34

CYBERNETICS AND INFORMATION TECHNOLOGY

MICHAEL S. MAHONEY

Since the Second World War, 'information' has emerged as a fundamental scientific and technological concept applied to phenomena ranging from black holes to DNA, from the organisation of cells to the processes of human thought and from the management of corporations to the allocation of global resources. In addition to reshaping established disciplines, it has stimulated the formation of a panoply of new subjects and areas of inquiry concerned with its structure and its role in nature and society. Embodied in the computer, theories based on the concept of 'information' have so permeated modern culture that it is now widely taken to characterise our times. We live in an 'information society', an 'age of information'.

Current awareness of the fundamental nature of information and of its determinative role in modern life makes it difficult to unravel the threads of its history, especially when following them back to a time when 'information' had only common meaning. One may speak today of the printing press, typewriter, telegraph or telephone as 'information machines', but that is not how they were originally conceived, and including them in the history of information risks losing sight of the time and place at which the concept itself emerged and of the process by which it came to shape modern thought and even self-consciousness.

Information in the modern technical sense emerged from a complex of activities which, underway just before the Second World War, gained impetus from wartime research and culminated in 1948 with the appearance of Norbert Wiener's *Cybernetics: or Control and Communication in the Animal and the Machine* (1948) and Claude E. Shannon's *The Mathematical Theory of Communication* (1948). Although neither work encompassed the full variety of current research

related to it, both provided initial points of reference for organising and directing subsequent investigation. In addition, Wiener's cybernetics and Shannon's 'information theory' (as it was commonly called, despite his disapproval) attracted the attention of scholars in other fields, who sought to apply or adapt these new concepts and methods to their own concerns.

1. CYBERNETICS AND INFORMATION THEORY

Wiener used the term 'cybernetics' to characterise the common elements of his work with Vannevar Bush on computing machines, with Yuk Wing Lee on electrical networks, with Julian Bigelow on the prediction of flight paths and with Arturo M. Rosenblueth and Walter Pitts on neuromuscular behaviour and neuro-physiology.[1] Fundamentally, Wiener concluded that control in both mechanical and biological systems depends on feedback, which in turn requires communication of information within the system. Secondly, 'the ultra-rapid computing machine, depending as it does on consecutive switching devices, must represent an ideal model of the problems arising in the nervous system' and conversely 'was in principle an ideal central nervous system to an apparatus for automatic control'.[2]

The intimate link between control and communications shifted attention from the specifics of electrical engineering to the more general notion of the message, however transmitted. Viewed as time series, messages become predictable through statistical analysis and prediction can be optimised by means of the calculus of variations. Messages transmitted over physical channels are subject to distortion by background noise and hence raise the problem of their accurate reconstruction. Consideration of that problem led Wiener to the question of the measure of information and through it to the relation of information and entropy, by which he arrived again at the living organism.

The nature of information linked Wiener's work to Shannon's, whose paper had the more narrowly defined goal of defining the capacity of a communications channel, of determining 'the effect of statistical knowledge about the source in reducing the required capacity of the channel, by the use of proper encoding of the information' and of setting the limits of possibility of correctly construing a message transmitted in the presence of noise.[3] Starting from the principle that information resolves uncertainty, Shannon measured information with reference to the number of possible messages that could be sent in a given time using a given set of symbols. The precise form of measure being arbitrary, Shannon chose as the unit of information for discrete channels the single binary digit or 'bit' (coined by John Tukey of Bell Labs.) A set of n symbols s_i, each with a corresponding probability p_i, has the informational content of $H=-\Sigma p_i \log_2(p_i)$ bits; if the symbols are equiprobable, the measure reduces to $H=\log_2 n$, which is the maximum. More usual are messages in which the

different p_i depend on previous sequences (Markov processes) and in which the distributions in randomly chosen messages of sufficient length are representative of distributions over all messages (ergodic processes).

Viewed as a measure of uncertainty or surprise, information can be construed as a form of entropy and Shannon chose his function with Boltzmann's H theorem in mind. By increasing the uncertainty about the source of signals and hence about the message being transmitted, noise adds to the entropy of a channel, detracting from its useful capacity. However, as Shannon showed, given sufficient capacity, the entropy of noise can be overcome to an arbitrary degree by a suitable encoding of the message, which in essence lowers the entropy of the source through redundancy. That one pays for certainty at the cost of information places a premium on efficiency of coding, a subject which, already under investigation at the time, gained new impetus from Shannon's results.[4]

Tied to systems of communications and thus to the problem of selecting the correct message from a range of possible ones, Shannon's measure of information did not meet the needs of others seeking to quantify 'constructive' information, for example Donald McKay and Dennis Gabor.[5] Nonetheless, it soon became a touchstone for work in a range of fields. Léon Brillouin's analysis in depth of the relation of entropy and information led in the mid-1950s to the formulation of statistical thermodynamics as a branch of information theory. Joining Shannon's theory with John von Neumann's research on automata and the construction of reliable organisms from unreliable components, Samuel Winograd and Jack D. Cowan established the principles of reliable computation in the presence of noise (1963). Other applications were less successful. Enthusiasm among biologists in the 1950s for applying Shannon's results to genetic information waned in the 1960s, largely owing to equivocal use of the term 'information' itself. Similar overextensions of the theory led to a certain disenchantment with it by the late 1960s. Nonetheless, Shannon's work remained fundamental to the theory of coding and hence made its continuing presence felt not only in communications research but also in computer science.

