing computers and had acquired a grant from the Royal Society of £30,000 for this purpose. Since he understood computers and I understood electronics the possibilities of fruitful collaboration were obvious. I remember Newman giving us a few lectures in which he outlined the organisation of a computer in terms of numbers being identified by the address of the house in which they were placed and in terms of numbers being transferred from this address, one at a time, to an accumulator where each entering number was added to what was already there. At any time the number in the accumulator could be transferred back to an assigned address in the store and the accumulator cleared for further use. The transfers were to

> be effected by a stored program in which a list of instructions was obeyed sequentially. Ordered progress through the list could be interrupted by a test instruction which examined the sign of the number in the accumulator. Thereafter operation started from a new point in the list of instructions. This was the first information I received about the organisation of computers. . . . Our first computer was the simplest embodiment of these principles, with the sole difference that it used a subtracting rather than an adding accumulator.[52]

Turing's early input to the developments at Manchester, hinted at by Williams in his above-quoted reference to Turing, may have been via the lectures on computer design that Turing and Wilkinson gave in London during the period December 1946 to February 1947 (see Chapter 22, 'The Turing–Wilkinson Lecture Series'). The lectures were attended by representatives of various organizations planning to use or build an electronic computer. Kilburn was in the audience.[53] (Kilburn usually said, when asked from where he obtained his basic knowledge of the computer, that he could not remember;[54] e.g., in a 1992 interview he said: 'Between early 1945 and early 1947, in that period, somehow or other I knew what a digital computer was . . . Where I got this knowledge from I've no idea'.[55])

Whatever role Turing's lectures may have played in informing Kilburn, there is little doubt that credit for the Manchester computer—called the 'Newman–Williams machine' by Huskey in a report written shortly after his visit in 1947 to the Manchester project (see Chapter 23, 'The State of the Art in Electronic Digital Computing in Britain and the United States')—belongs not only to Williams and Kilburn but also to Newman, and that the influence on Newman of Turing's 1936 paper was crucial (as was the influence of Flowers' Colossus).

The Manchester computer and the EDVAC

The Baby and the Manchester Mark I are sometimes said to have descended from the EDVAC (see, e.g., the American *Family Tree of Computer Design*, fig. 1 in Chapter 6). Newman was well aware of von Neumann's 'First Draft of a Report on the EDVAC'. In the summer of 1946 he sent David Rees, a lecturer in his department at Manchester and an ex-member of the Newmanry (Newman's section at Bletchley Park), to a series of lectures at the Moore School, where Eckert, Mauchly, and other members of the ENIAC–EDVAC group publicized their ideas on computer design.[56] In the autumn of 1946 Newman himself went to Princeton for three months.[57]

Newman's advocacy of 'a central accumulator'—a characteristic feature of the EDVAC but not of the ACE—was probably influenced by his knowledge of the American proposals. However, von Neumann's ideas seem to have had little influence on other members of the Manchester project. Kilburn spoke scathingly of the von Neumann 'dictat'.[58] Tootill said:

> Williams, Kilburn and I (the three designers of the first Manchester machine) had all spent the 1939–1945 war at the Telecommunications Research Establishment doing R & D on radiolocation equipments. The main U.S. ideas that we accepted in return for our initiatives on these and later on computers were the terms 'radar' and 'memory' . . . We disliked the latter term, incidentally, as encouraging the anthropomorphic concept of 'machines that think'.[59]

> To the best of my recollection FC [Williams], Tom [Kilburn] and I never discussed . . . von Neumann's . . . ideas during the development of the Small-Scale Experimental Machine [the Baby], nor did I have any knowledge of them when I designed the Ferranti Mk I. I don't think FC was influenced at all by von Neumann, because I think he was in general quite punctilious in acknowledging other people's ideas.[60]

Tootill added:

> As well as our own ideas, we incorporated functions suggested by Turing and Newman in the improvement and extension of the first machine. When I did the logic design of the Ferranti Mark I, I got them to approve the list of functions.[61]

Turing joins the Manchester project

In May 1948 Turing resigned from the NPL. Work on the ACE had drawn almost to a standstill (see Chapter 3, 'The Origins and Development of the ACE Project'). Newman lured a 'very fed up'[62] Turing to Manchester, where in May 1948 he was appointed Deputy Director of the Computing Machine Laboratory (there being no Director). Turing designed the input mechanism and programming system[63] of, and wrote a programming manual[64] for, the full-scale Manchester computer. (The first of the production models, marketed by Ferranti, was completed in February 1951 and was the first commercially available electronic digital computer.[65] The first US commercial machine, the Eckert-Mauchly UNIVAC, appeared a few months later.) At last Turing had his hands on a stored-program computer.

Artificial Intelligence

The myth

Artificial Intelligence (AI) is often said to have been born in the mid-1950s in the United States. For example:

> Artificial Intelligence, conceived at Carnegie Tech in the autumn of 1955, quickened by Christmas, and delivered on Johnniac in the spring, made a stunning debut at the conference from which it later took its name.[66]

The AI program 'delivered on Johnniac' (a Californian copy of the IAS computer) was the Logic Theorist, written by Newell, Simon, and Shaw and demonstrated at a conference, the Dartmouth Summer Research Project on Artificial Intelligence, held at Dartmouth College, New Hampshire.[67] The Logic Theorist was designed to prove theorems from Whitehead and Russell's *Principia Mathematica*.[68] (In one case the proof devised by the Logic Theorist was several lines shorter than the one given by Whitehead and Russell; Newell, Simon, and Shaw wrote up the proof and sent it to the *Journal of Symbolic Logic*. This was almost certainly the first paper to have a computer listed as a co-author, but unfortunately it was rejected.[69])

The reality

In Britain the term 'machine intelligence' pre-dated 'artificial intelligence', and the field of enquiry itself can be traced much further back than 1955. If anywhere has a claim to be the birthplace of AI, it is Bletchley Park. Turing was the first to carry out substantial research in the area. At least as early as 1941 he was thinking about machine intelligence—in particular the possibility of computing machines that solved problems by means of searching through the space of possible solutions, guided by what would now be called 'heuristic' principles—and about the mechanization of chess.[70] At Bletchley Park, in his spare time, Turing discussed these topics and also machine learning. He circulated a typescript concerning machine intelligence among some of his colleagues.[71] Now lost, this was undoubtedly the earliest paper in the field of AI.

The first AI programs ran in Britain in 1951–2, at Manchester and Cambridge. This was due in part to the fact that the first stored-program electronic computers ran in Britain and in part to Turing's influence on the first generation of computer programmers. (Even in the United States, the

Logic Theorist was not the first AI program to run. Arthur Samuel's Checkers (or Draughts) Player first ran at the end of 1952 on the IBM 701, IBM's first stored-program electronic digital computer.[72] In 1955 Samuel added learning to the program.)

The Bombe

The Bombe is the first milestone in the history of machine intelligence.[73] Central to the Bombe was the idea of solving a problem by means of a guided mechanical search through the space of possible solutions. In this instance, the space of possible solutions consisted of configurations of the Enigma machine (in another case it might consist of configurations of a chess board). The Bombe's search could be guided in various ways; one involved what Turing called the 'multiple encipherment condition' associated with a crib (described in Chapter 6 of Turing's recently declassified *Treatise on the Enigma*, written in the second half of 1940).[74] A search guided in this fashion, Turing said, would 'reduce the possible positions to a number which can be tested by hand methods'.[75] (A crib is a word or phrase that the cryptanalyst believes might be part of the German message. For example, it might be conjectured that a certain message contains 'WETTER FUR DIE NACHT' (weather for the night). Many Enigma networks were good sources of cribs, thanks both to the stereotyped nature of German military messages and to lapses of cipher security. One station sent exactly the same message ('beacons lit as ordered') each evening for a period of several months.[76])

Modern AI researchers speak of the method of *generate-and-test*. Potential solutions to a given problem are generated by means of a guided search. These potential solutions are then tested by an auxiliary method to find out if any is actually a solution. Nowadays in AI both processes, generate and test, are typically carried out by the same program. The Bombe mechanized the first process. The testing of the potential solutions (the 'stops') was then carried out manually (by setting up a replica Enigma accordingly, typing in the cipher text, and seeing whether or not German words emerged).

