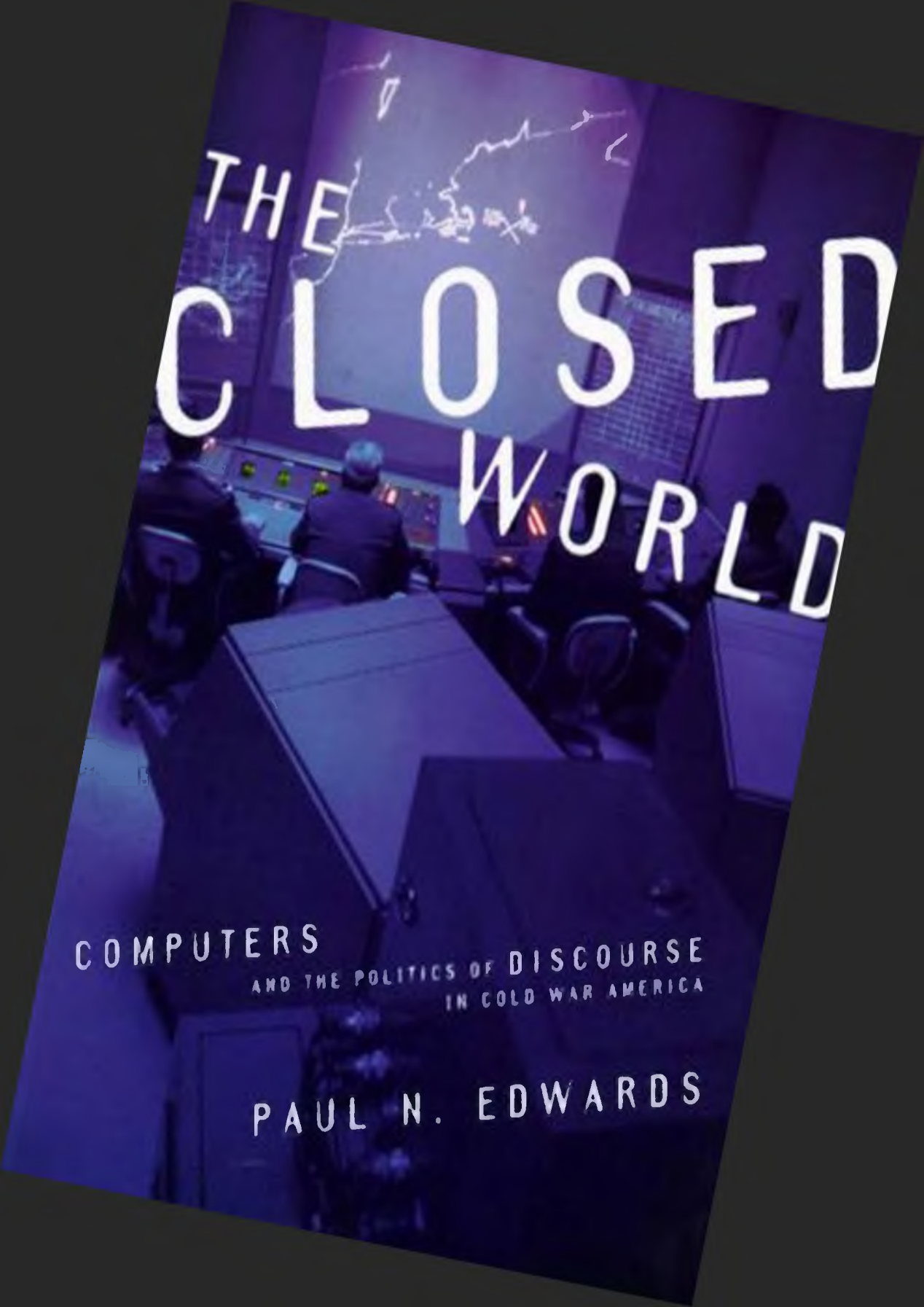

THE CLOSED WORLD
COMPUTERS
AND THE POLITICS OF DISCOURSE
IN COLD WAR AMERICA
PAUL N. EDWARDS

The Closed World

Inside Technology
edited by Wiebe E. Bijker, W. Bernard Carlson, and Trevor Pinch

Wiebe E. Bijker, *Of Bicycles, Bakelites, and Bulbs: Toward a Theory of Sociotechnical Change*

Wiebe E. Bijker and John Law, editors, *Shaping Technology/Building Society: Studies in Sociotechnical Change*

Stuart S. Blume, *Insight and Industry: On the Dynamics of Technological Change in Medicine*

Louis L. Bucciarelli, *Designing Engineers*

Geoffrey C. Bowker, *Science on the Run: Information Management and Industrial Geophysics at Schlumberger, 1920–1940*

H. M. Collins, *Artificial Experts: Social Knowledge and Intelligent Machines*

Paul N. Edwards, *The Closed World: Computers and the Politics of Discourse in Cold War America*

Pamela E. Mack, *Viewing the Earth: The Social Construction of the Landsat Satellite System*

Donald MacKenzie, *Inventing Accuracy: A Historical Sociology of Nuclear Missile Guidance*

Donald MacKenzie, *Knowing Machines: Essays on Technical Change*

The Closed World
Computers and the Politics of Discourse in Cold War America

Paul N. Edwards

The MIT Press
Cambridge, Massachusetts
London, England

This book was set in Baskerville by Pine Tree Composition, Inc. and printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Edwards, Paul N.
The closed world : computers and the politics of discourse in Cold War America / Paul N. Edwards
p. cm.—(Inside technology)
Includes bibliographical references and index.
ISBN 0-262-05051-X (alk. paper)
1. Computers—History. 2. Military art and science—Data processing—History. I. Title. II. Series.
QA76. 17.E34 1996
306.2—dc20 95-246
CIP

For Gabrielle

8

Constructing Artificial Intelligence

Where cognitive psychology analyzed human intelligence as an information process—human minds as cybernetic machines—artificial intelligence (AI) sought information processes that could exhibit intelligent behavior—cybernetic machines as thinking minds. AI established a fully symmetrical relation between biological and artificial minds through its concept of "physical symbol systems." It thus laid the final cornerstone of cyborg discourse.

Even more than that of cognitive psychology, AI's story has been written as a pure history of ideas.[1] AI inherited the ambitions of cybernetics, and like the cyberneticians, the founders of AI have been much concerned to document their own history, which they (quite naturally) view mainly in intellectual terms. Yet AI, too, has a prehistory, tightly linked to that of cybernetics, that dates to World War II and before. This chapter, like the last two, sets the birth of a new science against a wider background of postwar practical needs, political discourses, and social networks. Rather than rehearse the standard stories, I focus here on what has been obscured by the canonical accounts.

Instead of modeling brains in computer hardware—the central goal of cybernetics—AI sought to mimic minds in software. This move from biological to symbolic models has usually been interpreted as an abrupt intellectual break, a sudden shift in orientation from process to function. I see, on the contrary, a more gradual split. With the possible exception of John McCarthy, all of AI's founders were significantly influenced by the biological models of cybernetics. Even as late as the 1956 Dartmouth conference, the birthplace of AI as a systematic research program, the rift between brain modeling and symbolic processing remained incomplete.

I am going to argue that this shift occurred for reasons that had as

much to do with the practical problems and subjective environments of computer use as with purely theoretical concerns. Links, almost fortuitous in nature, between AI and time-sharing systems brought the budding discipline into close connection with military projects for human-machine integration in command-control systems. Because it tied AI to a realizable development program with a wide range of practical uses, this connection led to AI's most important institutional support: the Information Processing Techniques Office (IPTO) of the Advanced Research Projects Agency (ARPA). IPTO's founder, PAL and SAGE veteran J. C. R. Licklider, aggressively promoted a vision of computerized military command and control that helped to shape the AI research agenda for the next twenty-five years.

From Cybernetics to AI: Symbolic Processing

The founders of cybernetics were mostly mathematicians, like Wiener and Pitts, working with brain scientists, like Rosenblueth and McCulloch. While their theories of information and communication succeeded in erasing the boundary between humans and machines, allowing the construction of cyborgs, the cyberneticians themselves remained largely within the perspective of the mechanical. From W. Ross Ashby's *Design for a Brain* and Grey Walter's mechanical tortoises to von Neumann's self-reproducing factories and Shannon's maze-solving mechanical mice, the cyberneticians always ultimately sought to build *brains*. For them the formal machine itself, the logical model, was first and foremost a *machine*, an actual device with fixed properties and hard-wired channels for inputs and outputs. Even "the things that bothered von Neumann" at the Macy conferences had to do with whether a computer, despite its physical differences, could fruitfully be compared to the human brain; he spent his last days worrying over that very question.[2] The principal methods of cybernetics as practice, like those of its behaviorist antecedents, built complex structures out of simple, low-level, physical units such as neurons or reflex arcs. Frequently, based on then-current beliefs about how the brain worked, cyberneticians tried to design self-organizing machines that would achieve complex behavior through encounters with their environments. The subject, in both senses, of cybernetics was always the *embodied* mind.[3]

The next intellectual generation, the students of McCulloch, Pitts, Shannon, and Wiener, took one further step. They placed the em-

phasis of formal-mechanical modeling on the side of the formal, the disembodied, the abstract—on the side of the mind rather than that of the brain. Some of them were too young to have been shaped by the wartime experience of combining academic science with military engineering; for this generation, computers appeared not primarily as tools for solving practical problems, but as automated mathematical models with a powerful intellectual appeal. The key to a truly comprehensive theory of intelligence lay, for them, in the notion of "symbolic processing." This idea, in turn, acquired concrete standing and directionality through the craft activity of computer programming. This direction in computer evolution had been foreshadowed by Alan Turing's earliest work.

The Turing Machine

Turing's paper "On Computable Numbers, with an application to the *Entscheidungsproblem*," was published in 1937. It proved an ingenious solution to the last unsolved prong of mathematician David Hilbert's three-part challenge concerning the ultimate foundation of mathematical reasoning. Hilbert had asked whether mathematics was complete (whether every mathematical statement was either provable or disprovable), whether it was consistent (whether two contradictory mathematical statements could never be reached by a valid sequence of steps), and whether it was decidable (whether some mechanically applied method, which is to say some formal machine, could guarantee a correct analysis of the truth of any mathematical assertion). Kurt Gödel's famous theorem of 1930 had answered the first two questions, demonstrating that arithmetic and all other mathematical systems of comparable richness are incomplete, and that they cannot be proved consistent. Turing's paper answered the third, proving additionally that mathematics is not decidable—that some mathematical problems are not susceptible of any algorithmic solution.