Cybernetics followed a similar pattern of enthusiastic reception followed by overextension and disenchantment. By the late 1960s the term itself gradually disappeared from use in the United States and Western Europe, although it is still common in the Eastern bloc. Yet, the themes and approaches it encompassed have continued to develop since then, pursued separately in the disciplines Wiener sought to bring together and in new disciplines spawned from them.[6] Both the fate of cybernetics and its effect on other subjects are perhaps best understood by examining more closely the technological context in which it arose.

As Wiener himself emphasised, in both conception and implementation, cybernetics was intimately tied to the new forms of automatic computation

developing rapidly during the decade from the late 1930s to the late 1940s. In addition to providing a tool for solving the problems in applied mathematics that Wiener's methods entailed, it served as his model of the nervous system. The design of computers in turn posed problems of a cybernetic nature and hence constituted a field of application for his methods, as it did for Shannon's. Yet, the computer had its own intellectual and technical roots, which determined its development independently of its role in cybernetics and which thereby conditioned that role. Especially for those who viewed cybernetics in the wider sense of systems theory, the computer ultimately offered resources that far exceeded Wiener's vision or even contradicted it.

In particular, the computer both accommodated and encouraged a broader view of 'information', and of how it can be transformed and communicated over time and space, than that which underlay the new theory. It thereby transformed traditional methods of accounting and record-keeping into a new industry of data processing, posing both unprecedented possibilities and unanticipated dangers. Since the 1950s the computer, both as processor of information and as vehicle of communication over both space and time, has come to form the core of modern information technology. What the English-speaking world refers to as computer science is known to the rest of Western Europe as *informatique* (or *Informatik* or *informática*). Much of the concern over information as a commodity and as a natural resource derives from the computer and from computer-based communications technology.[7] Hence, the history of the computer and of computing is central to that of information science and technology, providing a thread by which to maintain bearing while exploring the ever-growing maze of disciplines and subdisciplines that claim information as their subject.

2. COMPUTERS AND COMPUTING

The computer itself is the product of two lines of historical development corresponding roughly to the distinction between the physical machine, or *hardware*, and the programs or *software*, that guide its operation. As a physical device, the first all-electronic calculator, ENIAC, crowned longstanding efforts to automate large-scale computation and tabulation. As logic machines, the first stored-program computers, EDSAC and EDVAC, emerged as by-products of theoretical inquiry into the nature and limits of logical thought, in particular as a foundation for mathematics.

2.1. The electronic calculator

The mechanisation of computation began in the late Middle Ages with the invention of such analog devices as mechanical clocks, planetaria and related

automata. The first digital mechanisms appeared in the seventeenth century. The machines of Wilhelm Schickard (1623) and Blaise Pascal (the 'Pascaline', 1654) used geared wheels to perform addition and substraction automatically upon entering the terms. While Schickard retained logarithmically scaled rods ('Napier's Bones') for multiplication and division, Gottfried Wilhelm Leibniz devised a shifting mechanism for translating those operations into repeated addition and subtraction (1672). Legend would have it that these machines responded to computational needs; one hears, for example, of Pascal seeking to ease his father's job as tax collector. Yet, in practice, none operated at the speed of a skilled human reckoner. Rather, they were viewed as objects of wonder in their capacities as automata to emulate not physical actions, but the highest form of rational thought.

Only in the early nineteenth century did the computational needs of science, especially the new fields of thermodynamics and electricity and magnetism, together with their applications to industry, begin to stimulate a steady development of practical devices of increasing power. The hand-driven mechanical calculator reached its final configuration, while new designs appeared in the form of the 'difference engine', the planimeter and the harmonic analyser. The first of these, invented in 1823 by Charles Babbage to automate the calculation of mathematical and astronomical tables and subsequently improved by Georg Scheutz and others, reflected the recent development of the calculus of finite differences and approached the limits of digital computation by mechanical linkage, thus encouraging a turn to analog devices. Various planimeters (J. H. Hermann (1814), Jacob Amsler (1854), Clerk Maxwell (1855), J. Thompson (1876)) translated the integration of curves into continuous compound mechanical action, while the harmonic analyser (Kelvin (1873), Michelson and Stratton (1898)) linked the integrators to model the solution of differential equations via Fourier series.