Machine intelligence, 1945–8

In designing the ACE, machine intelligence was not far from Turing's thoughts—he described himself as building 'a brain'[77] and declared 'In working on the ACE I am more interested in the possibility of producing models of the action of the brain than in the practical applications to computing'.[78] In 'Proposed Electronic Calculator' he said:

> 'Can the machine play chess?' It could fairly easily be made to play a rather bad game. It would be bad because chess requires intelligence. We stated at the beginning of this section that the machine should be treated as entirely without intelligence. There are indications however that it is possible to make the machine display intelligence at the risk of its making occasional serious mistakes. By following up this aspect the machine could probably be made to play very good chess.

Turing's point was probably that the use of heuristic search brings with it the risk of the machine's sometimes making mistakes.

In February 1947 (in the rooms of the Royal Astronomical Society in Burlington House, London[79]) Turing gave what is, so far as is known, the earliest public lecture to mention computer intelligence, providing a breathtaking glimpse of a new field.[80] He described the human brain as a 'digital computing machine'[81] and discussed the prospect of machines that act intelligently, learn, and beat human opponents at chess. He stated that '[w]hat we want is a machine that can learn from experience' and that '[t]he possibility of letting the machine alter its own instructions provides the mechanism for this'.[82] (The possibility of a computer's operating on and modifying its own program as it runs, just as it operates on the data in its memory, is implicit in the stored-program concept.) At the end of the 1947 lecture Turing set out what he later called the 'Mathematical Objection' to the hypothesis of machine intelligence. This is now widely known as the Gödel argument, and has been made famous by John Lucas and Roger Penrose. (In fact the objection originated with the mathematical logician Emil Post, as early as 1921.[83]) Turing proposed an interesting and arguably correct solution to the objection.[84]

In mid-1947, with little progress on the physical construction of the ACE, a thoroughly disheartened Turing applied for a twelve-month period of sabbatical leave to be spent in Cambridge. The purpose of the leave, as described by Darwin in July 1947, was to enable Turing

> to extend his work on the [ACE] still further towards the biological side. I can best describe it by saying that hitherto the machine has been planned for work equivalent to that of the lower parts of the brain, and [Turing] wants to see how much a machine can do for the higher ones; for example, could a machine be made that could learn

> by experience? This will be theoretical work, and better done away from here.[85]

Turing left the NPL for Cambridge in the autumn of 1947.[86]

In the summer of 1948 Turing completed a report describing the outcomes of this research. It was entitled 'Intelligent Machinery'.[87] Donald Michie recalls that Turing

> was in a state of some agitation about its reception by his superiors at N.P.L.: 'A bit thin for a year's time off!'.[88]

The headmasterly Darwin—who once complained about the 'smudgy' appearance of Turing's work[89]—was, as Turing predicted, displeased with 'Intelligent Machinery', describing it as a 'schoolboy's essay'[90] and 'not suitable for publication'.[91] In reality this far-sighted paper was the first manifesto of Artificial Intelligence; sadly Turing never published it.

'Intelligent Machinery' is a wide-ranging and strikingly original survey of the prospects of AI. In it Turing brilliantly introduced a number of the concepts that were later to become central in AI, in some cases after reinvention by others. These included the logic-based approach to problem-solving, now widely used in expert systems, and, in a brief passage concerning what he called 'genetical or evolutionary search'[92], the concept of a genetic algorithm—important in both AI and Artificial Life. (The term 'genetic algorithm' was introduced in *c.*1975.[93]) In the light of his work with the Bombe, it is not surprising to find Turing hypothesizing in 'Intelligent Machinery' that 'intellectual activity consists mainly of various kinds of search'.[94] Eight years later the same hypothesis was put forward independently by Newell and Simon and through their influential work[95] became one of the principal tenets of AI. 'Intelligent Machinery' also contains the earliest description of (a restricted form of) what Turing was later to call the 'imitation game' and is now known simply as the Turing test.[96]

The first AI programs

Both during and after the war Turing experimented with machine routines for playing chess: in the absence of a computer, the machine's behaviour was simulated by hand, using paper and pencil. In 1948 Turing and David Champernowne, the mathematical economist, constructed the loose system of rules dubbed the 'Turochamp'.[97] (Champernowne reported that his wife, a beginner at chess, took on the Turochamp and lost.) Turing began to program the Turochamp for the Manchester Ferranti Mark I but unfortunately never completed the task.[98] He later published a classic early article on chess programming.[99] Dietrich Prinz, who worked for Ferranti, wrote the first chess program to be implemented.[100] It ran in November 1951 on the Ferranti Mark I.[101] Unlike the Turochamp, Prinz's program could not play a complete game and operated by exhaustive search rather than under the guidance of heuristics. (Prinz 'learned all about programming the Mark I computer at seminars given by Alan Turing and Cecily Popplewell'.[102] Like Turing, he wrote a programming manual for the Mark I.[103] Prinz also used the Mark I to solve logical problems, and in 1949 and 1951 Ferranti built two small experimental special-purpose computers for theorem-proving and other logical work.[104])

Christopher Strachey's Draughts Player was—apart from Turing's 'paper' chess-players—the first AI program to use heuristic search. He coded it for the Pilot Model ACE in May 1951.[105] Strachey's first attempt to get his program running on the Pilot ACE was defeated by coding errors. When he returned to the NPL with a debugged version of the program, he found that a major hardware change had been made, with the result that the program would not run without substantial revision.[106] He finally got his program working on the Ferranti Mark I in mid-1952 (with Turing's encouragement and utilizing the latter's recently completed *Programmer's Handbook*[107]).[108] By the summer of 1952 the program could play a complete game of draughts at a reasonable speed.[109] The essentials of Strachey's program were taken over by Samuel in the United States.[110]

The first AI programs to incorporate learning, written by Anthony Oettinger at the University of Cambridge, ran in 1951.[111] Oettinger wrote his 'response learning programme' and 'shopping programme' for the Cambridge EDSAC computer. Oettinger was considerably influenced by Turing's views on machine learning,[112] and suggested that the shopping program—which simulated the behaviour of 'a small child sent on a shopping tour'[113]—could pass a version of the Turing test in which 'the questions are restricted to . . . the form "In what shop may article *j* be found?"'[114].

Turing's unorganized computing machines

Turing did not only invent the concept of the stored-program digital computer; he also pioneered the idea of computing by neural networks. The major part of 'Intelligent Machinery' is a discussion of (what Turing called) 'unorganised machines'.[115,116] His unorganized computing machines have been ignored by historians of the computer and merit a detailed introduction.