Of greatest practical importance in Turing's work, however, was not the mathematical result but the unique method he devised to reach it. Turing proposed an imaginary machine something like a typewriter, operating on a paper tape of infinite length divided into unit squares. Each square would either be blank or contain a single mark. The machine's operations would be limited to "reading" the tape (recognizing blanks and marks); erasing marks; writing marks in blank squares; advancing to the right or returning to the left on the tape; and stopping.

Using only these basic operations, the machine could be "configured" in an infinite number of ways. For example, it could be designed to move to the right upon encountering a marked square, fill in the first blank square it came to, and continue to the right until it encountered a second blank, at which point it would move one square to the left, erase that square, and stop. This operation would make a single string of marked squares from two adjacent strings, preserving the total number of marks—the equivalent of adding two numbers. Such a Turing machine could function as an adding machine. An infinite number of other configurations are possible, and ultimately, any rule-based symbolic operation could be modeled by the machine. Many of the key elements of the electronic digital computer were already present in this purely hypothetical device: binary logic (squares are either blank or marked), programs ("configurations"), physical "memory" (the tape), and basic reading and writing operations.

Turing proved that this extremely simple machine, given the appropriate finite sequence of "configurations" or instructions, could solve almost any precisely specified symbolic problem that had a solution.[4] In other words, we can simulate almost any mathematical operation, of any degree of complexity, with a Turing machine by reducing it to a series of simple steps. Thus the Turing machine can also simulate any other logic machine. Hence most actual digital computers, while quite different in design from the primitive tape reader of Turing's paper, are in fact specimen Turing machines. The Turing machine is thus a conceptual bridge between the universalist ideas of logical automata (developing since Babbage and Boole), the metamathematics and metalogic of Hilbert, Gödel, and the logical positivists, and the practical problem of numerical computing machinery.[5]

Even in this early work, Turing had considered the relationship between the infinite set of "configurations" of his simple machine and the mental states of human beings. A human "computer" (the word was not applied to machines until the mid-1940s) performing the operations of a Turing machine by hand would necessarily, on his view, proceed through a sequence of discrete "states of mind" directly parallel to the states of the machine. "The operation actually performed is determined . . . by the state of mind of the [human] computer and the observed symbols. In particular, they determine the state of mind of the computer after the operation is carried out. . . . We may now construct a machine to do the work of this computer."[6]

Elsewhere in the paper Turing made it clear that the essential move in his analogy was to reduce each state of mind of the (human) computer to its atomic units. This could be achieved by breaking down any complex operation the person performed into a series of definite steps—precisely the basic operating principle of the Turing machine. A computer programmed to carry out the same steps would thus be doing exactly the same thing as the person. Hence its activity would be at least directly analogous, and perhaps even identical, to the series of mental states experienced by its human counterpart in performing the same operation.

Despite its cogent formulation of the basic principles of machine computation, however, "On Computable Numbers" had very little effect on the invention of actual computers in America. Most of the early computer pioneers read Turing's paper, if they read it at all, only after their own work was done.[7] The sole exception was John von Neumann, who met Turing in Cambridge, England, and who also became the only one of the early computer pioneers to explore the computer-brain analogy in a serious way. Instead, Turing's first strong influence on American thought occurred through the cybernetics work of McCulloch and Pitts, whose theory of the brain as a logical automaton was partly inspired by "On Computable Numbers."

Turing's explorations of the mind-machine analogy were a minor, if significant, element of his original paper. But over the next twelve years these analogies grew more full-blooded, coming eventually to occupy a central place in Turing's thought. In World War II Turing found an opportunity to build real versions of his hypothetical machine at Bletchley Park (see chapter 1). By 1948 he had composed a technical report for the British National Physical Laboratory entitled "Intelligent Machinery." This essay relied heavily on the same behaviorist picture of the brain used by McCulloch, Pitts, and the cyberneticians. Namely, the brain was a self-organizing machine that required extensive experience and training (analogous, for Turing, to programming) in order to realize its potential for intelligence. At this stage, Turing's focus on learning gave his ideas about intelligent machines a quasi-biological cast, while his method of symbolic programming more closely resembled what AI would eventually become. Thus this work occupied a transitional place between the brain models of cybernetics and the symbolic processors of artificial intelligence.

For the future of psychology and AI, Turing's most important work was unquestionably his 1950 article in *Mind*, "Computing Machinery

and Intelligence." This paper, by proposing the Turing test, effectively focused most of the future debate about machine analogs to human thought processes. The Turing test's hypothetical computer could interpret and compose not only numbers and logical notation, but also *written language*. The computer became far more than a calculator—it was a symbol processor par excellence, a universal information machine.

No machine then in existence could produce anything remotely resembling the performance demanded by the Turing test. Nor did Turing provide even a hint of how such abilities might be created. Yet the concept of communicating with computers in ordinary language foreshadowed the evolution of computer software over the decade 1945–1955. During that period computer languages grew increasingly sophisticated and increasingly removed from the level of digital computation, albeit for practical reasons and not primarily because of Turing-like goals. Not until high-level languages began to become available did Turing's vision of a computerized mind become something more than a speculation or a dream. To understand how and why this happened, we must briefly review how computers work.

Symbolic Computing: Levels of Description

The operation of computing machinery can be described at a number of levels. The "lowest" of these is the electronics of hardware: electronic switches, resistors, magnetic storage devices, and so on. Descriptions are causal and physical: electrons, currents, voltages. The second level is digital logic. The hardware electronics are designed to represent logical or mathematical operations, such as "AND" or the addition of binary digits or "bits"; the hardware itself is thus described as a set of logic "gates." Descriptions are logical, not physical, but they are still tied to the hardware itself, whose structure determines how each operation affects its successors. At a third level lies the "machine language" of the programs that "run" on a particular machine. Machine language consists of the binary representation of program instructions—the language the machine itself "speaks." Machine language also remains tied to a particular machine; programs written in one machine's language generally will not run on a different machine. However, machine-language programs are not a fixed part of the hardware (like its logic gates), determined by its physical structure; they are user-determined instructions. Assembly languages, consisting

of alphanumeric mnemonics for machine language instructions, constitute yet another hardware-dependent tier.

At an even higher level of description are compiler programs. Compilers are metaprograms, originally written in machine language, that translate programs written in "user-oriented" or "high-level" languages (now usually just called "computer languages," such as Pascal, Ada, and C) *into* machine language. Compilers make high-level languages "machine-independent"; given a compiler, the same program in a high-level language may run on any machine, irrespective of its particular electronics, digital logic, or machine language. (High-level languages cannot, in fact, function without compilers, since only a machine-language program can actually run a computer.) By translating statements in high-level languages into the calculations of machine language, compilers also create the possibility of easy symbolic manipulation.

High-level languages *may* be mathematical, like the algebraic languages FORTRAN or ALGOL, but they may also be symbolic—that is, they may take the form of language or logic, as in the business language COBOL or the AI language LISP. At the highest level of all lie another set of metaprograms—user interfaces and operating systems—usually themselves written in some high-level computer language. These metaprograms control how users interact with the machine, managing its processors, memory, disk, and other resources. Examples include the Unix operating system and the Windows or Macintosh graphical user interfaces.

These nested levels are relatively opaque to one another. In general, those concerned with one level need not know anything about levels "below" or above the ones they use. Programmers need not know how the interface works or concern themselves with the particular hardware of the machines on which their programs will run. Designers of user interfaces need only understand the high-level computer language in which they compose their programs, not those of compilers, machine languages, digital logic, or chip hardware. Chip designers need not understand or even consider compilers or high-level languages. Someone writing a program to manipulate symbols, such as a word processor, need not comprehend how the compiler translates each instruction into a series of calculations in binary arithmetic.

Each level is *conceptually* independent of the ones below and above, while remaining *practically* dependent on the lower levels.[8] Higher

levels are "reducible" to lower levels in the brute sense that any higher-level operation may be described as a definite series of lower-level operations. But while the cliché that computers understand nothing but ones and zeroes is true in this sense, it is false in the more important sense that "what the computer is doing" *for its users* is defined by the higher levels of description—for example, by the computer language and the user's goals and methods. This insight into the possibility of a symbolic, machine-independent level of description in computing was the conceptual foundation of artificial intelligence.

As with other issues, such as the advantages of digital over analog computing, the practical conditions of computer development obscured the possibility of symbolic computing foreseen by Turing and others. To the majority of computer designers and users in the first decade of electronic computing, the conceptual relationships I have just described were scarcely visible. An intellectual history might say, anachronistically, that those practical conditions simply delayed the genesis of symbolic computing, a development that merely played out an inevitable conceptual logic. But in fact symbolic computation did *not* emerge mainly from theoretical concerns. Instead, its immediate sources lay in the practice of the programming craft, the concrete conditions of hardware, computer use, and institutional context, and the metaphors of "language," "brain," and "mind": in other words, the discourse of the cyborg.

Writing Programs, Building Levels: Programming and Symbolic Computing

The (extremely limited) memory capacity and other hardware features of early digital computers placed a premium on efficient algorithms and compact program code. In the ENIAC, programming still involved the mechanical operation of physically interconnecting the machine's registers by means of plugboards. From the EDVAC on, programs themselves were coded as binary numbers and stored in memory along with the data. In their case the numbers stood for instructions (e.g., ADD *a* TO *b*) or addresses in memory (i.e., the location where the values of *a* or *b* were currently stored), thus incorporating the fundamental insight of Gödel and Turing that algorithms could be written in the same language as the data upon which they would operate. Until the late 1940s all program code was written in machine language; any numerical data had to be converted from decimal to binary form before the machine could process it. Because

it consisted entirely of concatenations of only two symbols (ones and zeroes), machine language was extremely difficult to use and even more difficult to debug.