By the middle of the nineteenth century, new commercial and industrial organisations, especially those that conveyed the increasing output of new forms of production to an ever-widening consumerate, enhanced the demand for improved accounting machines while adding to it the need for new means of mechanical record-keeping and tabulation of data. Governments, too, added to the demand as industrialisation placed new responsibilities on them in the realms of finance, regulation, public health and social services and thus multiplied both the volume and the importance of social statistics. In the United States, where apportionment of congressional representation rests on a decennial census, it was clear by the late 1880s that without automatic tabulation the count for 1890 could not be completed before the year 1900. Herman Hollerith's electrically activated punched-card tabulating machine (1894) responded directly to that need and became the basis for a new industry.

While responding to immediate needs, some inventors looked beyond them to longer-range possibilities, thus focusing attention on the nature of

computation itself. Foremost among them was Charles Babbage. Before completing the construction of his Difference Engine, he became absorbed in overcoming its limitation to executing only a single compound calculation at a time, the form of which is built into the machine, and in 1834 conceived an 'Analytical Engine' which would automatically carry out a variety of operations in response to a sequence of commands. His design, modelled in part after Jacquard's loom and in part after the maze of gears and shafts that was the early English factory, was a microcosm of the Industrial Revolution: values placed in 'storage' moved in and out of an arithmetical 'mill' performing operations recorded on punched cards. Since its mechanical form dictated that power varied with size, Babbage looked beyond hand-driven models to the driving force of steam. Except for a few sub-units, the engine never went beyond the stage of intricate drawings and was soon all but forgotten. Its structure probably exceeded the capabilities of machine technology and would in any case have been too slow to be useful. It did, however, provoke the first effort to specify the nature of the operations involved in instructing a machine and thereby earned Ada, Countess Lovelace the reputation of the world's first programmer.

The distribution of electrical power and telephone service in the early twentieth century posed new computational problems, while providing new models for their solution. The design of electric power grids entailed the solution of large systems of simultaneous equations, both finite and differential. Here analog devices taking increased advantage of electrical technology continued to lead the field. A 'product integraph' developed at MIT under the guidance of Vannevar Bush (1927) integrated expressions of the form $f_1(x)f_2(x)dx$ by mechanical analog multiplication followed by integration via electric meter and mechanical rotation. Bush's differential analyser (1931) reversed that action, adding the output of electronic torque amplifiers to multiply via integration, i.e. $uv = \int udv + \int vdu$.

Telephone switching systems involved increasingly complex configurations of electrical relays opening and closing circuits in response to input from users, a process which was soon recognised as itself a form of computation. The logical analysis of such systems (in particular by Shannon at MIT in the late 1930s) suggested the linking of relays to model binary arithmetic. A simple adder circuit designed by George Stibitz at Bell Telephone Laboratories in 1937 soon led to a relay device capable of handling complex numbers (Model I, 1939); reached in New York via telegraph lines from Hanover, New Hampshire, it also provided the first demonstration of remote computation in 1940. During the same period, Howard Aiken at Harvard, with the support of IBM, began the construction of a general-purpose electromechanical computer, the Mark I (1943).

While electrical relays made binary computation practical, the remaining mechanical components placed a physical limit on the speed of computation.

Even as the relay machines were being designed, they stimulated experiments with circuits using electronic tubes instead of relays. J. V. Atanasoff constructed a small device at the University of Iowa (1939–40). Under circumstances still not entirely clarified, Atanasoff's work came to the attention of John Mauchly soon before he moved from Ursinus College to the Moore School of Engineering at the University of Pennsylvania to take charge of a Bush differential analyser then being pressed into national service for the computation of firing tables for the United States Army's Ballistic Research Laboratory. With the support of Herman Goldstine, the Army's technical liaison to the project, Mauchly and his colleague J. Presper Eckert gained approval in 1943 for the development of the first all-electronic computer, the Electronic Numerical Integrator and Automatic Calculator (ENIAC), which came on line in 1946.[8]

Using some 18,000 vacuum tubes in a circuit design that seemed to defy the inherent probability of failure, ENIAC retained in its design some of its origins in mechanical calculation. It did arithmetic decimally, using ring counters and storing intermediate results in accumulators consisting of banks of such counters. Fixed data (constants) were held in function tables set manually by switches and, in a manner similar to computation using electrical accounting machinery, the computer followed a fixed program set up manually by setting switches and connecting wires. Nonetheless, operating at a speed of some 5000 ten-digit additions a second, it marked the advent of modern high-speed computing and served as prototype for all hardware to follow.