Turing described three types of unorganized machine. A-type and B-type unorganized machines consist of randomly connected two-state 'neurons' whose operation is synchronized by means of a central digital clock. We call these networks 'Turing Nets'. Turing's P-type unorganized machines are not neuron-like but are modified Turing machines: they have 'only two interfering inputs, one for "pleasure" or "reward" ... and the other for "pain" or "punishment" '.[117] Turing studied P-types in the hope of discovering procedures for 'training' a machine to carry out a task. (It is a P-type machine that Turing was speaking of when, in the course of his famous discussion of strategies for building machines to pass the Turing test, he said 'I have done some experiments with one such child-machine, and succeeded in teaching it a few things'.[118])

Turing had no doubts concerning the significance of his unorganized machines. Of Turing Nets, he said

> [M]achines of this character can behave in a very complicated manner when the number of units is large ... A-type unorganised machines are of interest as being about the simplest model of a nervous system with a random arrangement of neurons. It would therefore be of very great interest to find out something about their behaviour.[119]

He theorized that 'the cortex of the infant is an unorganised machine, which can be organised by suitable interfering training'.[120] Turing found 'this picture of the cortex as an unorganised machine ... very satisfactory from the point of view of evolution and genetics'.[121]

A-type unorganized machines

Turing introduced the idea of an unorganized machine by means of an example:

> A typical example of an unorganised machine would be as follows. The machine is made up from a rather large number N of similar units. Each unit has two input terminals, and has an output terminal which can be connected to the input terminals of (0 or more) other units. We may imagine that for each integer r, $1 \leq r \leq N$, two numbers $i(r)$ and $j(r)$ are chosen at random from $1 \ldots N$ and that we connect the inputs of unit r to the outputs of units $i(r)$ and $j(r)$. All of the units are connected to a central synchronising unit from which synchronising

> pulses are emitted at more or less equal intervals of time. The times when these pulses arrive will be called 'moments'. Each unit is capable of having two states at each moment. These states may be called 0 and 1.[122]

Turing then gave (what would now be called) a propagation rule and an activation rule for the network. A propagation rule calculates the net input into a unit, and an activation rule calculates the new state of a unit, given its net input. The propagation rule is:

> The net input into unit *r* at moment *m*, net(*r*, *m*), is the product of the state of *i*(*r*) at *m*-1 and the state of *j*(*r*) at *m*-1.

The activation rule is:

> The state of *r* at *m* is 1-net(*r*, *m*).

A network of the sort that Turing described in the above quotation and whose behaviour is determined by these two rules is an A-type unorganized machine.

In modern terminology an A-type machine is a collection of NAND units. The propagation rule in effect takes the conjunction of the values on the unit's two input lines, and the activation rule forms the negation of this value. Alternative choices of propagation rule and/or activation rule will cause the units to perform other Boolean operations. As is well known, NAND is a fundamental operation in the sense that any Boolean operation can be performed by a circuit consisting entirely of NAND units. Thus any such operation can be performed by an A-type machine.[123]

B-type unorganized machines

The most significant aspect of Turing's discussion of unorganized machines is undoubtedly his idea that an initially random network can be organized to perform a specified task by means of what he described as 'interfering training'.[124]

> Many unorganised machines have configurations such that if once that configuration is reached, and if the interference thereafter is appropriately restricted, the machine behaves as one organised for some definite purpose.[125]

Turing illustrated this idea by means of the circuit shown in fig. 1.[126] (He stressed that this particular circuit is employed 'for illustrative purposes'

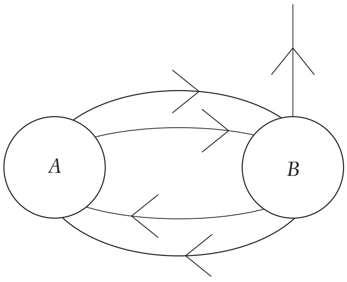

Fig. 1 Introverted pair.

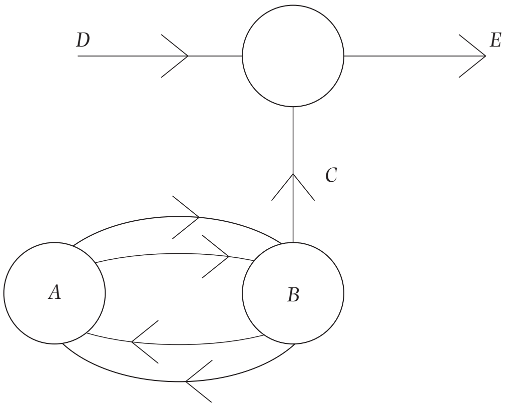

Fig. 2 Connection-modifier.

and not because it is 'of any great intrinsic importance'.[127]) We call a pair of units (*A* and *B*) connected in the way shown an 'introverted pair'. By means of external interference the state of unit *A* may be set to either 0 or 1; this state will be referred to as the 'determining condition' of the pair. The signal produced in unit *B*'s free output connection is constant from moment to moment and the polarity of the signal depends only upon the determining condition of the pair. Thus an introverted pair functions as an elementary memory.[128]

Turing defined B-type machines in terms of a certain process of substitution applied to A-type machines: a B-type results if every unit-to-unit connection within an A-type machine is replaced by the device shown in fig. 2.[129] That is to say, what is in the A-type a simple connection between points *D* and *E* now passes via the depicted device.[130]

Depending on the polarity of the constant signal at *C*, the signal at *E* is either 1 if the signal at *D* is 0 and 0 if the signal at *D* is 1, or always 1 no matter what the signal at *D*. In the first of these cases the device functions as a negation module. In the second case the device in effect disables the connection to which it is attached. That is to say, a unit with the device attached to one of its input connections delivers an output that is a function only of the signal arriving along its other input connection. (If the devices on both the unit's input connections are placed in disable mode then the unit's output is always 0.) Using these devices an external agent can organize an initially random B-type machine, selectively disabling and enabling connections within it.

Turing claimed that it is a 'general property of B-type machines . . . that with suitable initial [i.e. determining] conditions they will do any required job, given sufficient time and provided the number of units is sufficient'.[131] This follows from the more specific claim that given 'a B-type unorganised machine with sufficient units one can find initial conditions which will make it into a universal [Turing] machine with a given storage capacity'.[132] This claim first opened up the possibility, noted by Turing,[133] that the human brain is (in part) a UTM implemented in a neural network.

B-types redefined

Concerning his claim that one can configure a B-type network (with sufficient units) such that it is a UTM with a finite tape, Turing remarked: 'A formal proof to this effect might be of some interest, or even a demonstration of it starting with a particular unorganised B-type machine, but I am not giving it as it lies rather too far outside the main argument'.[134] It is unfortunate that Turing did not give any details of the proof, for this might have cast some light on what appears to be an inconsistency in 'Intelligent Machinery'. It is reasonably obvious that not all Boolean functions can be computed by B-type machines as defined. (A good way to get a feel for the difficulty is to attempt to design a B-type circuit for computing XOR—exclusive disjunction.)

'Intelligent Machinery' contains no clues as to Turing's own procedure for dealing with this problem. The simplest remedy seems to be to modify the substitution in terms of which B-type machines are defined: a B-type results if every unit-to-unit connection within an A-type machine is replaced by *two* of the devices shown in fig. 2, linked in series. That is to say, what is in the A-type a simple connection between two units now passes through two additional units each with its own introverted pair attached. It is trivially the case that if a function can be computed by some A-type machine then it can also be computed by some machine satisfying the modified definition of a B-type.

Turing's anticipation of connectionism

So far as we have been able to discover, Turing was the first person to consider building computing machines out of simple, neuron-like elements connected together into networks in a largely random manner. His account of B-types anticipated the modern approach to AI known as connectionism (i.e. computation by neural networks).[135] Rosenblatt—the inventor of the type of neural net called the 'perceptron'—introduced the term 'connectionist' in the following way:

> [According to] theorists in the empiricist tradition ... the stored information takes the form of new connections, or transmission channels in the nervous system (or the creation of conditions which are functionally equivalent to new connections) ... The theory to be presented here takes the empiricist, or 'connectionist' position.[136]

Turing's arrangement of selectively disabling and enabling connections within a B-type machine is functionally equivalent to one in which the stored information takes the form of new connections within the network.