In 1949 Eckert and Mauchly introduced Short Code (for the ill-fated BINAC). Employing alphanumeric equivalents for binary instructions, Short Code constituted an "assembly language" that allowed programmers to write their instructions in a form somewhat more congenial to human understanding. In Short Code, to represent the equation "$a = b + c$" one would write the instruction "S0 03 S1 07 S2," where S0, S1, and S2 stood for a, b, and c respectively; 03 meant "equal to"; and 07 signified "add."[9] A separate machine-language program called an "interpreter" translated Short Code programs into machine language, one line at a time. Interpreters were quick and efficient, and they elevated the symbolic level of programming a notch upwards toward algebraic language. But instructions still had to be entered in the exact form and order in which the machine would execute them, which frequently was not the conventional form or order for composing algebraic statements. It also required breaking down every complex operation into simple components.

The earliest programmers were primarily mathematicians and engineers who not only programmed but also designed logic structures and/or built hardware; for this tiny group, interpreters often seemed adequate to the job. Among them, the mathematical aesthetic of brevity-as-elegance, also motivated by the economics of memory and computer time, prevailed. Short, efficient algorithms and highly mathematical code ruled this culture's values. (As late as the middle 1960s at MIT, such an aesthetic reigned among the hackers who wrote much of the operating system software and many important utility subroutines for MIT's computer systems, even though by then higher-level languages had become the norm.)[10] Somewhat ironically, perhaps, programming a computer became a kind of art form.

But in order for nonexperts to write computer programs, some representation was required that looked much more like ordinary mathematical language. Higher-level languages, in turn, required compilers able not merely to translate statements one-for-one into machine code, as interpreters did, but to organize memory addressing, numerical representation (such as fixed- or floating-point), and the order of execution of instructions. (Frequently a single instruction in a higher-level language will be compiled into a dozen or more

machine-language instructions.) Such programs required what then amounted to exorbitant quantities of memory and machine time.

Philip Morse (whose NDRC Committee on Sound Control in Aircraft had spawned the Psycho-Acoustic Laboratory during the war) chaired MIT's Committee on Machine Methods of Computation in the early 1950s and was named director of its Computation Center in 1955. He recalled that bitter struggles over higher-level languages sprang from the aesthetics of programming and the economics of machine time and memory.

> The argument over the desirability of compiler programs was intense and acrimonious. The hardware experts were horrified. It was inefficient, they protested; every computation would have to run twice through the machine, once using the compiler to prepare a program in machine language, then running this program to direct the machine to do the actual calculation. In addition, the compiler itself... would have to be a program in machine language, and it would have to be the great-grandfather of all programs, because it would have to foresee all the different operations used in any sort of computation and would have to guard against all the logical errors that might occur. It would take dozens of man-years of the best programmers' time to write one, and, when written, it would be so big there would not be room for anything else in the machine's memory.[11]

At first, these predictions proved accurate enough. Compilers *did* produce inefficient machine code. The first compilers, written between 1950 and 1953, slowed even huge, superfast machines such as Whirlwind to a veritable crawl. Furthermore, debugging a compiled program often required sorting through the compiler-generated machine code itself, a task rendered more difficult by the fact that the code followed the machine's logic rather than that of a human author.

As sales of commercial computers mounted rapidly, so did the number of programmers. In 1950, with only a few computers in existence, there were only a few hundred programmers. Most of these people would not have identified themselves as such, since they also played other roles in machine design or construction. By 1955 the one thousand or so extant general-purpose computers required the services of perhaps 10,000 programmers. Five years later, in the midst of a booming commercial computer market, programming had suddenly become a profession in its own right, with about 60,000 practitioners servicing some five thousand machines. Programming began to emerge as a craft, a specialized practice; already the amount of

mathematical skill it required had begun to diminish. In the words of Michael Mahoney, it was "technical rather than technological, mathematical only in appearance."[12]

As we saw in chapter 3, programming for the SAGE system contributed its massive momentum to this process through the Systems Development Corporation and its many spin-offs. SDC hired and trained thousands of new programmers beginning in the mid-1950s. Music teachers and women without specialized backgrounds were among the most successful recruits.[13] Such groups could not be expected to learn machine code or produce mathematically elegant algorithms; to make this new work force effective required symbolic languages easily learned by nonspecialists. Businesses, too, wanted to write their own software without hiring expensive experts. Not only did they want to do this, they had to if they were going to use computers at all, since before the 1960s "off-the-shelf" software was virtually unknown. Software had to be written specially for each type of machine and, in many cases, for each *individual* machine. The exponentially increasing demand for software, and thus for nonspecialist programmers, helped drive the movement toward higher-level languages.

The first true algebraic-language compiler was written in 1953 for Whirlwind, but its slow speed and the unavailability of other equally capable machines prevented its widespread use. In an attempt to kick-start the field, the Office of Naval Research sponsored symposia on "automatic programming" in 1954 and 1956. The name "automatic programming" for compilers is itself revealing. "Programming" still referred to the composition of the machine-language instruction list. Algorithms written in a higher-level language were not yet seen as actual programs, but rather as directions to the compiler to compose a program; compilers thus performed "automatic programming." The independence of symbolic levels in computing had not yet achieved the axiomatic status it later acquired.

The first commercially viable higher-level language was FORTRAN, an algebraic, scientific programming language. IBM researchers completed the FORTRAN compiler in 1956–57. In 1959, at the request of a consortium of universities, computer users, and computer manufacturers, the Defense Department convened a Conference on Data Systems Languages (CODASYL). CODASYL soon produced specifications for the English-like, data-processing-oriented high-level language COBOL (for COmmon Business-Oriented Language).

Among the most durable of all computer languages, COBOL still finds widespread use today. Though COBOL was designed and used primarily for business data processing, the DoD's leadership in the CODASYL standard-setting reflected its continuing status as the major customer and primary supporter of computer research.[14]

Though programmers developed user-oriented computer languages for purely practical ends, the process of creating these languages generated awareness of the potential of computers as manipulators, not just of numbers, but of symbols of *any* type. The compilers turned English-like commands such as PRINT, ADD, and GOTO into binary code. Thus while the *theoretical* possibility of computers conversing in natural language had been raised by Turing and others in the 1940s, it was in fact the *practical* work of composing software that led to the first symbolic languages. Even for many of AI's founders, the idea of the computer as embodying a universal representational system, a "language of thought" in a later phrase,[15] emerged not abstractly but *in their experiences with actual machines*, as both users and developers of higher-level languages.

Intelligence as Software

Allen Newell and Herbert Simon, as we saw in chapter 4, both recalled their first experiences with programming computers to simulate live radar screens as moments of epiphany in which they perceived computers as symbol systems. Another epiphany for Newell was a November 1954 Rand presentation by Oliver Selfridge, who was then working at Lincoln Laboratories. Selfridge had been Norbert Wiener's assistant, in which capacity he had proofread the manuscript of *Cybernetics* and attended some of the Macy cybernetics conferences. At Lincoln, Selfridge was working on the computerized pattern-recognizing system that later evolved into the highly influential "Pandemonium,"[16] a device that recognized letter forms and simple shapes. To accomplish this, a number of subprocesses analyzed various features of a figure; they then, in effect, "voted" for the result by computing a value and comparing it with a set of norms. In their use of a large number of simple processors that combined values to yield an outcome, and especially in their ability to "learn" by adjusting their own functions, Pandemonium and its predecessors resembled the neural nets of McCulloch and Pitts. At the same time, they were symbolic models in the sense that each subprocess used logic to ana-

lyze and categorize features. Thus, like Turing's speculations, Selfridge's programs occupied the transitional space between brain models and symbolic information processing.

Newell recalled the Selfridge presentation as utterly galvanizing: "I can remember sort of thinking to myself, you know, we're there.... And [Selfridge's program] turned my life. I mean that was a point at which I started working on artificial intelligence. Very clear—it all happened one afternoon."[17] Despite its cybernetic flavor, what Newell took from Pandemonium was the concept of symbolic processing. He began working on chess programs. By the time of their Logic Theorist experiments, Newell, Simon, and Shaw were also developing their own symbolic computer language, the IPL (Information Processing Language) series of list processing languages.

Edward Feigenbaum, one of Simon's students and later the major exponent of the "expert systems" approach to AI, remembered that when Simon handed out IBM 701 manuals to his course in early 1956, he went home and "read it straight through, like a good novel."[18] Feigenbaum went on to intern at IBM that summer, down the hall from the team working on the still-unreleased FORTRAN, and to help develop IPL-IV in the summer of 1957 at Rand. John McCarthy, as we shall see, also developed a list processing language. For all of these AI founders, the experience of symbolic programming constituted a significant step toward their vision of intelligent symbolic machines.

In symbolic processing the AI theorists believed they had found the key to understanding knowledge and intelligence. Now they could study these phenomena and construct truly *formal-mechanical* models, achieving the kind of overarching vantage point on both machines and organisms that cybernetics had attempted to occupy. Newell and Simon, first of the post-cybernetic AI theorists, rapidly raised to axiomatic status the idea that intelligence was a *symbol manipulation process* capable of being modeled by a computer.

Following on the methods used in Newell's experimental work on air defense simulation, Newell and Simon began around 1955 to work with "protocols" in studies of human problem-solving behavior. To determine a protocol, the experimenters would pose a complex logic problem to a subject and ask him to talk while he attempted to solve it, describing every one of his thoughts and perceptions as best he could. A collection of individual protocols would often reveal common strategies and techniques. Following Turing's plan of reducing each

"state of mind" to a sequence of atomic procedures, these could be specified as lists of instructions, programmed on the computer, tested, and refined. One result of this research program was the 1957 General Problem Solver (GPS), a program that could solve certain formal puzzles such as cryptarithmetic equations and the missionary-cannibal problem. In Newell and Simon's view, programs like the GPS simultaneously constituted both a psychological theory and a rudimentary form of artificial intelligence.