2.2. The stored-program computer

As progress on ENIAC demonstrated the feasibility of large-scale electronic calculation, thought turned to internalising its programs by enabling it to modify its instructions in response to the results it was generating. Again, the precise details have not been resolved, but it seems clear that John von Neumann played a major role in laying out the logical structure of the stored-program computer. In addition to his own ideas on automata, he brought to the task an understanding of the work of Alan Turing, whose article, 'On Computable Numbers, with an Application to the *Entscheidungsproblem*', defined the abstract structure of machines capable of logical calculation and who was himself at work designing one of England's earliest computers, the Pilot ACE.[9]

Turing's theoretical work belonged to a quite different line of scientific development from that of mechanical calculators, yet one that also stemmed from the early nineteenth century, with roots reaching back initially to the seventeenth. In addition to designing the earliest four-operation calculator, Leibniz also explored binary arithmetic and discussed the idea of a symbolic logical calculus.[10] But these ideas remained separate in his mind and undeveloped in his works. Not until the nineteenth century did they again emerge as

sustained themes of mathematical inquiry, first in the efforts of George Boole to develop an algebra of logic that might serve as a vehicle for the 'laws of thought'. His study of the theory of operators led him to the notion of classes as operators on sets of objects and then to an algebra of those classes. For example, if C is a deck of cards, one may take x as an operator selecting the hearts from the deck, y as one selecting the diamonds and z as one selecting face cards. Then $xC + yC = (x + y)C$ selects all red cards and $z(x + y)C$, all red face cards. Clearly, $x^2(C) = x(x(C)) = x(C)$. Moreover, if $C - xC = (1 - x)C$ designates all non-hearts, then $x + (1 - x) = 1$, and $x(1 - x) = x - x^2 = x - x = 0$ and so on. Since the criterion of selection in each case is binary, Boole's classes correspond to logical functions and their relations to an algebra of logic.

Given the cultural presence of machinery in the nineteenth century, it is not surprising that efforts to mathematise or quantify the rules of logic should issue in the design of logic machines such as those of James Jevons in England and Allan Marquand in the United States. Marquand's friend Charles S. Peirce even suggested replacing the rods and strings of the machine with electrical circuits. But the devices served no practical purposes and remained at best aids to understanding the structure of logical reasoning.

In the late nineteenth century attention shifted from the mathematics of logic to the logical foundations of mathematics, as new fields like non-Euclidean geometry, vector algebra and abstract finite algebras undermined reliance on intuition and spurred a drive toward rigorous formalisation. In that context, Gottlob Frege attempted a reduction of arithmetic to logic (*Grundlagen der Arithmetik*, 1884) and produced instead the first of the antinomies (Russell's paradox) that soon cast the consistency of logic itself in doubt; an equally troubling paradox issued from the work of Georg Cantor and Cesare Burali-Forti on transfinite numbers (1895–7). David Hilbert's famous Program of 1900 included in its agenda for mathematics a formalisation of the subject capable of resolving what he later (*Grundzüge der theoretischen Logik*, 1928, with W. Ackerman) termed the *Entscheidungsproblem*: is there a general procedure for deciding whether a statement of a given axiom system has a proof in that system? Bertrand Russell's and Alfred North Whitehead's *Principia mathematica* (1910) unintentionally emphasised the centrality of the problem when Kurt Gödel ('On formally undecidable propositions in *Principia mathematica*', 1931) showed that any logic powerful enough to generate arithmetic had to be either incomplete – that is, include formally undecidable propositions – or inconsistent.

Gödel's proof employed the technique of assigning numbers to the statements of a system and then couching the proof of a proposition from the axioms in terms of an effective procedure for computing the proposition's number from those of the axioms. Decidability came down to computability. Turing's paper of 1936 gave a precise definition to the notion of an effective computational

procedure by expressing it as an abstract finite-state machine consisting of a potentially infinite tape divided into cells and passing under a device capable of reading from a cell, or writing to it, one of a finite set of symbols and of moving right or left to the neighbouring cell. The machine carried out an effective procedure by passing through a finite set of states in a finite number of steps from an initial configuration of symbols on the tape to a desired final configuration. Each state could be characterised as a vector containing the symbol in the current cell, the action to be taken and the state to follow; a vector of the state vectors then characterised the procedure and thus the machine itself.

On the basis of this definition, Turing showed that there exist numbers which are definable but not computable. Moreover, since by a suitable convention the last vector could be translated into a configuration of symbols on the tape, Turing machines themselves were subject to computation by Turing machines. On that basis, Turing proved that 'there can be no general process for determining whether a given formula [A] of the [restricted Hilbert] functional calculus K is provable, *i.e.* that there can be no machine which, supplied with any one [A] of these formulae, will eventually say whether [A] is provable'.[11]

Turing set out as the most general form of his machine a universal procedure for reading the description of any Turing machine at the beginning of a tape and then instantiating it for the configuration on the remainder of the tape. That universal machine contained the germ of the stored-program computer, for it meant that the same tape could contain both instructions and data, interpreted as one or the other in accordance with the state of the scanning head. In essence, the computer replaced the tape by an addressable memory holding sequences of binary digits and the read-write head by separate arithmetical and control units, the former of which operates on those sequences and the latter of which interprets them as instructions. Treated as data, instructions can be modified in the course of computation; more importantly a set of instructions can be self-modifying in response to the results it is generating.