Turing also envisaged the procedure—nowadays used extensively by connectionists—of programming training algorithms into a computer simulation of the unorganized machine. In modern architectures repeated applications of a training algorithm (e.g. the 'back propagation' algorithm) cause the encoding of the problem solution to develop gradually within the network during the training phase. Turing had no algorithm for training his B-types.[137] He regarded the development of training algorithms for unorganized machines as a central problem. With characteristic farsightedness Turing ended his discussion of unorganized machines by sketching the research programme that connectionists are now pursuing:

> I feel that more should be done on these lines. I would like to investigate other types of unorganised machines ... When some electronic machines are in actual operation I hope that they will make this more feasible. It should be easy to make a model of any particular machine that one wishes to work on within such a UPCM [universal practical computing machine] instead of having to work with a paper machine as at present. If also one decided on quite definite 'teaching policies'

> these could also be programmed into the machine. One would then allow the whole system to run for an appreciable period, and then break in as a kind of 'inspector of schools' and see what progress had been made.[138]

As a result of his lukewarm interest in publication Turing's work on neuron-like computation remained unknown to others working in the area. Modern connectionists regard the work of Donald Hebb[139] and Frank Rosenblatt[140] as the foundation of their approach. Turing's unorganized machines were not mentioned by the other pioneers of neuron-like computation in Britain—Ross Ashby,[141] Beurle,[142] Taylor,[143] and Uttley.[144] The situation was identical on the other side of the Atlantic. Rosenblatt seemed not to have heard of Turing's unorganized machines.[145] Nor was Turing's work mentioned in Hebb's influential book *The Organization of Behavior*—the source of the so-called Hebbian approach to neural learning studied in connectionism today.[146] Modern discussions of the history of connectionism by Rumelhart, McClelland et al.[147] show no awareness of Turing's early contribution to the field.[148]

Turing himself was unable to pursue his research into unorganized machines very far. At the time, the only electronic stored-program computer in existence was the Manchester Baby. By the time Turing had access to the Ferranti Mark I, in 1951, his interests had shifted and he devoted his time to modelling biological growth. (It was not until 1954, the year of Turing's death, that Farley and Clark, working independently of Turing at MIT, succeeded in running the first computer simulation of a small neural network.[149]) In Mathematics Division Davies and Woodger pursued Turing's ideas on learning (see Chapter 15, 'The ACE Simulator and the Cybernetic Model'). Their Cybernetic Model, constructed in 1949, was a hardware simulation of six Boolean neurons. In a demonstration on BBC TV in 1950, the Cybernetic Model mimicked simple learning in an octopus.

McCulloch and Pitts

It is interesting that Turing made no reference in the 1948 report to the work of McCulloch and Pitts, itself influenced by his 'On Computable Numbers'. Their 1943 article represents the first attempt to apply what they refer to as 'the Turing definition of computability' to the study of neuronal function.[150] McCulloch stressed the extent to which his and Pitts' work was indebted to Turing in the course of some autobiographical remarks (made during the public discussion of a lecture by von Neumann in 1948):

> I started at entirely the wrong angle ... and it was not until I saw Turing's paper ['On Computable Numbers'] that I began to get going the right way around, and with Pitts' help formulated the required logical calculus. What we thought we were doing (and I think we succeeded fairly well) was treating the brain as a Turing machine.[151]

Like Turing, McCulloch and Pitts considered Boolean nets of simple two-state 'neurons'. In their 1943 article they showed that such a net augmented by an external tape can compute all (and only) numbers that can be computed by Turing machines; and that, without the external tape, some but not all of these numbers can be computed by nets. Unlike modern connectionists, but like Turing, McCulloch and Pitts made no use of weighted connections or variable thresholds. (Part of the burden of their argument is to show that the behaviour of a net of binary units with variable thresholds can be exactly mimicked by a simple Boolean net without thresholds 'provided the exact time for impulses to pass through the whole net is not crucial'.[152]) McCulloch and Pitts did not discuss universal machines.

Turing had unquestionably heard something of the work of McCulloch and Pitts. Von Neumann mentioned the McCulloch–Pitts article—albeit very briefly—in the 'First Draft of a Report on the EDVAC' and (as noted above) employed a modified version of their diagrammatic notation for neural nets. Wiener would almost certainly have mentioned McCulloch in the course of his 'talk over the fundamental ideas of cybernetics with Mr Turing' at the NPL in the spring of 1947.[153] (Wiener and McCulloch were founding members of the cybernetics movement.) Turing and McCulloch seem not to have met until 1949. After their meeting Turing spoke dismissively of McCulloch, referring to him as 'a charlatan'.[154] It is an open question whether the work of McCulloch and Pitts had any influence whatsoever on the development of the ideas presented in 'Intelligent Machinery'. (Max Newman remarked of Turing 'It was, perhaps, a defect of his qualities that he found it hard to use the work of others, preferring to work things out for himself'.[155])

Whatever the influences were on Turing at that time, there is no doubt that his work on neural nets goes importantly beyond the earlier work of McCulloch and Pitts. The latter gave only a perfunctory discussion of learning, saying no more than that the mechanisms supposedly underlying learning in the brain—they specifically mentioned threshold change and the formation of new synapses—can be mimicked by means of nets whose connections and thresholds remain unaltered.[156] Turing's idea of using supervised interference to train an initially random arrangement of units to compute a specified function was nowhere prefigured.

P-type unorganized machines

Turing's main purpose in studying P-type machines seems to have been to search for general training procedures. A P-type machine is a modified Turing machine. Chief among the modifications is the addition of two input lines, one for reward ('pleasure') and the other for punishment ('pain').[157] Unlike standard Turing machines a P-type has no tape. Initially a P-type machine is unorganized in the sense that its instruction table is 'largely incomplete'.[158] Application of either pleasure or pain by the trainer serves to alter an incomplete table to some successor table. After sufficient training a complete table may emerge.

The P-types that Turing explicitly considered have instruction tables consisting of three columns (unlike the four-column tables of 'On Computable Numbers' described above). An example (a simplified version of Turing's own[159]) is:

State	Control symbol	External action
1	U	A
2	D0	B
3	T1	B
4	U	A
5	D1	B

'U' means 'uncertain', 'T' means 'tentative', and 'D' means 'definite'. (The nature of the external actions A and B is not specified.) This table is incomplete in that no control symbol at all is specified for states 1 and 4 and the control symbol 1 has been entered only tentatively in the line for state 3. Only in the case of states 2 and 5 are definite control symbols listed. The table is complete only when a definite control symbol has been specified for each state.

The control symbol determines the state the machine is to go into once it has performed the specified external action. The rules that Turing gave governing the state transitions are:

1. If the control symbol is 1, either definitely or tentatively, then *next state* is the remainder of ((2 × *present state*) + 1) on division by the total number of states (in this case 5).

For example, if the machine is in state 3 then *next state* is 2.

2. If the control symbol is 0, either definitely or tentatively, then *next state* is the remainder of (2 × *present state*) on division by the total number of states.

For example, if the machine is in state 2 then *next state* is 4.

Let us suppose that the machine is set in motion in state 2. It performs the external action B, shifts to state 4, and performs the action A. No control symbol is specified in state 4. In this case the machine selects a binary digit at random, say 0, and replaces U by T0. The choice of control symbol determines the next state, in this case 3.

The trainer may apply a pleasure or pain stimulus at any time, with the effect that '[w]hen a pain stimulus occurs all tentative entries are cancelled, and when a pleasure stimulus occurs they are all made permanent'.[160] In other words, pleasure replaces every T in the table by D and pain replaces all occurrences of T0 and T1 by U.