To write effective protocols, Newell and Simon did not need access to lower-level processes such as neural nets, just as with the IPLs they no longer required access to the computer's machine language to compose programs. As long as the program produced results similar to those of the human problem-solver, its purely symbolic level of description was all that mattered. In their first article directed at academic psychologists, they stressed the point: "our theory is a theory of the information processes involved in problem-solving and *not a theory of neural or electronic mechanisms* for information processing."[19] Ultimately Newell and Simon established this principle as an axiom of both AI and cognitive science: their "physical symbol system hypothesis," under which people and all intelligent entities were essentially active symbol systems, physically instantiated.[20]

Newell and Simon thus began to displace the cybernetic computer-brain analogy with the even more comprehensive and abstract computer-*mind* metaphor of artificial intelligence. The two metaphors shared concepts of coding and information. But where the cyberneticians' ideal systems were weakly structured, self-organizing, and engaged with the environment, AI systems were highly structured, manipulating pre-encoded and pre-organized knowledge rather than building it through sensory encounters. Instead of feedback, reflex, and neural networks, the AI theorists thought in terms of instructions, languages, goals, and logical operations. The physical machine became little more than an arbitrary vehicle for the interactions of pure information.[21]

The Dartmouth Conference

In 1956, the Summer Research Project on Artificial Intelligence met for two months at Dartmouth College. The Dartmouth conference is generally recognized as the conceptual birthplace of AI (though Newell and Simon's work in fact predates it). The second MIT Sym-

posium on Information Theory, from which George Miller dates the inception of cognitive psychology, occurred a few months later, and its protagonists included many of the same people.

The Dartmouth institute's chief organizer was John McCarthy, who coined the term for the grant proposal. As a twenty-year-old undergraduate mathematician, McCarthy had literally wandered into the 1948 Hixon Symposium at the California Institute of Technology, where he heard von Neumann talk on automata theory. He claims to have conceived the idea of artificial intelligence then and there (though not the phrase itself). He spoke to von Neumann about the idea, receiving the curt reply, "Write it up."[22] Other organizers included Claude Shannon, Marvin Minsky, and Nathaniel Rochester of IBM. Both McCarthy and Minsky had worked for Shannon at Bell Laboratories in the summer of 1952, and Shannon and McCarthy had already edited a volume on *Automata Studies* together.[23] Newell and Simon attended the conference as well, bringing with them their already-operating Logic Theorist and their computer language IPL.[24]

In the proposals for work at the conference, the still-nascent split between the computer-brain and computer-mind metaphors already appears clearly. For example, Shannon, representing the cyberneticians, planned to do brain modeling (using information theory). But McCarthy, in his work on the *Automata Studies* volume, had increasingly come to see Shannon's agnostic approach toward meaning as conservative and restricting. He felt that automata studies would never lead to true artificial intelligence since it deliberately avoided what was, for him, precisely the central issue. (This was in large part why he had insisted on the more radical and provocative term "artificial intelligence," over Shannon's objections, when organizing the conference.)

McCarthy's own goal for the summer was

> to attempt to construct an artificial language which a computer can be programmed to use on problems requiring conjecture and self-reference. It should correspond to English in the sense that short English statements about the given subject matter should have short correspondents in the language and so should short arguments or conjectural arguments. I hope to try to formulate a language having these properties and in addition to contain the notions of physical object, event, etc.[25]

At the time of the conference, Marvin Minsky had just made a transition from mathematics and neurology in the McCulloch-Pitts line to

symbolic models. With Dean Edmonds, Minsky's first major project, as a Harvard undergraduate, had been a working simulation of a neural network based on the neurological learning theory of the influential psychologist D. O. Hebb.[26] The machine, essentially an analog computer, used three hundred vacuum tubes, motors, potentiometers, and automatic clutches to simulate a highly interconnected set of neuron-like elements with "learning" abilities. Following on this experiment, Minsky had written a dissertation on *Neural Nets and the Brain-Model Problem.*[27] By the time of the Dartmouth conference, however, he had become disenchanted with the neural-network approach, in part under Shannon's influence and in part because it had become "boring" since "it didn't work."[28] The appeal of symbolic processing, for him, had a practical flavor: computer programs, unlike his analog brain models, could *do interesting things* with symbols, whether or not they resembled how brains worked.

Minsky wanted to use the Dartmouth institute to develop a geometry theorem-proving program something like the Newell-Simon-Shaw Logic Theorist. He circulated drafts of what later became his influential paper "Steps Toward Artificial Intelligence," essentially an essay-review of significant work on symbolic processing, including search, pattern recognition, learning systems, problem-solving, and heuristic programming. The paper noted but chose not to discuss brain modeling research as less significant to AI than symbolic processing.

McCarthy soon went on to develop the primary artificial intelligence programming language LISP (LISt Processing), known for its high level of symbolic abstraction. LISP manipulates lists (of symbols of any sort) and lists of lists; programs written in LISP are themselves lists. The language is highly recursive, meaning that definitions of terms may include those terms and that program processes may invoke themselves.[29] Among LISP's chief advantages is its ability to manipulate terms with many complex interrelations, like the words of a natural language.

By 1957, the year after the Dartmouth conference, Newell and Simon had completed their General Problem Solver and firmly believed they were on the track of a general, *hardware-independent* model of intelligence. Symbolic processing was the key. Where the PAL psychologists had engineered human language to render it compatible with electrical communications systems, the AI theorists engineered

computer languages to make them compatible with human thought processes.

Newell and Simon were confident enough of the basic vitality of their approach to make four very strong predictions about their information-processing psychology. Within ten years, they claimed,

- a computer would be world chess champion,
- a computer would compose aesthetically valuable music,
- a computer would discover and prove an important unknown mathematical theorem, and
- most psychological theories would take the form of computer programs.[30]

An ultimate rapprochement between human and computer thought loomed on the horizon. Human mental software would be decoded and recompiled in machine language.

By 1958 McCarthy and Minsky had teamed up to establish an Artificial Intelligence Group at MIT's Research Laboratory of Electronics. That year McCarthy "proposed that all human knowledge be given a formal, homogeneous representation, the first-order predicate calculus."[31] Similar optimism reigned throughout the field. George Miller and Yale psychologist Carl Hovland had worked with Simon, Newell, and Minsky that summer at the Rand Research Training Institute on the Simulation of Cognitive Processes. Miller, Pribram, and Galanter were composing what would become *Plans and the Structure of Behavior*. Psychology, cognitive simulation, and artificial intelligence seemed increasingly to be parts of a single whole, united through the abstraction of symbolic processing. Cyborg discourse crystallized around its Foucaultian "support": the computer-centered research programs of AI and information-processing psychology.

In rejecting the cybernetic brain models and learning machines, AI also rejected a model of mind as inherently embodied. The brain-model approach relied intrinsically on interaction with the world; repeated experience, not formal analysis, was supposed to shape the weighted connections of neural elements into a functional system. Symbolic AI instead sought first to formalize knowledge of the world, injecting it into computer systems predefined and predigested. Logic, not experience, would determine its conclusions. (Yet this logical method reflected the AI founders' own experiences of the power of

symbolic programming.) In its Enlightenment-like proposals for an encyclopedic machine, AI sought to enclose and reproduce the world within the horizon and the language of systems and information.[32] Disembodied AI, cyborg intelligence as formal model, thus constructed minds as miniature closed worlds, nested within and abstractly mirroring the larger world outside.

Time-Sharing: Linking AI to Command and Control

At this point, through a rather convoluted series of historical connections, AI intersected with another practical concern of programmers: the availability of computer time. The linkages that coalesced around this tie proved extremely significant to AI's success as a research program. Without them, AI might have had a very different future.

Until the late 1950s all computers (with the sole, partial exception of the SAGE machines) were "batch processors." This meant that each program, with its data, was run on the machine as a unit or "batch"; while that program was running, the computer could do nothing else. However, the speed of input and output (I/O) devices was far lower than that of the computer's central processing unit (CPU). This meant that the CPU, the computer's heart and (then) its most expensive element, actually sat idle most of the time, while it waited for I/O units to do their jobs. The use of compiler programs further slowed "run" times.

Furthermore, programs usually had to be run many times before all errors were found and fixed. Since the debugging process was slower than CPU or I/O times by yet further orders of magnitude, after receiving their output and fixing their programs programmers would have to wait, frustrated, in a queue until the machine was again free. Finally, the relatively small number of available computers (especially in universities) meant intense competition for computer time.

Jay Forrester had foreseen the bottleneck created by the disparity between CPU, I/O, and human time scales as early as 1948. His group designed Whirlwind to mitigate the problem with a technique known as "multiprocessing," which allowed the CPU to perform a number of predetermined tasks simultaneously.[33] With a later technique known as "multiprogramming," several programs could be run at once under the direction of an "executive" program. But the multiprogramming systems, sometimes also called "time-sharing" systems,

were still designed with only a single user-operator in mind. More programs could be run faster under multiprogramming, but the essential approach remained batch processing.

To allow many users at once to take advantage of the computer's speed, John McCarthy conceived the modern concept of time-sharing around 1958. In time-sharing the computer (under one typical technique) partitions its memory into sections, each containing some program and its data. The CPU then cycles among these programs, performing one operation from each on every cycle. Many users connect to the same machine via remote terminal displays. Since the computer operates so rapidly, these users typically experience little or no delay between starting a program and receiving its results. Dozens of terminals may be connected to a single computer, but each user experiences the computer as a private domain.

Time-sharing (in McCarthy's sense) would produce two major changes in how computers were used. First, it would take full advantage of the CPU's speed, allowing one computer to perform work previously requiring many. Second, it would permit individual users to operate the computer "interactively"—privately, personally engaged with the machine, without the need for queues and delays between program runs. This, in turn, would create the possibilities of on-line debugging (fixing programs while they were running, with the effects of each change instantly visible to the operator), use of graphic displays rather than paper output, and a myriad of other "interactive" features.

At MIT in the late 1950s, time bottlenecks on computer systems had reached major proportions. MIT's Computation Center used the Whirlwind I until 1956, when IBM donated one of its 704 machines. This computer was shared by three users: MIT, IBM, and a consortium of New England area colleges and universities. IBM used the machine ten hours a day, while MIT and the university consortium each got a seven-hour shift. The 704 was replaced by the more powerful 709 in 1960. By this point MIT had also acquired several other computers, including an IBM 650 at its Instrumentation Laboratory and a Bendix G15 at the Naval Supersonic Laboratory. Lincoln Laboratory had donated its experimental TX-0 to the Research Laboratory of Electronics. Some, but not all, of these machines were available for general use, but demands for computer time increasingly outstripped this very limited supply.