The original ENIAC group had taken some steps toward the internal storage of its instructions when von Neumann joined their efforts, but the fundamental logical structure of a stored-program device emphasised in his 'First draft of a report on the EDVAC' (1945) appears to have stemmed from him. With that report and the 'Preliminary Discussion of the Logical Design of an Electronic Computing Instrument' composed together with Arthur Burks and Herman Goldstine in 1946, the electronic digital computer assumed its basic form. Cambridge University's EDSAC, under the direction of Maurice Wilkes, first realised that form in a working machine in 1949. Eckert and Mauchly's BINAC soon followed in the United States.

3. THE COMPUTER INDUSTRY

Viewed before the war as an esoteric device of unproved value, the electronic computer was born of extensive government funding in response to specific military needs. In the immediate post-war years also, even after ENIAC had demonstrated the feasibility of the computer and its importance to scientific research, further development still depended on government funding, either by direct subvention or through contracts for the development of specific machines. Although Ferranti Ltd joined with the University of Manchester in 1949 to develop a commercial version of its prototype Mark I, it was not until 1950, as the first stored-program computers went into operation, that major corporations in the United States began to take an interest in them as potential products. Remington Rand acquired Engineering Research Associates and the Eckert and Mauchly Company, then building the famous Univac, and was in turn acquired by Sperry. IBM, which in 1948 had built the (partially electromechanical) Selective Sequence Electronic Computer as a single showpiece for its world headquarters, entered the market only in response to Univac's threat to its line of electrical accounting machinery. Its first commercial machines, the 701 and 704 appeared in 1953, followed a year later by the popular 650 with its magnetic drum storage.

The first trickle of commercial machines soon became a flood. By 1965, customers in the United States could choose among more than 100 models offered by over twenty manufacturers; an additional 100 models were available from over 25 companies world-wide. The advent of the minicomputer around 1970, and of the microcomputer a decade later, brought similar spurts of growth to the industry. Side by side with the manufacture of the devices themselves emerged a new data-processing industry, which by 1970 accounted for some 2 per cent of the gross national product of the United States.

The commercialisation of the computer shaped its development both as a scientific and technical device and as the focus of an emerging professional discipline with its attendant institutions. Conceived for scientific purposes and born of military needs, the device initially responded to no immediate demand from the world outside science and engineering. Few, if any, of IBM's customers clamoured for electronic versions of their electrical and electromechanical accounting equipment. Rather, that demand had to be created by devising applications for the computer in the realms of finance, management and communications. Those applications and the machines needed to implement them meant keeping pace with scientific and technological development, which in turn meant closer and more open ties between industrial research institutions and the rapidly expanding scientific community in the universities and at now-permanent government installations. Dependent in turn on the computer industry for funding and for technical support, academic computer science took

shape partly in response to corporately defined research needs. From the outset, the careers of computer people show a characteristic pattern of regular and easy movement between campus, industry and government facilities.

3.1. Computer science

From about 1950 onwards, computing gradually assumed a shape and place of its own among the disciplines of science and engineering. In 1954, the seven-year-old Association for Computing Machinery (ACM) announced in the first number of its *Journal* that it would henceforth leave questions of hardware to the American Institute of Electrical Engineering and the Institute of Radio Engineers and direct its efforts instead to 'the other phases of computing systems, such as numerical analysis, logical design, application and use and, last but not least, to programming'. Far from last or least, programming soon emerged as a body of concepts and techniques distinct from the architectures of particular machines, as its practitioners identified the algorithms, data structures, search and sort routines on which programs rest. *Preparation of Programs for an Electronic Digital Computer* (1951) by Maurice Wilkes, D. J. Wheeler and Stanley Gill marks perhaps the beginning of that line of development, while Donald E. Knuth's *The Art of Computer Programming* (1969) represents a major stage of consolidation.

In the early 1960s the ACM followed the National Academy of Sciences and the Mathematical Association of America in recognising 'computer science' as a distinct field of study 'embracing such topics as numerical analysis, theory of programming, theory of automata, switching theory, etc.' and undertook, with considerable discussion, to define a suitable curriculum at graduate and undergraduate levels. Soon after, in 1967, recurrent difficulties encountered by large-scale programming projects prompted the NATO Science Committee to set up a study group on computer science, which in turn urged the establishment of a discipline of 'software engineering', which would base the manufacture of software 'on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering'. With professionalisation and ramification into sub-disciplines came a panoply of new organisations and publications. By 1970 some 400 journals strove to meet the informational needs of several hundred thousand computer scientists and data-processing professionals.