Turing suggested that 'it is probably possible to organise these P-type machines into universal machines' but warned that this 'is not easy'.[161] He continued:

> If, however, we supply the P-type machine with a systematic external memory this organising becomes quite feasible. Such a memory could be provided in the form of a tape, and the [external actions] could include movement to right and left along the tape, and altering the symbol on the tape to 0 or 1 . . . I have succeeded in organising such a (paper) machine into a universal machine . . . This P-type machine with external memory has, it must be admitted, considerably more 'organisation' than say the A-type unorganised machine. Nevertheless the fact that it can be organised into a universal machine still remains interesting.[162]

As a search for 'teaching policies' Turing's experiments with P-types were not a great success. The method he used to train the P-type with external memory required considerable intelligence on the part of the trainer and he described it as ‘perhaps a little disappointing’, remarking that ‘[i]t is not sufficiently analogous to the kind of process by which a child would really be taught’.[163]

Artificial Life

In his final years Turing worked on (what since 1987 is called) Artificial Life (A-Life). The central aim of A-Life is a theoretical understanding of naturally occurring biological life—in particular of the most conspicuous feature of living matter, its ability to self-organize (i.e. to develop form and structure spontaneously). A-Life characteristically makes use of computers to simulate living and life-like systems. Langton, who coined the term ‘Artificial Life’, wrote

> Computers should be thought of as an important laboratory tool for the study of life, substituting for the array of incubators, culture dishes, microscopes, electrophoretic gels, pipettes, centrifuges, and other assorted wet-lab paraphernalia, one simple-to-master piece of experimental equipment.[164]

Turing was the first to use computer simulation to investigate a theory of ‘morphogenesis’—the development of organization and pattern in living things.[165] He began this investigation as soon as the first Ferranti Mark I to be produced was installed at Manchester University. He wrote in February 1951:

> Our new machine is to start arriving on Monday. I am hoping as one of the first jobs to do something about ‘chemical embryology’. In particular I think one can account for the appearance of Fibonacci numbers in connection with fir-cones.[166]

Shortly before the Ferranti computer arrived, Turing wrote about his work on morphogenesis in a letter to the biologist J. Z. Young.[167] The letter connects Turing’s work on morphogenesis with his interest in neural networks, and to some extent explains why he did not follow up his suggestion in ‘Intelligent Machinery’ and use the Ferranti computer to simulate his unorganized machines.

> I am afraid I am very far from the stage where I feel inclined to start asking any anatomical questions [about the brain]. According to my notions of how to set about it that will not occur until quite a late stage when I have a fairly definite theory about how things are done.

> At present I am not working on the problem at all, but on my mathematical theory of embryology . . . This is yielding to treatment, and it will so far as I can see, give satisfactory explanations of -
>
> i) Gastrulation.
> ii) Polyogonally symmetrical structures, e.g., starfish, flowers.
> iii) Leaf arrangement, in particular the way the Fibonacci series (0, 1, 1, 2, 3, 5, 8, 13, . . .) comes to be involved.
> iv) Colour patterns on animals, e.g., stripes, spots and dappling.
> v) Patterns on nearly spherical structures such as some Radiolaria, but this is more difficult and doubtful.
>
> I am really doing this now because it is yielding more easily to treatment. I think it is not altogether unconnected with the other problem. The brain structure has to be one which can be achieved by the genetical embryological mechanism, and I hope that this theory that I am now working on may make clearer what restrictions this really implies. What you tell me about growth of neurons under stimulation is very interesting in this connection. It suggests means by which the neurons might be made to grow so as to form a particular circuit, rather than to reach a particular place.

In June 1954, while in the midst of this groundbreaking work, Turing died. He left a large pile of handwritten notes concerning morphogenesis, and some programs.[168] This material is still not fully understood.

Notes

1. Turing, A. M. (1936) 'On computable numbers, with an application to the Entscheidungsproblem', *Proceedings of the London Mathematical Society*, Series 2, 42, (1936–7), 230–65. Reprinted in Copeland, B. J. (ed.) (2004) *The Essential Turing*. Oxford & New York: Oxford University Press.
2. Church, A. (1937) 'Review of Turing's "On computable numbers, with an application to the Entscheidungsproblem"', *Journal of Symbolic Logic*, 2, 42–3.
3. 'On computable numbers, with an application to the Entscheidungsproblem', p. 231.
4. Ibid., p. 233.
5. Newman in interview with Christopher Evans (*The Pioneers of Computing: An Oral History of Computing*. London: Science Museum); 'Dr. A. M. Turing', *The Times*, 16 June 1954, p. 10.

6. On the Church–Turing thesis, see Copeland, B. J. (1996) 'The Church–Turing thesis', in E. Zalta (ed.), *The Stanford Encyclopedia of Philosophy* <http://plato.stanford.edu>.
7. Turing, A. M. (1948) 'Intelligent machinery' (National Physical Laboratory Report, 1948), in Copeland (ed.) *The Essential Turing* (see note 1), p. 414. (A copy of the original typescript is in the Woodger Papers, National Museum of Science and Industry, Kensington, London; a digital facsimile is in The Turing Archive for the History of Computing <www.AlanTuring.net/intelligent_machinery>.)
8. Ibid., p. 414.
9. See, for example, White, I. (1988) 'The limits and capabilities of machines—a review', *IEEE Transactions on Systems, Man, and Cybernetics*, 18, 917–38.
10. Newell, A. (1980) 'Physical symbol systems', *Cognitive Science*, 4, 135–83; the quotation is from p. 150.
11. See further Copeland, B. J. (2002) 'Hypercomputation', *Minds and Machines*, 12, 461–502; Copeland, B. J. (2000) 'Narrow versus wide mechanism', *Journal of Philosophy*, 97, 5–32 (reprinted in Scheutz, M. (ed.) (2002) *Computationalism: New Directions*. Cambridge, MA: MIT Press).
12. For a full account of Turing's involvement with the Bletchley Park codebreaking operation, see Copeland (ed.) *The Essential Turing.*
13. A memo, 'Naval Enigma Situation', dated 1 November 1939 and signed by Knox, Twinn, Welchman, and Turing, said: 'A large 30 enigma bomb [sic] machine, adapted to use for cribs, is on order and parts are being made at the British Tabulating Company'. (The memo is in the British National Archives: Public Record Office (PRO), Kew, Richmond, Surrey; document reference HW 14/2.)
14. Mahon, P. 'The History of Hut Eight, 1939–1945' (June 1945), p. 28 (PRO document reference HW 25/2; a digital facsimile of the original typescript is in The Turing Archive for the History of Computing <www.AlanTuring.net/mahon_hut_8>).
15. Welchman, G. (2000) *The Hut Six Story: Breaking the Enigma Codes*, 2nd edn., Cleobury Mortimer: M&M Baldwin; 'Squadron-Leader Jones, Section' (PRO document reference HW 3/164; thanks to Ralph Erskine for sending a copy of this document).
16. Flowers in interview with Copeland (July 1998).
17. Flowers in interview with Copeland (July 1996).
18. Newman in interview with Evans (see note 5).
19. Flowers in interview with Copeland (July 1996).
20. Goldstine, H. (1972) *The Computer from Pascal to von Neumann*. Princeton: Princeton University Press, pp. 225–6.

21. For a history of Colossus see Copeland, B. J. 'Colossus and the Dawning of the Computer Age', in R. Erskine and M. Smith (eds) (2001) *Action This Day*. London: Bantam.
22. 'General Report on Tunny, with Emphasis on Statistical Methods' (PRO document reference HW 25/4, HW 25/5 (2 Vols)). This report was written in 1945 by Good, Michie, and Timms—members of Newman's section at Bletchley Park. A digital facsimile is in The Turing Archive for the History of Computing <www.AlanTuring.net/tunny_report>.
23. Von Neumann, J. (1954) 'The NORC and problems in high speed computing', in A. H. Taub (ed.) (1961) *Collected Works of John von Neumann*, Vol 5. Oxford: Pergamon Press; the quotation is from pp. 238–9.
24. Flowers in interview with Copeland (July 1996).
25. Ibid.
26. Letter from Newman to von Neumann, 8 February 1946 (in the von Neumann Archive at the Library of Congress, Washington, DC; a digital facsimile is in The Turing Archive for the History of Computing <www.AlanTuring.net/newman_vonneumann_8feb46>).
27. Turing, S. (1959) *Alan M. Turing*. Cambridge: W. Heffer, p. 74.
28. Bigelow, J. (1980) 'Computer development at the Institute for Advanced Study', in N. Metropolis, J. Howlett, and G. C. Rota (eds), *A History of Computing in the Twentieth Century*. New York: Academic Press, p. 308.
29. McCulloch, W. S. and Pitts, W. (1943) 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics*, 5, 115–33.
30. As noted by Hartree (1949) *Calculating Instruments and Machines*. Illinois: University of Illinois Press, pp. 97, 102.
31. Figure 3 is on p. 198 of the reprinting of the 'First Draft' in Stern, N. (1981) *From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchly Computers*. Bedford, MA: Digital Press.
32. *Evening News*, 23 December 1946. The cutting is among a number placed by Sara Turing in the Modern Archive Centre, King's College, Cambridge (catalogue reference K 5).
33. Turing, A. M. (1947) 'Lecture on the Automatic Computing Engine', in Copeland (ed.), *The Essential Turing* (see note 1); the quotation is from pp. 378, 383.
34. Turing in an undated letter to W. Ross Ashby (in the Woodger Papers (catalogue reference M11/99); a digital facsimile is in The Turing Archive for the History of Computing <www.AlanTuring.net/turing_ashby>).
35. 'I know that von Neumann was influenced by Turing . . . during his Princeton stay before the war', said Stanislaw Ulam (in interview with Evans in 1976, *The Pioneers of Computing: An Oral History of Computing*. London: Science Museum). When Ulam and von Neumann were touring in Europe during the summer of 1938, von Neumann devised a mathematical game involving