By 1960, courses in computing, thesis work, data reduction, and

numerical processing devoured most of the available computer time. Yet symbolic processing—such as virtually all of the work McCarthy, Minsky, and their AI group wanted to do—required an increasingly large proportion (in 1960, 28 percent of total research computing time).[34] AI research, computer-aided design, machine translation, and library information retrieval systems all required ever-larger programs in symbolic languages. By early 1961 an MIT report on computer capacity, composed by a committee that included McCarthy, Minsky, and V. H. Yngve (working on machine translation of natural languages), foresaw "needs for extreme capacities in the way of memory sizes and operating speeds" in order to carry out research in "language translation and analysis . . . heuristic problem solving [i.e., AI] . . . and computer-aided design." The latter would require "real-time operation . . . since the system will use pictorial and written language forms through a real-time console." For AI, the report concluded, "to tackle problems of more than trivial interest, very large memory and high processing speeds are needed." By this point McCarthy's work on time-sharing was well along under an NSF grant. IBM had provided hardware modifications to MIT's 704 allowing his plans to be put into action, but this machine's capacities remained too limited to accommodate McCarthy's vision. The report accordingly recommended acquiring an "extremely fast" computer with a "very large core memory" and a time-sharing operating system as the solution to the time bottleneck.[35]

McCarthy's time-sharing idea bore no inherent connection to AI. It was simply an efficient kind of operating system, an alternative way of constructing the basic hardware and software that controls how the computer processes programs. Yet AI work influenced his invention of time-sharing in two major ways.

First, AI programs consumed vast quantities of memory and machine time. Time-sharing would allow AI workers more access to the essential tools of their trade. Second, and more importantly in this connection, McCarthy needed time-sharing to *provide the right subjective environment* for AI work. He promoted time-sharing "as something for artificial intelligence, for I'd designed LISP in such a way that working with it interactively—giving it a command, then seeing what happened, then giving it another command—was the best way to work with it."[36] McCarthy and many of his coworkers wanted not simply to employ but to *interact* with computers, to use them as "thinking

aids" (in their phrase), cyborg partners, second selves. They wanted a new subjective space that included the machine.

Through McCarthy's work, AI became linked with a major change not only in computer equipment but in the basic social structure and the subjective environment of computer work. Under time-shared systems, rather than submit their programs to what many referred to as the "priesthood" of computer operators, users would operate computers themselves from terminals in other rooms, other buildings—in their offices and, eventually, even at home. They would see programs operate before their eyes, rather than find their results hours later in a lifeless printout. They could engage with the machine, create new kinds of interaction, experience *unmediated* communication between human and computer.

Human-machine integration, driver of computer development from the beginning, would now be applied to computers themselves. In the process AI researchers would acquire the interactive access to machines McCarthy's vision of AI demanded. This connection between AI and time-sharing eventually led, as we shall now see, to AI's most important institutional relationship.

The Advanced Research Projects Agency

By the mid-1950s military research agencies were already not only aware of, but actively seeking out, work in the emerging field of computer simulation of cognitive processes. Though the Dartmouth institute's primary sponsor was the Rockefeller Foundation, the Office of Naval Research was also involved.

The ONR had long since decided that future military forces would require computer technologies for "decision support." As Marvin Denicoff, one of its leaders, later recalled, "In the early fifties, it occurred to some of us [at the ONR] that we ought to begin supporting decision makers where that decision maker could be an inventory specialist, a command and control specialist, a tactical officer, a battlefield officer, a pilot—any of many categories. Also, we should begin to go out to universities that had strong programs or emerging programs in computation and fund them."[37]

Denicoff's brief memoir stresses the similarity between the operations research (OR) methods familiar to the ONR's leaders and the emerging programming techniques of AI. Like OR, AI sought close fits, best approximations, and heuristic rules rather than the more

rigid full-solution methods of other techniques. When McCarthy planned the 1956 Dartmouth conference, the ONR gladly footed part of the bill. By then it had already sought out Newell, Simon, and others at Carnegie Tech and the Graduate School of Industrial Administration (where Simon taught): "We wrote a long-term contract essentially to explore new approaches to decision making with all of those people [from Carnegie] involved."[38]

Yet AI was only one of a vast number of human-machine relationships military sponsors sought to explore. It might have remained a minor part of this wider program had it not been for the Advanced Research Projects Agency (ARPA, renamed the Defense Advanced Research Projects Agency, DARPA, in 1972) and in particular for one person within ARPA: J. C. R. Licklider. In Licklider and his Information Processing Techniques Office, the closed-world military goals of decision support and computerized command and control found a unique relationship to the cyborg discourses of cognitive psychology and artificial intelligence.

The Eisenhower administration founded ARPA hastily early in 1958, in the aftermath of the 1957 Sputnik launch. It served as an interim space agency, a kind of holding tank for civilian and military space programs while the government devised a more permanent institution to lead the space race. In an attempt to neutralize bitter competition among the three services over which should control what kinds of missiles and space technology, the administration chose an institutional location under the central Office of the Secretary of Defense. When space programs were transferred to the newly founded National Aeronautics and Space Administration (NASA) in 1960, other development-oriented military programs (such as the Intercontinental Ballistic Missile) reverted to the services. This left ARPA, at the start of the Kennedy administration, in charge of a widely varied roster of experimental scientific and technological research projects.

ARPA's post-NASA mandate was broad and also unique. According to a recent history of the agency, its new role would be

> to advance defense technology in many critical areas and to help the DoD create military capabilities of a character that the Military Services and Departments were not able or willing to develop for any of several reasons: because the risks could not be accepted within the limits of the Service R&D and procurement budgets; because those budgets did not allow timely enough response to newly appearing needs; because the feasibility or military values of the new capabilities were not apparent at the beginning, so that the Services

declined to invest in them; or because the capabilities did not fall obviously into the mission structure of any one Service.[39]

As an early official chronicle notes, ARPA was "spawned in an atmosphere that equated basic research with military security."[40]

Because of this military association, ARPA-sponsored research was subject neither to peer review, like that of the National Science Foundation, nor to other traditional equalizing principles designed to distribute research money widely. In effect, ARPA reincarnated the World War II OSRD. The agency's small directorate of scientists and engineers chose research directions for the military based on their own professional judgment, with minimal oversight. ARPA concentrated its funding in a small number of elite "centers of excellence," primarily in universities. Lacking both heavy congressional oversight and the development orientation of the services, the agency could support far-sighted, high-risk projects that might help avoid "technological surprises" (in the phraseology of a later era) from other countries.[41] ARPA's tacit funding agreements could also span much longer terms than those of other grantor agencies. These features of ARPA sponsorship had the effect of permanently "addicting" ARPA-supported laboratories to Defense Department funding.[42]

The newly inaugurated president, John F. Kennedy, in a March 1961 message to Congress, called for stepped-up research on command and control.[43] In June, the Director of Defense Research and Engineering (DDR&E) assigned command and control research to ARPA under the heading of the ballistic missile defense project DEFENDER. (DEFENDER also included research on phased-array radar tracking, high-energy lasers, and a number of other technologies picked up again in the 1980s by the Strategic Defense Initiative.)[44]

Computers were to be the focal technology of the ARPA command-control effort, in part for reasons of pure expedience:

> DDR&E's problem appears to have been the existence of a rather expensive computer (the [transistorized] AN/FSQ-32 XD1A), built as a backup for the SAGE air defense program, which the Air Force had determined was no longer needed and hence was available for other purposes. There was also considerable interest within DDR&E in computer applications to war gaming, command systems studies and information processing related to command and control, as well as concern about the continued utilization of the System Development Corporation (the major software contractor for the Air Force), which apparently was experiencing some cutbacks in support due to the stage

of development of Air Force programs [i.e., the imminent completion of SAGE]. DDR&E thus had a major piece of computer hardware begging for use, strong interest in computer applications to command and control, and an available contractor asset with appropriate credentials.[45]

DDR&E assembled the pieces of this puzzle by giving ARPA the FSQ-32. ARPA then initiated a command-control project at SDC for something called the "Super Combat Center" (a kind of super-SAGE), simultaneously rescuing SDC and providing a use for the extra computer.[46] This new direction led to the creation of ARPA's Information Processing Techniques Office (IPTO) in 1962.

Thus in the aftermath of SAGE, by this rather circuitous route, cutting-edge computer research once again found support under the aegis of air defense, part of the increasingly desperate, impossible task of enclosing the United States within an impenetrable bubble of high technology.

J. C. R. Licklider

IPTO's first director was none other than Psycho-Acoustic Laboratory and Macy Conference veteran J. C. R. Licklider.

In the postwar years Licklider's interest in computing had grown into a career. As we saw in chapter 7, in 1950 he had organized the psychological side of the Lincoln Laboratory "presentation group" responsible for SAGE interface design, which also included PAL veterans and information-processing psychologists George Miller and Walter Rosenblith. Like his PAL colleagues, Licklider's concern with human-machine integration dated to his wartime work in psychoacoustics; his particular specialties had been high-altitude communication and speech compression for radio transmission. At MIT he joined the series of summer study groups, beginning with 1950's Project Hartwell on undersea warfare and overseas transport and continuing with the 1951–52 Project Charles air defense studies that led to SAGE. As he recalled later, the summer studies

brought together all these people—physicists, mathematicians. You would go one day and there would be John von Neumann, and the next day there would be Jay Forrester having the diagram of a core memory in his pocket and stuff—it was fantastically exciting. Project Charles was two summer studies, with a whole year in between, on air defense. At that time, some of the more impressionable ones of us were expecting there would be 50,000 Soviet bombers coming in over here.[47]

In his first years at MIT Licklider split his time between the Lincoln presentation group, MIT's new psychology program, and the Acoustics Laboratory.

The MIT Acoustics Laboratory, originally formed by Philip Morse, was then being directed by Leo Beranek, wartime head of the Harvard Electro-Acoustic Lab (the PAL's engineering arm) and Licklider's former colleague. The Acoustics Lab, within the Electrical Engineering (EE) Department, functioned as a "sister laboratory" of the Research Laboratory of Electronics (RLE, successor to the famed wartime Radiation Laboratory), whose programs had doubled in size "under strong military support" with the outbreak of the Korean War in 1950.[48] Licklider soon became deeply involved.