The intellectual origins of the computer combined neatly with the exigencies of its commercialisation in directing much of the research agenda during the next decades. Making the computer accessible to users from a wide range of backgrounds was essential both to exploring its theoretical potential and to marketing it. Computer scientists and business users had a common stake in the

development of programming languages, of operating systems and of applications to non-numerical data.

3.1.1. Programming languages

Essential both to Turing's theory of computing machines and to von Neumann's theory of self-reproducing automata was the notion of the self-programming computer. The stored program realised that notion to the extent of enabling the computer to modify its instructions and thus control the flow of computation in response to the results being generated. But those instructions initially had to be specified in excruciating detail using the machine's own language of sequences of binary digits, or 'bits'. 'Automatic programming' aimed at bringing the language of specification closer to human usage and at automating such standard tasks as allocation of memory, calling sequences for subroutines, assembly of subroutines from libraries, input/output protocols and loading formats.

The first 'assemblers' in the early 1950s gave rise to a 'Tower of Babel' of programming languages by the late 1960s and then to a measure of standardisation in the 1970s. The major achievements include FORTRAN (Backus, 1956), LISP (McCarthy, 1958), COBOL (Hopper, 1959), APL (Iverson, 1962), BASIC (Kemeny and Kurtz, 1965) and the succession of ALGOLs (international committees, 1958, 1960, 1968) capped by PASCAL (Wirth, 1971). The first report on ALGOL in 1959 set the standard for specifying the syntax of languages with John Backus's Normal Form (BNF), later modified by Peter Naur (whence 'Backus Naur Form'). That technique and others drawn from ongoing research in automata theory and mathematical linguistics, especially the work of Noam Chomsky, placed the design of languages and, to some extent, their compilers on theoretical foundations independent of any particular machine and hence made the languages themselves portable between different machines.

At the same time most of these languages remained tied through their primitive terms (integers, real numbers, characters and arrays) to the basic architecture of the computer. During the 1970s emphasis began to shift to languages which included a wider and more flexible range of data structures and objects – including programs – and which focused on the objects of computation and relations between them rather than on procedures. That development, in particular FORTH (Moore, 1969), C (Ritchie, 1972), Small Talk (Kay, 1972–80), ADA (Ichbiah, 1978) and MIT's LISP machine, aimed at embedding languages in a programming environment and reflected both the concomitant development of operating systems and the recent advent of powerful small computers.

3.1.2. Operating systems

Suitably programmed, the computer also had the capacity to oversee its own efficient operation. The high cost of computers spurred efforts to minimise the time during which the processor stood idle, either between jobs or during transfers of data (I/O for 'input/output') to and from much slower peripheral devices. Operating systems began in the early 1950s with simple monitor programs to schedule and set up jobs handled sequentially. Between 1955 and 1964, more elaborate multiprogramming systems supervised several programs at once, switching them in and out of the main processor as they waited for I/O; new multiprocessing systems allocated the work of programs over several processors. Beginning in 1962 'timesharing' systems adapted the notion of multiplexing to enable several users to share the resources of a single computer and to use the device interactively. In the United States these systems derived in part from the earlier development of the SAGE air defence system (1951–8) and SABRE airline reservation system (1963), both of which monitored transactions from many users working simultaneously, and served as prototypes for later real-time process control systems.[12]

Operating systems entered a new phase of development in 1964 with IBM's OS/360, designed to provide users with a common interface over a range of different machines and hence with generically defined systems services. Subsequent versions embodied the notion of the 'virtual machine', which presented each of many users with the image of an independent machine with its own operating system and permitted different users to use different systems. UNIX, introduced by Bell Labs in 1976, offered a highly interactive, multi-user environment linked by single file system. Operating systems perhaps reached the peak of their size and complexity in the late 1970s, as subsequent development, sparked by the advent of minicomputers and microcomputers, focused attention on the sharing of information and services between computers distributed over a network.

3.1.3. Software and its problems

Operating systems constituted only one sort of sophisticated, large-scale programming project undertaken in the 1960s. Others included, in addition to SAGE and SABRE, air-traffic control systems, large databases for government and industry, electronic switching systems for communications networks and control and communications systems for the United States's space programme. While many remarkable achievements issued from such projects, they also encountered difficulties that caused growing concern among practitioners

about the industry's capacity to produce reliable software on time and at reasonable cost. By 1969, leaders in the field were speaking of a 'software crisis' and urging that software be placed on a more scientific footing. One response was the concept of 'structured programming'. Introduced by Edsger Dijkstra in 1969 and developed further by C.A.R. Hoare in England and Niklaus Wirth in Switzerland and the United States, it aimed at a discipline of programming supported by appropriate languages and under-pinned by methods of theoretical verification (as opposed to empirical debugging). However, the movement encountered resistance among data processing workers. Despite the growing effort of software engineering during the 1970s software continued to resist efforts to automate its production and to establish standards of productivity and reliability.