Turing-machine-like descriptions of numbers (Ulam reported by William Aspray (1990) in his *John von Neumann and the Origins of Modern Computing*. Cambridge, MA: MIT Press, pp. 178, 313). The word 'intrigued' is used in this connection by von Neumann's friend and colleague Herman Goldstine (*The Computer From Pascal to von Neumann,* p. 275 (see note 20)).

36. Letter from Frankel to Brian Randell, 1972 (first published in Randell (1972) 'On Alan Turing and the origins of digital computers' in B. Meltzer and D. Michie (eds) *Machine Intelligence* 7. Edinburgh: Edinburgh University Press. (Copeland is grateful to Randell for giving him a copy of the letter.)
37. Burks (a member of the ENIAC group) summarized matters thus in his 'From ENIAC to the stored-program computer: two revolutions in computers', in Metropolis, Howlett, and Rota (eds), *A History of Computing in the Twentieth Century.*

    > Pres [Eckert] and John [Mauchly] invented the circulating mercury delay line store, with enough capacity to store program information as well as data. Von Neumann created the first modern order code and worked out the logical design of an electronic computer to execute it. (p. 312)

    Burks also recorded (ibid., p. 341) that von Neumann was the first of the Moore School group to see the possibility, implicit in the stored-program concept, of allowing the computer to modify selected instructions in a program as it runs (e.g. in order to control loops and branching). The same idea lay at the foundation of Turing's theory of machine learning (see below).
38. Letter from Bigelow to Copeland, 12 April 2002. See also Aspray, *John von Neumann and the Origins of Modern Computing*, p. 178.
39. Bigelow in a tape-recorded interview made in 1971 by the Smithsonian Institution and released in 2002. (Copeland is grateful to Bigelow for sending a transcript of excerpts from the interview.)
40. Letter dated 29 November 1946 (in the von Neumann Archive at the Library of Congress, Washington, DC).
41. The text of 'The general and logical theory of automata' is in Taub (ed.), *Collected Works of John von Neumann*, Vol 5; the quotation is from pp. 313–14.
42. The text of 'Rigorous theories of control and information' is printed in von Neumann, J. (1966) *Theory of Self-Reproducing Automata*. Urbana: University of Illinois Press; the quotation is from p. 50.
43. The first papers in the series were the 'First Draft of a Report on the EDVAC', by von Neumann (1945), and 'Preliminary Discussion of the Logical Design of an Electronic Computing Instrument' by Burks, Goldstine, and von Neumann (1946).

44. Section 3.1 of Burks, A. W., Goldstine, H. H., and von Neumann, J. 'Preliminary Discussion of the Logical Design of an Electronic Computing Instrument', 28 June 1946, Institute for Advanced Study; reprinted in Taub (ed.), *Collected Works of John von Neumann*, Vol 5.
45. Letter from Burks to Copeland, 22 April 1998. See also Goldstine, *The Computer from Pascal to von Neumann*, p. 258.
46. Williams (1975) described the Computing Machine Laboratory on p. 328 of his 'Early computers at Manchester University', *The Radio and Electronic Engineer*, 45, 327–31:

    > It was one room in a Victorian building whose architectural features are best described as 'late lavatorial'. The walls were of brown glazed brick and the door was labelled 'Magnetism Room'.

47. Peter Hilton in interview with Copeland (June 2001).
48. Williams in interview with Evans in 1976 (*The Pioneers of Computing: An Oral History of Computing*. London: Science Museum).
49. Williams, 'Early Computers at Manchester University, p. 328.
50. Newman, M. H. A. (1948) 'General principles of the design of all-purpose computing machines', *Proceedings of the Royal Society of London*, Series A, 195, 271–4; the quotation is from pp. 271–2.
51. Ibid., pp. 273–4.
52. Letter from Williams to Randell, 1972 (in Randell 'On Alan Turing and the origins of digital computers', p. 9).
53. Bowker, G. and Giordano, R. (1993) 'Interview with Tom Kilburn', *Annals of the History of Computing,* 15, 17–32.
54. Letter from Brian Napper to Copeland, 16 June 2002.
55. Bowker and Giordano, 'Interview with Tom Kilburn', p. 19.
56. Letter from Rees to Copeland, 2 April 2001.
57. Newman, W. 'Max Newman: Mathematician, Codebreaker and Computer Pioneer', to appear.
58. Kilburn in interview with Copeland (July 1997).
59. Letter from Tootill to Copeland, 18 April 2001.
60. Letter from Tootill to Copeland, 16 May 2001.
61. Letter from Tootill to Copeland, 18 April 2001.
62. Gandy in interview with Copeland (November 1995).
63. Letter from Williams to Randell, 1972 (see note 52).
64. 'Programmers Handbook for Manchester Electronic Computer', Computing Machine Laboratory, University of Manchester (no date, *c*.1950); a digital facsimile is available in The Turing Archive for the History of Computing <www.AlanTuring.net/programmers_handbook>).

65. Lavington, S. H. (1975) *A History of Manchester Computers.* Manchester: NCC Publications, p. 20.
66. Haugeland, J. (1985) *Artificial Intelligence: The Very Idea.* Cambridge, MA: MIT Press, p. 176.
67. Newell, A., Shaw, J. C., and Simon, H. A. (1957) 'Empirical explorations with the logic theory machine: a case study in heuristics', *Proceedings of the Western Joint Computer Conference*, 15, 218–39 (reprinted in E. A. Feigenbaum and J. Feldman (eds) (1963) *Computers and Thought*. New York: McGraw-Hill).
68. Whitehead, A. N. and Russell, B. (1910) *Principia Mathematica*, Vol 1. Cambridge: Cambridge University Press.
69. Shaw in interview with Pamela McCorduck (McCorduck, P. (1979) *Machines Who Think*. New York: W. H. Freeman, p. 143).
70. Donald Michie in interview with Copeland (October 1995).
71. Donald Michie in interview with Copeland (February 1998).
72. Letter from Samuel to Copeland, 6 December 1988; Samuel, A. L. (1959) 'Some studies in machine learning using the game of checkers', *IBM Journal of Research and Development*, 3, 211–29 (reprinted in Feigenbaum and Feldman (eds), *Computers and Thought*).
73. We include in this claim the Polish Bomba, a more primitive form of the Bombe which also employed guided search (although cribs were not used). The Polish machine in effect used the heuristic 'Ignore the Stecker'. This heuristic was satisfactory during the period when the Enigma machine's stecker-board affected only 10–16 of the 26 letters of the alphabet (see Rejewski, M. (1980) 'Jak Matematycy polscy rozszyfrowali Enigme' [How the Polish mathematicians broke Enigma], *Annals of the Polish Mathematical Society, Series II: Mathematical News*, 23, 1–28; English translation in Kozaczuk, W. (1984) *Enigma: How the German Machine Cipher Was Broken, and How It Was Read by the Allies in World War Two*. London: Arms and Armour Press, trans. C. Kasparek).
74. The title 'Treatise on the Enigma' was probably added to Turing's document by a third party outside the Government Code and Cypher School, most likely in the United States. The copy of the otherwise untitled document held in the US National Archives and Records Administration (document reference RG 457, Historic Cryptographic Collection, Box 201, NR 964) is prefaced by a page typed some years later than the document itself; it is this page that bears the title 'Turing's Treatise on the Enigma'. Another copy of the document, held in the British Public Record Office (document reference HW 25/3), carries the title 'Mathematical theory of ENIGMA machine by A. M. Turing'; this too was possibly added at a later date. At Bletchley Park the document was referred to as 'Prof's Book'. (A digital facsimile of the PRO copy is in

The Turing Archive for the History of Computing <www.AlanTuring.net/profs_book>. Chapter 6 is printed in Copeland (ed.) *The Essential Turing* (see note 1).)