The early 1950s were a period of profound intellectual excitement in these laboratories. In the years following Licklider's arrival, the list of his RLE and EE Department colleagues reads like an almost deliberate merger of cyberneticians, Macy Conference veterans, Psycho-Acoustic Laboratory staff, and AI researchers. Norbert Wiener, Alex Bavelas, Claude Shannon, Walter Rosenblith, Warren McCulloch, Walter Pitts, Jerome Lettvin (a key McCulloch-Pitts collaborator), George Miller, John McCarthy, and Marvin Minsky were all part of the lab, as were the linguists Noam Chomsky, Morris Halle, and Roman Jakobson. As Jerome Wiesner later recalled, "the two decades of RLE [were] like an instantaneous explosion of knowledge." Wiener "was the catalyst," making daily rounds of the laboratory to investigate everyone else's research and to hold forth on whatever new idea had happened to seize his restless mind.[49] Licklider counted himself among the members of the cyberneticians' inner circle, attending Wiener's weekly Cambridge salons. He himself "exercised a profound influence on the growth of the EE Department," according to its official chronicles.[50]

In the Lincoln presentation group, the RLE, and the Acoustics Laboratory, Licklider learned about the most advanced digital machines and the problems of human-computer interaction (though his own work at the time involved building *analog* computer models of hearing). When Miller went to Lincoln to become a full-time group leader in 1953, Licklider remembered, "I stayed [at MIT]. Then I found out I really had to learn digital computing, because I couldn't do this stuff with analog computers, and there was no way, as a psychologist over in the Sloan Building, for me to get a digital

computer." He eventually "struck this deal with Bolt Beranek and Newman" (BBN) to use the firm's Digital Equipment Corporation PDP-1.[51]

BBN was a research and consulting organization established by Leo Beranek and two other members of the MIT Acoustics Laboratory. It was among the most successful of the MIT/Lincoln spin-off companies and is still a major force in advanced computing research. In 1957 Licklider left MIT to work at BBN full-time on "man-machine" research sponsored by the Air Force Office of Scientific Research.[52] He led BBN's Psychoacoustics, Engineering Psychology, and Information Systems Research Departments, and eventually he became the firm's vice-president. Around the time Licklider arrived, BBN had begun a large project on time-sharing systems in which both Minsky and McCarthy were involved. Licklider audited one of McCarthy's MIT courses and underwent, in his own words, a "religious conversion to interactive computing."[53]

The groundwork for this "conversion" had been long prepared by Licklider's experience with the SAGE system's interactive computers. In a different interview, he recalled: "I was one of the very few people, at that time, who had been sitting at a computer console four or five hours a day."[54]

"Man-Computer Symbiosis"

Licklider's paper on "Man-Computer Symbiosis," published in 1960 and bearing the visible stamp of McCarthy's ideas, argued that batch-processing computers failed to take full advantage of the computer's power. Computerized systems, he wrote, generally served an automation paradigm, replacing people; "the men who remain are there more to help than to be helped."[55] Licklider proposed a more integrated arrangement to which computers would contribute speed and accuracy, while men (sic) would provide flexibility and intuition, "programming themselves contingently," as he put it in a perfect cyborg metaphor.

By this point Licklider was clearly an AI partisan, though his expectations—echoing the 1943 robot cat example of Rosenblueth, Wiener, and Bigelow—reflected uncertainty about whether biocybernetic or computer methods would first reach the goal. "It seems entirely possible that, in due course, electronic *or chemical* 'machines' will outdo the human brain in most of the functions we now consider exclusively

within its province," he wrote, referencing the work of Newell, Simon, Shannon, Wesley Clark (designer of Lincoln Laboratories' TX-0 and TX-2 computers), and others.[56] Citing an Air Force study that had "estimated that it would be 1980 before developments in artificial intelligence make it possible for machines alone to do much thinking or problem-solving of military significance," he contended that while awaiting AI's maturity a regime of "man-computer symbiosis" could be attempted. The key to such a goal was time-sharing for real-time, interactive computing.

Licklider's example of the problems with batch processing shows the influence of his military concerns:

> Imagine trying . . . to direct a battle with the aid of a computer on such a schedule as this. You formulate your problem today. Tomorrow you spend with a programmer. Next week the computer devotes 5 minutes to assembling your program and 47 seconds to calculating the answer to your problem. You get a sheet of paper 20 feet long, full of numbers that, instead of providing a final solution, only suggest a tactic that should be explored by simulation. Obviously, the battle would be over before the second step in its planning was begun.

Calling upon both early AI and the SAGE system as precedents, he speculated that men and computers might become so tightly coupled that

> in many operations . . . it will be difficult to separate them neatly in analysis. That would be the case if, . . . for example, both the man and the computer came up with relevant precedents from experience and if the computer then suggested a course of action that agreed with the man's intuitive judgment. (In theorem-proving programs, computers find precedents in experience, and in the SAGE system, they suggest courses of action. The foregoing is not a far-fetched example.)

High-level symbolic computer languages were another key to man-computer symbiosis. Procedural languages such as FORTRAN and ALGOL should be supplanted by goal-statement languages more similar to those of human beings; here Licklider mentioned the work on "problem-solving, hill-climbing, self-organizing programs," referring to the work of both AI theorists and cyberneticians. Another issue in the "language problem," and the point at which Licklider's psychoacoustics research linked with AI, was speech recognition and production. Computers should eventually be able to converse directly with humans in ordinary spoken language. Licklider noted that business

computing, with its relatively slow time scales, might not require the rapid interactive capability of computer speech. But, in a direct reference to military command-and-control issues, he wrote:

> The military commander . . . faces a greater probability of having to make critical decisions in short intervals of time. It is easy to overdramatize the notion of the 10 minute war, but it would be dangerous to count on having more than 10 minutes in which to make a critical decision. As military system ground environments and control centers grow in capability and complexity, therefore, a real requirement for automatic speech production and recognition in computers seems likely to develop.

"Man-Computer Symbiosis" rapidly achieved the kind of status as a unifying reference point in computer science (and especially in AI) that *Plans and the Structure of Behavior*, published in the same year, would attain in psychology. It became the universally cited founding articulation of the movement to establish a time-sharing, interactive computing regime. The following year, for example, when MIT's Long Range Computation Study Group recommended purchasing a very fast, large-memory computer for a time-sharing system, it pointed "the reader interested in a discussion of the idea of a time-shared machine used as a 'thinking center'" to Licklider's paper. The same report noted that AI research *required* a symbiotic method: "We will have to use man-machine interaction for such research; we do not yet know enough to set the machine up completely on its own to solve complex analytic problems."[57]

Licklider's metaphor of "man-computer symbiosis" crystallizes many facets of cyborg discourse in a single phrase. Recalling chapter 1's summary, we can see that the phrase and its history combine

- *techniques* of automation and integration of humans into computerized systems;
- *experiences* of computers as symbol processors and of intimate interaction with computers (and the deep desire for more of this);
- the computer as a *technology* with linguistic, interactional, and heuristic problem-solving capacities;
- particular *practices* of computer use, especially high-level languages, heuristic programming, and the interactive computing developed for SAGE and other military systems;
- *fictions* and *fantasies* about cyborgs, robots, and intelligent machines, like those of the cyberneticians, military futurists, and emer-

gent AI, as well as *ideologies* of AI and interactive computing as military weapons, "thinking aids," and scientific research goals; and

- computer *languages* for formal representation of natural language and logic, such as LISP, whose optimal use requires interactive computing.

The idea of "symbiosis," a biological metaphor applied to human-machine interaction, perfectly captures a moment in the history of this discourse. The computer is not yet, in Licklider's metaphor, a fully realized artificial intelligence or cognitive simulation. It remains a *second* self, in Sherry Turkle's phrase, a partner in thought.[58] "Programming themselves contingently," men will work alongside computers, passing out of the age of "mechanically extended man" into a world of human-machine collaboration, "to think in interaction with a computer in the same way that you think with a colleague whose competence supplements your own."[59]

The Information Processing Techniques Office

Before his appointment to IPTO, Licklider met with then-ARPA director Jack Ruina to discuss computers for command and control. He was accompanied by Fred Frick, George Miller's PAL collaborator from the late-1940s statistical behavioristics studies. (Frick had served as a division head at the Air Force Human Factors Operations Research Group from 1950 to 1954. After that he had worked in various posts at the Air Force Cambridge Research Center, a locus of frequent contacts between MIT, Lincoln Laboratories, and Air Force scientists since World War II.) Licklider recalled their meeting:

> I had this little picture in my mind of how we were going to get people and computers really thinking together. Ruina was thinking of this in terms of command and control, and it didn't take really very much to see how this would work . . . so I had the notion [that] "command and control essentially depends on interactive computing and there isn't any interactive computing so the military really needs this." I was one of the few people who, I think, had this positive feeling toward the military. It wasn't just to fund our stuff, but they really needed it and they were good guys. So I set out to build this program.[60]

Ruina assigned Licklider responsibility not only for IPTO, but also for ARPA's Behavioral Sciences program. Licklider interpreted his behavioral-science mandate to encompass the interactive computing work, reasoning that cognitive psychology, too, would be best served

by investments in advanced computer technology. Thus a range of existing connections both broad and deep carried Licklider into ARPA as an advocate of interactive computing, time-sharing systems, cognitive simulation, and artificial intelligence.