3.2. Miniaturisation

The problems of software loomed even larger by contrast to the rapid development of hardware over the same period. During the 1950s the limitations of hardware both shaped and hindered the design of software. Assemblers and compilers require large memories to hold the intermediate tables and files they create in translating source code. So too do operating systems, which must remain resident (at least in part) while monitoring programs and responding to requests for system support. The more sophisticated the compiler or operating system, the greater the demand for memory and processor speed. Hence many of the developments in software depended on a series of revolutionary developments in electronics that produced order-of-magnitude increases in the speed and internal capacity of computers while reducing their cost and external dimensions to the same degree. As computer designers in the late 1940s looked to mercury delay lines, cathode ray tubes and diodes to reduce the number of vacuum tubes, researchers at Bell Laboratories assembled the first transistor (Brattain, Bardeen and Schockley, 1947–9). By the late 1950s, as transistor based computers were coming on the market, offering more computing power in less space for less money, experiments got underway with the first integrated circuits (Kilby, 1958; Noyce, 1959) which in turn made their commercial appearance in IBM's System 360 in 1964. Thereafter, the combination of miniaturisation and large-scale integration, spurred and supported by military research and the space program, produced a succession of ever more powerful chips, which by the late 1970s were capable of holding an entire central processing unit, thereby giving rise not only to supercomputers capable of carrying out 80–250 million operations per second but to the microcomputer and to the creation in the 1980s of a consumer market for computers and a return to the single-user system.

4. FROM CYBERNETICS TO COMPUTATION: ARTIFICIAL INTELLIGENCE

The increasing power and accessibility of computers ironically reinforced the autonomous tendencies of fields Wiener had envisioned united by cybernetics. By making feasible techniques like linear programming using George Dantzig's simplex method (1947), computers directed such fields as operations research and management science away from traditional applied mathematics and toward new methods of modelling and simulation using discrete mathematics, thus triggering new developments in that area while appropriating a major segment of Wiener's agenda. Interactively adapting to the computer, each discipline designed applications that best suited its own needs and in turn shaped its research to take advantage of what the device could do.

That turned out to be particulary true for the agenda perhaps closest to Wiener's heart: the computer as a model of the human nervous system and hence of human thought. Here, cybernetics' initially suggestive concepts of negative feedback, pattern recognition and stochastic learning proved unfruitful, as did efforts to emulate human perception by modelling neural structures.[13] Yet, by suggesting and supporting a representational approach to the meaning of 'control and communication' in the realm of human thought, that is as a tool for modelling rather than as a model itself, the computer became the foundation of the new field of artificial intelligence. (See art. 11.)

From the beginning, Turing viewed the computer as a manipulator of symbols and hence as capable of emulating any behaviour representable symbolically, in particular, game playing and similar manifestations of human intelligence. Turing machines, no less than Boole's algebra of logic, were meant to embody the laws of thought and hence to express human thinking. His design of the ACE reflected that intent, as did his early efforts to write a chess program and his espousal of machine intelligence. By contrast, von Neumann originally thought of the computer as a tool for the large-scale, high-speed calculations necessary for solving non-linear systems of equations and the basic architecture that bears his name likewise reflected this view, as did most programming languages with their focus on functions and procedures, even in the realm of data processing.

While scientific calculation and business data processing dominated the field of computing during its first decades, Turing's vision had its American adherents. In the early 1950s Shannon, impressed by chess-playing programs, took up the question of logical automata capable of self-directed behaviour. His colleagues at MIT included Marvin Minsky, then engaged in modelling neural networks on the computer in an effort to emulate perception and John McCarthy, working on a formal logic of human thought. Elsewhere, attracted to the computer as a tool of rational decision-analysis in management, Herbert

Simon teamed in the early 1950s with Allen Newell and J.C. Shaw to explore the heuristic capabilities of information processing. A 'Logical Theorist' program capable of finding and proving mathematical theorems led to the design of a 'General Problem Solver'. These first efforts came together in 1956 at a conference at Dartmouth College on 'artificial intelligence' (AI).