75. *The Essential Turing*, p. 317.
76. Mahon, 'The History of Hut Eight, 1939–1945', p. 40.
77. Turing, S. *Alan M. Turing*, p. 75; Don Bayley in interview with Copeland (December 1997).
78. Turing in a letter to W. Ross Ashby (see note 34).
79. Entry in Woodger's diary for 20 February 1947. (Copeland is grateful to Woodger for this information.)
80. 'Lecture on the Automatic Computing Engine' (see note 33).
81. Ibid., p. 382.
82. Ibid., p. 393.
83. Post, E. L. 'Absolutely unsolvable problems and relatively undecidable propositions: account of an anticipation', in M. Davis (ed.) (1965) *The Undecidable: Basic Papers On Undecidable Propositions, Unsolvable Problems And Computable Functions*. New York: Raven, pp. 417, 423.
84. See Copeland, B. J. (2000) 'The Turing test', *Minds and Machines*, 10, 519–39; Copeland, B. J. (1998) 'Turing's O-Machines, Penrose, Searle, and the brain', *Analysis*, 58, 128–38; and Piccinini, G. (2003) 'Alan Turing and the mathematical objection', *Minds and Machines*, 13, 23–48.
85. Letter from Darwin to Appleton, 23 July 1947 (PRO document reference DSIR 10/385; a digital facsimile is in The Turing Archive for the History of Computing <www.AlanTuring.net/darwin_appleton_23jul47>).
86. Probably at the end of September (see Chapter 3, note 67). Turing was on half-pay during his sabbatical (Minutes of the Executive Committee of the NPL for 28 September 1948, p. 4 (NPL library; a digital facsimile is in The Turing Archive for the History of Computing <www.AlanTuring.net/npl_minutes_sept1948>)).
87. Turing, 'Intelligent machinery' (see note 7).
88. Michie, unpublished note (in the Woodger Papers).
89. Letter from Darwin to Turing, 11 November 1947 (in the Modern Archive Centre, King's College, Cambridge (catalogue reference D 5); a digital facsimile is in The Turing Archive for the History of Computing <www.AlanTuring.net/darwin_turing_11nov47>).
90. Gandy in interview with Copeland (November 1995).
91. Minutes of the NPL Executive Committee for 28 September 1948, p. 4 (see note 86).
92. Turing, 'Intelligent machinery', p. 431.
93. Holland, J. H. (1992) *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press, p. x.
94. Turing, 'Intelligent machinery', p. 431.

95. See, for example, Newell, A. and Simon, H. A. (1976) 'Computer science as empirical inquiry: symbols and search', *Communications of the Association for Computing Machinery*, 19, 113–26.
96. Turing, A. M. (1950) 'Computing machinery and intelligence', *Mind*, 59, 433–60; reprinted in Copeland (ed.), *The Essential Turing* (see note 1).
97. Letter from Champernowne (January 1980) *Computer Chess*, 4, 80–1.
98. Michie, D. (1966) 'Game-playing and game-learning automata', in L. Fox (ed.), *Advances in Programming and Non-numerical Computation*. New York: Pergamon, p. 189.
99. Turing, A. M. (1953) 'Chess', part of ch. 25 in B. V. Bowden (ed.), *Faster Than Thought*, London: Sir Isaac Pitman & Sons; reprinted in Copeland (ed.), *The Essential Turing*.
100. Prinz, D. G. (1952) 'Robot chess', *Research*, 5, 261–6.
101. Bowden, *Faster than Thought*, p. 295.
102. Gradwell, C. 'Early Days', reminiscences in a Newsletter 'For those who worked on the Manchester Mk I computers', April 1994. (Copeland is grateful to Prinz's daughter, Daniela Derbyshire, for sending him a copy of Gradwell's article.)
103. Prinz, D. G. 'Introduction to Programming on the Manchester Electronic Digital Computer', no date, Ferranti Ltd. (a digital facsimile is in The Turing Archive for the History of Computing <www.AlanTuring.net/prinz>). Turing, A. M. 'Programmers' Handbook for Manchester Electronic Computer' (see note 64).
104. Mays, W. and Prinz. D. G. (1950) 'A relay machine for the demonstration of symbolic logic', *Nature*, 165, no. 4188, 197–8; Prinz D. G. and Smith, J. B. 'Machines for the Solution of Logical Problems', in Bowden (ed.), *Faster than Thought*.
105. Letter from Strachey to Woodger, 13 May 1951 (in the Woodger Papers).
106. Letters from Woodger to Copeland, 15 July 1999 and 15 September 1999.
107. Turing, A. M. 'Programmers' Handbook for Manchester Electronic Computer' (see note 64).
108. Campbell-Kelly, M. (1985) 'Christopher Strachey, 1916–1975: a biographical note', *Annals of the History of Computing*, 7, 19–42, p. 24.
109. Strachey, C. S. (1952) 'Logical or non-mathematical programmes', *Proceedings of the Association for Computing Machinery* (Toronto, September, 1952), 46–9, p. 47.
110. Samuel 'Some studies in machine learning using the game of checkers', in Feigenbaum and Feldman (eds), p. 104 (see note 72).
111. Letter from Oettinger to Copeland, 19 June 2000; Oettinger, A. (1952) 'Programming a digital computer to learn', *Philosophical Magazine*, 43, 1243–63.
112. Oettinger in interview with Copeland (January 2000).
113. 'Programming a digital computer to learn', p. 1247 (see note 111).
114. Ibid., p. 1250.

115. This section is an edited version of our paper 'On Alan Turing's anticipation of connectionism', *Synthese*, 108 (1996), 361–77, and appears here by kind permission of Springer Science and Business Media.
116. For additional discussion of these machines, see Copeland, B. J. and Proudfoot, D. (1999) 'Alan Turing's forgotten ideas in computer science', *Scientific American*, 280, 76–81.
117. Turing, 'Intelligent machinery', p. 425 (see note 7).
118. Turing, 'Computing machinery and intelligence', p. 457 (see note 96).
119. Turing, 'Intelligent machinery', pp. 417–18.
120. Ibid., p. 424.
121. Ibid., p. 424.
122. Ibid., p. 417.
123. Each unit of an A-type circuit introduces a delay of one moment into the circuit. Suppose, for example, that the job of some particular unit U in the circuit is to compute XNANDY for some pair of specific truth-functions X and Y. The subcircuits that compute X and Y may deliver their outputs at different moments yet obviously the values of X and Y must reach U at the same moment. Turing did not say how this is to be achieved. A nowadays familiar solution to this problem—which also arises in connection with CPU design—involves the concept of a 'cycle of operation' of *n* moments duration. Input to the machine is held constant, or 'clamped', throughout each cycle and output is not read until the end of a cycle. Provided *n* is large enough, by the end of a cycle the output signal will have the desired value.
124. Turing, 'Intelligent machinery', p. 424.
125. Ibid., p. 422.
126. Ibid., p. 418.
127. Ibid., p. 418.
128. Concerning specific interfering mechanisms for changing the state of *A*, Turing remarked '[i]t is … not difficult to think of appropriate methods by which this could be done' and he gave one simple example of such a mechanism (ibid., pp. 422–3).
129. Turing, 'Intelligent machinery', p. 418.
130. All B-types are A-types but not vice versa.
131. Turing, 'Intelligent machinery', p. 422.
132. Ibid., p. 422.
133. Ibid., p. 424.
134. Ibid., p. 422. Such proofs have been given for a number of modern connectionist architectures, for example, by Pollack, J. B. (1987) *On Connectionist Models of Natural Language Processing* (Ph.D. Dissertation, University of Illinois, Urbana) and by Siegelmann, H. T. and Sontag, E. D. (1992) 'On the computational power

of neural nets', *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 440–9. Siegelmann and Sontag establish the existence of a network capable of simulating a finite-tape universal Turing machine in linear time. They are able to give an upper bound on the size of the network: at most 1058 units are required.