Licklider's career path, like George Miller's, demonstrates how military research problems, and a location within institutions dedicated to their solution, could shape scientists' intellectual interests and visions of the future. As a psychologist, Licklider worked on the human side of the wartime human-machine integration problems of the PAL. At Lincoln he confronted the earliest issues of computer interface design for SAGE. In both laboratories the overarching context was the problem of command and control in electronically mediated, partially automated, eventually computerized systems. At BBN Licklider encountered time-shared computer systems and artificial intelligence. Throughout, as a psychoacoustician and a close colleague of George Miller, he maintained strong interests in language as both a formal system and a human-machine interface. The ideas Licklider enunciated in "Man-Computer Symbiosis" reflected all of these concerns. So Licklider "came to ARPA in 1962 because he interpreted fundamental advances in command and control to be heavily dependent on fundamental advances in computer technology, and [he] was committed to seeking advances in that field, particularly interactive computing."[61]

Histories of ARPA and AI frequently refer to the role of Licklider's "vision" in establishing IPTO as the major backer of AI, time-sharing, and computer networking. This version of the story undoubtedly holds some truth. But our exploration of Licklider's intellectual biography reveals that the "vision" was much more than a personal ideal. It was at least as much a product of the wider discourses of the closed world and the cyborg, technological approaches to politico-military problems and cybernetic metaphors of computers as minds and brains. Licklider's vision emerged from interactions with institutions and other individuals working on specifically military research problems. While it also, of course, encompassed more general concerns, its roots in these interactions played a major role in its construction *as a problem of command and control.*

Furthermore, Licklider did not carry this "vision" to a benighted military. A number of military agencies, including the ONR, the Air Force Office of Scientific Research, Lincoln Laboratories, the Defense Director of Research and Engineering, and Robert McNamara's Of-

fice of the Secretary of Defense itself had more or less definite plans of their own for increasing the use of computers in command and control and "decision support." What Licklider did contribute, through his contacts with AI, was an understanding of time-sharing systems as the specific vehicle of high-technology command-control solutions. Thus AI piggybacked its way into ARPA, riding along with interactive computing.

The goals articulated in "Man-Computer Symbiosis" became, almost without revision, the agenda of ARPA's IPTO under Licklider: time-sharing, interactive computing, and artificial intelligence. Upon his arrival he immediately initiated support for two major time-sharing projects.

The first of these projects took advantage of the recent cancellation of the Systems Development Corporation's Super Combat Center project. Once again rescuing SDC from the loss of military funding, Licklider altered the company's contract, refocusing it on the new goal of a time-sharing system. In December 1962, Licklider challenged SDC to produce a functioning system in six months. SDC met the deadline. Its system, known simply as the Time-Sharing System (TSS), incorporated a General Purpose Display System (GPDS) that included a primitive graphical user interface—the first of its kind. With graphical symbols for input and output functions, the GPDS thus foreshadowed modern icon-based graphical user interfaces such as Windows or the Apple operating system.[62]

The second, far better known project was MIT's Project MAC, begun in 1963. MAC stood variously for Man and Computer, Machine-Aided Cognition, or Multi-Access Computing. With a mandate broader than that of the SDC, Project MAC explored a wide range of interactive computing technologies. MIT had its first time-sharing system up and running by November 1963, and Project MAC ultimately produced a wide range of highly influential time-sharing systems (including CTSS and MULTICS), high-level computer languages (such as MACSYMA), interactive graphic display technologies (such as the KLUDGE computer-aided design system), and programming utilities (compilers and interactive debugging systems). ARPA spent about $25 million (current) on Project MAC between 1963 and 1970.

MAC was part of ARPA's centers-of-excellence strategy, and one of its effects was to establish MIT as the country's premier computer research laboratory. AI research benefited enormously from Project

MAC's support of Marvin Minsky and his many students. The program also had major impacts on commercial computing. IBM lost a competition for MAC hardware contracts and collaborative development to General Electric. This loss "resulted in a considerable reaction by IBM," which instituted a large time-sharing development program of its own as a direct consequence.[63] Military purchases of MAC-based equipment represent yet a third major impact: "The WWMCCS had spent, by 1979, about $700 million [current] on Honeywell 6000-type computers, peripherals and software" developed from the GE equipment originally built for MULTICS.[64]

In 1962, John McCarthy had left MIT to join the growing computer science group at Stanford University. The following year, soon after Project MAC began, he founded the Stanford AI Laboratory, which immediately became another major locus of AI research. Funding from ARPA was virtually automatic; Licklider simply asked McCarthy what he wanted and then gave it to him, a procedure unthinkable for most other government agencies. Licklider remembered that "it seemed obvious to me that he should have a laboratory supported by ARPA. . . . So I wrote him a contract at that time."[65]

Artificial intelligence per se was only one of many kinds of computer research IPTO backed; ARPA budgets did not even include AI as a separate line item until 1968. Numerous other IPTO-funded projects reaped major advances. Perhaps most significant of these was the ARPANET computer network, which eventually spawned the MILNET military network and the modern worldwide network of networks known as the Internet. Other ARPA-supported work has included supercomputing (as in the ILLIAC IV) and advanced microprocessor research (including work on gallium arsenide semiconductors and very-large-scale integrated circuits). IPTO supported advanced computing and AI not only at MIT and Stanford but at Rand, Carnegie-Mellon, SRI, SDC, BBN (Licklider's own company), and seven other "centers of excellence."

As the project with the least immediate utility and the farthest-reaching ambitions, AI came to rely unusually heavily on ARPA funding. As a result, ARPA became the primary patron for the first twenty years of AI research. Former director Robert Sproull proudly concluded that "a whole generation of computer experts got their start from DARPA funding" and that "all the ideas that are going into the fifth-generation [advanced computing] project [of the mid-1980s]—artificial intelli-

gence, parallel computing, speech understanding, natural-languages programming—ultimately started from DARPA-funded research."[66] In the late 1980s, DARPA remained the largest single funding source within the military for computer and behavioral sciences.[67] Since its founding, IPTO has typically provided between 50 and 80 percent of the federal government's share, which is usually by far the largest share, of AI research budgets in the academic centers it funds.[68]

Conclusion: The Closed World and the Cyborg

When Vannevar Bush submitted his report on postwar prospects for government-sponsored scientific research to President Truman in July 1945, he stressed the necessity for national security of a continuing OSRD-style scientific research program in peacetime. Military strength was no longer simply a matter of competent, well-equipped armies. War, he noted, "is increasingly total war, in which the armed services must be supplemented by active participation of every element of the civilian population."[69] "Every element" of that population could be researched, rationalized, and reorganized and its efficiency improved. With computers, in Licklider's vision, a new "population" could "actively participate" in the preparations, thinking alongside human beings as if "with a colleague whose competence supplements your own." Both could be integrated into combat machines through an analysis of two complementary problems in high-technology war.

One was a kind of automation: how to "get man out of the loop" of precision-critical machines, to duplicate and then improve on human prediction and control functions by artificial means. The other was integration: how to incorporate men more smoothly and efficiently into those "loops" where their presence remained necessary—into the chain of command—by analyzing them as mechanisms of the same type and knowable through the same kinds of formalisms as the machines themselves. As we have seen, such a program constituted a central agenda of postwar research. Computers promised *general* solutions to problems of this nature. Thus they made rigorous theoretical analysis on the basis of the erasure of human/machine boundaries both practical and necessary for the first time.

Military sponsors, staffed by veterans of World War II agencies and laboratories concerned with operations research and human-machine integration, could easily perceive, in broad terms, the military utility of research programs aimed at integrating humans and machines.

Such programs took many forms, from directly military engineering (as in SAGE, its "Big L" C^3I offspring, or the McNamara Line of Operation Igloo White) and "human factors" research to experimental studies of cognitive processes and artificial intelligence software to demonstrate the empirical validity of information-processing psychology. Yet all reached toward the same general ends. Thus, however basic and benign it may have seemed to the researchers working on individual projects, the process of producing knowledge of (and for) cyborgs was militarized by the practical goals and concrete conditions of its formation.

Academic psychologists and computer scientists generally did not understand the major role they played in orienting the military toward automation, "symbiosis," and artificial intelligence as practical solutions to military problems. Some caught a glimpse of the overall pattern—such as Wiener, who renounced all military associations in 1946, and Licklider, who deeply desired to contribute to new military technologies from his areas of expertise. But many others simply pursued their own interests, oblivious or indifferent to their place in a larger scheme. They could do so precisely because for the most part there *was* no scheme, in the sense of some deliberate plan or overarching vision. Instead, this larger pattern took the form of a discourse, a heterogeneous, variously linked ensemble of metaphors, practices, institutions, and technologies, elaborated over time according to an internal logic and organized around the Foucaultian support of the electronic digital computer.

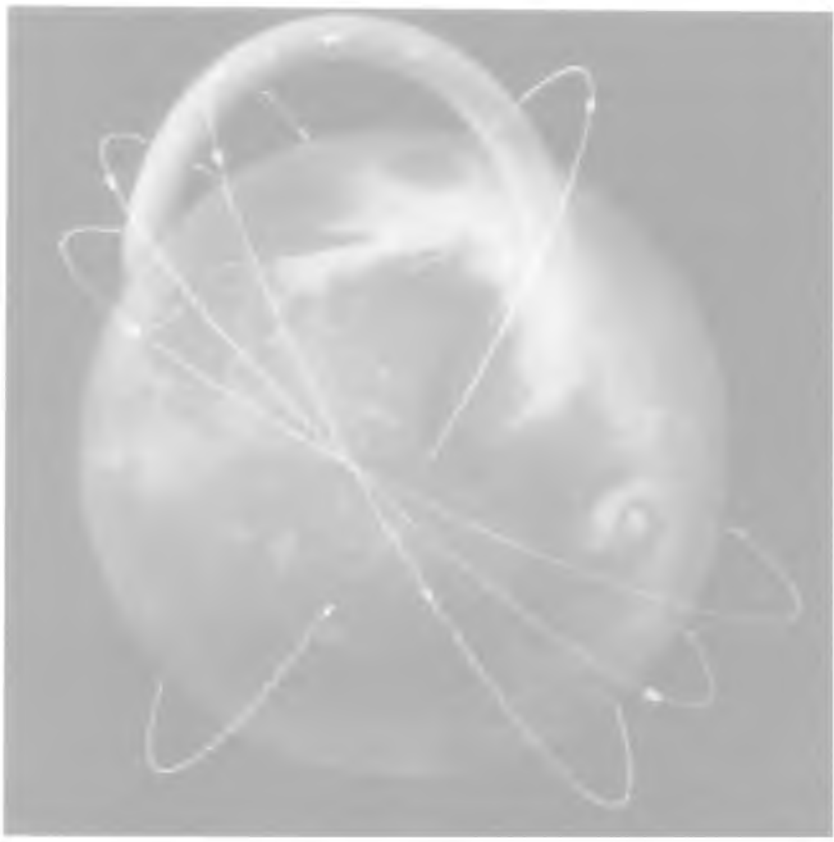
Supported by computers, the military-industrial drive to engineer semiautomatic command-control systems intersected with the postwar political hegemony of the United States in closed-world discourse. The central metaphor of "containment" combined the closures of Cold War ideology and military global reach with computerized systems for total central defense. Likewise supported by computers, the academic-intellectual drive to create artificial intelligence, cognitive psychology, and "man-computer symbiosis" intersected with the psychology of information processing in cyborg discourse. The central metaphor of the computer as a brain or mind combined information theory with concepts of complex mental processes that could ultimately be modeled with computers.