Despite the common designation, artificial intelligence encompassed from the outset a range of methods and agendas of research, diverse views concerning its short-range and long-term goals and changing relations with other disciplines such as psychology, neurophysiology and linguistics. Its (yet unwritten) history is correspondingly tangled and unclear, but one abiding feature has been the common emphasis on working programs as the embodiment of theory.[14] Main lines of investigation in the United States, Europe and Japan have included goal-directed planning, learning, understanding of natural language both syntactically and semantically, analysis and understanding of visual images, the organisation of knowledge about the world, the production of knowledge, and the emulation of expert problem-solving based on detailed knowledge in specific domains.[15] This last area, which emerged in the mid-1970s with programs capable of analysing molecular structure, diagnosing diseases and designing computer circuits, represented a new orientation of the field towards empirically-based systems aimed at assisting humans in solving problems of immediate practical significance. With expert systems, artifical intelligence moved into the market-place.

Although largely independent of mainstream computer science until the 1980s, AI spun off several major contributions to computing in general, for example time-sharing, techniques of graphical simulation, interactive debugging and computer design of VSLI circuits. In turn, its development by the end of the 1970s made it clear to many that application of such data-intensive strategies to more general realms of human thought lay beyond the limits of von Neumann architecture, as did significant progress on the earlier goals reached. Computers that can understand spoken and written speech, that can translate common language into programs and that can learn from their own experience, it was argued, require a 'Fifth Generation' of computer hardware and software. In the early 1980s Japan made the development of such computers a national goal, while several private ventures got underway in the United States. The outcome remains at present undecided.

NOTES

1. From the Greek *kybernetes* for 'steersman' (= Latin *gubernator*); often credited to Wiener, the neologism stemmed from Ampère's *Philosophie des sciences* (Paris, 1838), where it denoted a subdiscipline of the study of government.
2. Wiener, *Cybernetics* (2nd ed., Cambridge, Mass., 1961), p. 14, p. 26.
3. Shannon, in Claude E. Shannon and Warren Weaver, *The mathematical theory of communication*,

(Urbana, IL., 1949), p. 39; Shannon's paper originally appeared in the *Bell system technical journal*, 27 (1948), 379–423; 623–56.

4. Cf. Richard W. Hamming, 'Error detecting and error correcting codes', *Bell system technical journal*, 29 (1950), 147–60.
5. Cf. Donald M. MacKay, 'The wider scope of information theory', in Machlup and Mansfeld, *The study of information* (New York, 1984) pp. 485–92; esp. pp. 486–7.
6. Cf. Peter Elias, 'Cybernetics: past and present, east and west', in Machlup and Mansfeld, *The study of information*, pp. 441–4; see especially p. 442.
7. As in Simon Nora and Alain Minc's *L'Informatisation de la société* (Paris, 1978), translated into English under the title *The computerisation of society* (Cambridge, Mass., 1981). To characterise the unprecedented capabilities of computers linked to telecommunications, Nora and Minc coined the term *télématique.*
8. Meanwhile in Germany between 1934 and 1945 Konrad Zuse independently and single-handedly recapitulated the development from mechanical calculator through relay machine to electronic, stored-program device in his computers $Z1$–$Z4$. The last of these formed the basis for the early computing program at Zurich's ETH.
9. *Proc. Lond. Math. Soc.*, series 2, 42 (1936), 230–65.
10. Cf. Wiener, *Cybernetics* (1961), p. 12: 'If I were to choose a patron saint for cybernetics out of the history of science, I should have to choose Leibniz. The philosophy of Leibniz centers about two closely related concepts – that of a universal symbolism and that of a calculus of reasoning.'
11. Turing, 'On computable numbers', p. 259.
12. SAGE = Semi-Automatic Ground Environment. The computer was an outgrowth of the massive Whirlwind computer built by Jay Forrester at MIT in the late 1940s.
13. Marvin L. Minsky, 'Computer science and the representation of knowledge', in Michael L. Dertouzos and Joel Moses (eds.), *The computer age: a twenty-year view* (Cambridge, Mass., 1979), pp. 392–421; esp. pp. 401–2. Recent developments in parallel distributed processing (PDP) have revived research in neural networks, overriding some of Minsky's conclusions.
14. Allen Newell has made a stimulating attempt to sort out the main lines in 'Intellectual issues in the history of artifical intelligence', in Machlup and Mansfeld, *The study of information*, pp. 187–227.
15. While the agendas of AI in the United States and elsewhere have looked much the same, the research communities differ on the question of language. *Lingua franca* for Americans is John McCarthy's LISP (for LISt Processing), an outgrowth of his work on the mathematical foundations of thought which was better suited than procedural languages to expressing the patterns of inference and search that researchers were investigating. During the 1970s European and Japanese investigators turned increasingly to PROLOG (Colmerauer and Roussel, 1972), aimed at facilitating the development of programs based on logical relations.

FURTHER READING

S. J. Heims, *John von Neumann and Norbert Wiener* (Cambridge, Mass., 1980).
Andrew Hodges, *Alan Turing: the enigma of intelligence* (London, 1983).
Vernon Pratt, *Thinking machines. The evolution of artificial intelligence* (Oxford, 1987)