135. That Turing anticipated connectionism was first suggested to us by Justin Leiber in correspondence. Leiber gives a brief discussion on pp. 117–18 and 158 of his *An Invitation to Cognitive Science*, Oxford: Basil Blackwell (1991), and on p. 59 of his 'On Turing's Turing test and why the matter matters', *Synthese*, 104 (1995), 59–69. (We cannot endorse Leiber's claim (*An Invitation to Cognitive Science*, p. 118) that Turing made use of weighted connections.)
136. Rosenblatt, F. (1958) 'The perceptron: a probabilistic model for information storage and organization in the brain', *Psychological Review*, 65, 386–408; the quotation is from p. 387.
137. The key to success in the search for training algorithms was the use of weighted connections or some equivalent device such as variable thresholds. During training the algorithm increments or decrements the values of the weights by some small fixed amount. The relatively small magnitude of the increment or decrement at each step makes possible a smooth convergence towards the desired configuration. In contrast there is nothing smooth about the atomic steps involved in training a B-type. Switching the determining condition of an introverted pair from 0 to 1 or vice versa is a savage all-or-nothing shift. Turing seems not to have considered employing weighted connections or variable thresholds.
138. Turing, 'Intelligent machinery', p. 428.
139. Hebb, D. O. (1949) *The Organization of Behavior: A Neuropsychological Theory*. New York: John Wiley.
140. Rosenblatt, F. (1957) 'The Perceptron, a Perceiving and Recognizing Automaton', Cornell Aeronautical Laboratory Report No. 85-460-1; (1958) 'The Perceptron: a Theory of Statistical Separability in Cognitive Systems', Cornell Aeronautical Laboratory Report No. VG-1196-G-1; (1958) 'The perceptron: a probabilistic model for information storage and organization in the brain' (see note 136); (1959) 'Two theorems of statistical separability in the perceptron', in (anon.) *Mechanisation of Thought Processes*, Vol 1, London: HMSO; (1962) *Principles of Neurodynamics*, Washington, DC: Spartan.
141. Ross Ashby, W. (1952) *Design for a Brain*. London: Chapman and Hall.
142. Beurle, R. L. (1957) 'Properties of a mass of cells capable of regenerating pulses', *Philosophical Transactions of the Royal Society of London*, Series B, 240, 55–94.
143. Taylor, W. K. (1956) 'Electrical simulation of some nervous system functional activities', in C. Cherry (ed.) *Information Theory*. London: Butterworths.

144. Uttley, A. M. 'Conditional probability machines and conditioned reflexes' and 'Temporal and spatial patterns in a conditional probability machine', both in C. E. Shannon and J. McCarthy (eds) (1956) *Automata Studies*, Princeton: Princeton University Press; and 'Conditional probability computing in a nervous system', in *Mechanisation of Thought Processes*, Vol 1 (see note 140).
145. Rosenblatt, *Principles of Neurodynamics*, especially pp. 5 and 12 ff.
146. Hebb, *The Organization of Behavior*.
147. Rumelhart, D. E., McClelland, J. L., and the PDP Research Group (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol 1: *Foundations*, Cambridge, MA: MIT Press; see, for example, pp. 41 ff., 152 ff., 424.
148. The pioneering work of Beurle, Taylor, and Uttley has been neglected almost to the same extent as Turing's. According to connectionist folklore the field of neuron-like computation originated with Rosenblatt, influenced by McCulloch, Pitts, and Hebb. However this is incorrect. Rosenblatt recorded that the 'groundwork of perceptron theory was laid in 1957' (p. 27 of his *Principles of Neurodynamics*). A series of memoranda by Uttley concerning his probabilistic approach to neuron-like computation survives from as early as 1954 (Uttley, A. M. (1954) 'Conditional Probability Machines and Conditioned Reflexes' (RRE Memorandum No. 1045); (1954) 'The Classification of Signals in the Nervous System' (RRE Memorandum No. 1047); (1954) 'The Probability of Neural Connections' (RRE Memorandum No. 1048); (1954) 'The Stability of a Uniform Population of Neurons' (RRE Memorandum No. 1049). Published accounts of the work of Beurle, Taylor, and Uttley appeared prior to 1957 (see the references given above). Rosenblatt's work was also prefigured in the United States by that of Clark and Farley (Farley, B. G. and Clark, W. A. (1954) 'Simulation of self-organizing systems by digital computer', *Institute of Radio Engineers Transactions on Information Theory*, 4, 76–84; Clark, W. A. and Farley, B. G. (1955) 'Generalization of pattern recognition in a self-organizing system', *Proceedings of the Western Joint Computer Conference*, 86–91). In 1954 Clark and Farley simulated a network of threshold units with variable connection weights. The training algorithm (or 'modifier') that they employed to adjust the weights during learning is similar to the algorithms subsequently investigated by Rosenblatt. Rosenblatt acknowledged that 'the mechanism for pattern generalization proposed by Clark and Farley is essentially identical to that found in simple perceptrons' (*Principles of Neurodynamics*, p. 24).
149. Farley and Clark, 'Simulation of self-organizing systems by digital computer'.
150. McCulloch and Pitts, 'A logical calculus of the ideas immanent in nervous activity', p. 129 (see note 29).
151. Taub (ed.), *Collected Works of John von Neumann*, Vol 5, p. 319.

152. McCulloch and Pitts, 'A logical calculus of the ideas immanent in nervous activity', pp. 119, 123–4.
153. Wiener, N. (1948) *Cybernetics*. New York: John Wiley, p. 32.
154. Gandy in interview with Copeland (November 1995).
155. *Manchester Guardian*, 11 June 1954.
156. McCulloch and Pitts, 'A logical calculus of the ideas immanent in nervous activity', pp. 117, 124.
157. Turing considered other modifications, in particular sensory input lines and internal memory units.
158. Turing, 'Intelligent machinery', p. 425.
159. Ibid., pp. 426–7.
160. Ibid., p. 425.
161. Ibid., p. 427.
162. Ibid., pp. 427–8.
163. Ibid., p. 428.
164. Langton, C. G. (1989) 'Artificial life', in C. G. Langton (ed.), *Artificial Life: The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*. Redwood City, CA: Addison-Wesley, p. 32.
165. Turing, A. M. (1952) 'The chemical basis of morphogenesis', *Philosophical Transactions of the Royal Society of London*, Series B, 237, 37–72; reprinted in Copeland (ed.), *The Essential Turing*. Turing employed nonlinear differential equations to describe the chemical interactions hypothesized by his theory and used the Manchester computer to explore instances of such equations. He was probably the first researcher to engage in the computer-assisted exploration of nonlinear systems. It was not until Benoit Mandelbrot's discovery of the 'Mandelbrot set' in 1979 that the computer-assisted investigation of nonlinear systems gained widespread attention.
166. Letter from Turing to Woodger, undated, marked as received on 12 February 1951 (in the Woodger Papers; a digital facsimile is in The Turing Archive for the History of Computing <www.AlanTuring.net/turing_woodger_feb51>).
167. 8 February 1951. A copy of Turing's letter (typed by his mother, Sara Turing) is in the Modern Archive Centre, King's College, Cambridge (catalogue reference K1.78).
168. These are in the Modern Archive Centre, King's College, Cambridge (catalogue reference C 24–C 27).