The closed world and the cyborg have in common not only their history and their technological support, the computer, but also their institutional backing by the armed forces, an ideology of formal-

mechanical modeling, a metaphorical system, and a cybernetic language of natural-technical objects. Together, the two discourses constituted an organized, coordinated collection of institutions, technologies, and ideas for integrating human beings and information machines.

Ultimately, closed-world discourse represents the political logic of the cyborg. Seen against its backdrop, military support for cognitive research and artificial intelligence is part of the practical future of military power. The closed world, with its mathematical models, tactical simulations, and electronic battlefields, represents the form of politics and war for brains seen as computers and minds conceived as information processors.

Cyborg discourse is the political subject position, the "psychologic," of the closed world's inhabitants. Artificial intelligence, man-computer symbiosis, and human information processing represent the reductions necessary to integrate humans fully into command and control. The cybernetic organism, with its General Problem Solvers, expert systems, and interactive real-time computing, is the form of a mind in symbiotic relationship with the information machines of the electronic battlefield.



An artist's conception of the Strategic Defense Initiative space-based ballistic missile defense, circa 1985. Courtesy Lawrence Livermore National Laboratory.

70. George Miller and Jerome Bruner, "Center for Cognitive Studies: General Description," April 8, 1960. Harvard University Archives.

71. Ibid.

72. Ibid.

73. Ibid.

74. Miller, in an interview with the *Baltimore Sun* (October 23, 1982). Neisser, who consolidated the field with his *Cognitive Psychology* (New York: Appleton-Century-Crofts, 1967), which explicitly touted the computer metaphor, repudiated it almost bitterly nine years later in *Cognition and Reality* (San Francisco: Freeman, 1976).

Chapter 8

1. The quasi-official versions of AI history, based primarily on interviews with the founders, are Daniel Crevier, *AI: The Tumultuous History of the Search for Artificial Intelligence* (New York: Basic Books, 1993), and Pamela McCorduck, *Machines Who Think* (New York: W. H. Freeman, 1979). Other semipopular, interview-based works include Howard Rheingold, *Tools for Thought* (New York: Simon & Schuster, 1985), and Frank Rose, *Into the Heart of the Mind* (New York: Harper & Row, 1984). The founders' own accounts include Allen Newell, "Intellectual Issues in the History of Artificial Intelligence," in Fritz Machlup and Una Mansfield, eds., *The Study of Information: Interdisciplinary Messages* (New York: Wiley, 1983), 187–228; the Appendix to Allen Newell and Herbert Simon, *Human Problem Solving* (Englewood Cliffs, NJ: Prentice-Hall, 1972); and Herbert A. Simon, *Models of My Life* (New York: Basic Books, 1991). A serious intellectual history of cognitive science, including AI, is Howard Gardner, *The Mind's New Science* (New York: Basic Books, 1985).

2. See John von Neumann, *The Computer and the Brain* (New Haven: Yale University Press, 1958), posthumously published.

3. One of the most significant offshoots of the cybernetic approach was early work on neural network modeling. On this approach and why it languished for two decades after the early 1960s, despite important early successes, see Mikel Olazaran Rodriguez, *A Historical Sociology of Neural Network Research* (unpublished Ph.D. thesis, Department of Sociology, University of Edinburgh, 1991). The best-known of the 1950s neural-model schemes was the "perceptron." See Frank Rosenblatt, *Principles of Neurodynamics* (New York: Spartan, 1962), and Marvin Minsky and Seymour Papert, *Perceptrons* (Cambridge, MA: MIT Press, 1969).

4. Interestingly, while Turing is usually remembered for the extremely wide range of problems his hypothetical machine can solve, the mathematically significant point of his paper consists in a proof that there exist some problems that *cannot*, even in principle, be solved by a Turing machine.

5. See Michael S. Mahoney, "The History of Computing in the History of Technology," *Annals of the History of Computing*, Vol. 10, No. 2 (1988), 113–125, and William F. Aspray, "The Scientific Conceptualization of Information: A Survey," *Annals of the History of Computing*, Vol. 7, No. 2 (1985), 117–140.

6. Andrew Hodges, *Alan Turing: The Enigma* (New York: Simon & Schuster, 1983), 106.

7. Stan Augarten, *Bit by Bit* (New York: Ticknor & Fields, 1984), 145. Similarly, of the early computer pioneers only Howard Aiken had studied the work of Charles Babbage, the nineteenth-century inventor of a mechanical protocomputer.

8. As Winograd and Flores point out, this notion of opacity is part of the ideology of cognitivism and may in fact be violated in situations of breakdown. See Terry Winograd and Fernando Flores, *Understanding Computers and Cognition: A New Foundation for Design* (Norwood, NJ: Ablex, 1987).

9. Example borrowed from Augarten, *Bit by Bit*, 213.

10. Cf. Steven Levy, *Hackers* (New York: Anchor Press, 1984), Part I.

11. Morse, cited in Karl L. Wildes and Nilo A. Lindgren, *A Century of Electrical Engineering and Computer Science at MIT, 1882–1982* (Cambridge, MA: MIT Press, 1985), 335.

12. Mahoney, 120, 117. On programming as a craft skill and its work organization, see Philip Kraft, *Programmers and Managers* (New York: Springer-Verlag, 1977).

13. See Henry S. Tropp et al., "A Perspective on SAGE: Discussion," *Annals of the History of Computing*, Vol. 5, No. 4 (1983), 386.

14. On the history of programming languages see Jean Sammet, *Programming Languages: History and Fundamentals* (Englewood Cliffs, NJ: Prentice-Hall, 1969).

15. Jerry A. Fodor, *The Language of Thought* (New York: Thomas Y. Crowell, 1975).

16. Oliver G. Selfridge, "Pattern Recognition and Modern Computers," *IRE Proceedings of the 1955 Western Joint Computer Conference* (1955); Oliver G. Selfridge, "Pandemonium, a Paradigm for Learning," in D. V. Blake and A. M. Utley, eds., *Proceedings of the Symposium on Mechanisation of Thought Processes* (London: H. M. Stationery Office, 1959).

17. McCorduck, *Machines Who Think*, 134.

18. Ibid., 116.

19. A. Newell, J. C. Shaw, and H. A. Simon, "Elements of a Theory of

Human Problem Solving," *Psychological Review*, Vol. 65, No. 3 (1958), 151–166, italics added.

20. Allen Newell, "Physical Symbol Systems," *Cognitive Science*, Vol. 4 (1980), 135–183.

21. For a more extended discussion of differences between AI and cybernetics, see Newell, "Intellectual Issues."

22. McCorduck, *Machines Who Think*, 67.

23. Claude Shannon and John McCarthy, *Automata Studies* (Princeton, NJ: Princeton University Press, 1956).

24. My account of the conference follows McCorduck, *Machines Who Think*, chapter 5, and Gardner, *The Mind's New Science*, chapter 6.

25. From the grant proposal for the Dartmouth conference, cited in McCorduck, *Machines Who Think*, 99.

26. Donald O. Hebb, *Organization of Behavior* (New York: Wiley, 1949).

Marvin Minsky, *Neural Nets and the Brain-Model Problem* (Ph.D. thesis, Princeton University, 1954).

28. McCorduck, *Machines Who Think*, 86.

29. On the LISP language, see John Haugeland, *Artificial Intelligence: The Very Idea* (Cambridge, MA: MIT Press, 1985), 147–157.

30. Cited in Hubert Dreyfus, *What Computers Can't Do*, 2nd ed. (New York: Harper Colophon, 1979), 81–82.

31. McCorduck, *Machines Who Think*, 42.

32. Explicit references to the Enlightenment project for an encyclopedia of all human knowledge are common in the AI literature. See Edward Feigenbaum and Pamela McCorduck, *The Fifth Generation: Japan's Computer Challenge to the World* (Reading, MA: Addison-Wesley, 1983), and Douglas Lenat et al., "Cyc: Toward Programs with Common Sense," *Communications of the ACM*, Vol. 33, No. 8 (1990), 31–49.

33. Jay W. Forrester, "Whirlwind and High-Speed Computing" (1948), reprinted in Kent C. Redmond and Thomas M. Smith, *Project Whirlwind* (Boston: Digital Press, 1980), 225–236.

34. D. N. Arden et al., *Report of the Long Range Computation Study Group* (Massachusetts Institute of Technology, 1961), 20. MIT Archives.

35. Ibid., 25, 31.

36. McCarthy, interviewed in McCorduck, *Machines Who Think*, 217.

37. Marvin Denicoff, "AI Development and the Office of Naval Research," in Thomas C. Bartee, ed., *Expert Systems and Artificial Intelligence* (Indianapolis: Howard W. Sams & Co., 1988), 272.

38. Ibid.

39. Sidney G. Reed, Richard H. Van Atta, and Seymour J. Deitchman, *DARPA Technical Accomplishments,* Vol. 1 (Alexandria, VA: Institute for Defense Analyses, 1990), 1.

40. Richard J. Barber Associates Inc., *The Advanced Research Projects Agency 1958–1974* (Washington, DC: National Technical Information Service, 1975), I-26.

41. Arthur L. Norberg and Judy E. O'Neill, with Kerry J. Freedman, *A History of the Information Processing Techniques Office of the Defense Advanced Research Projects Agency.* The Charles Babbage Institute, University of Minnesota (October, 1992), 33.

42. See Terry Winograd, "Strategic Computing Research and the Universities," in Charles Dunlop and Rob Kling, eds., *Computerization and Controversy* (New York: Academic Press, 1991), 704–716.

43. "Message from the President of the United States Relative to Recommendations Relating to Our Defense Budget," 87th Congress, 1st Session, 28 March 1961, House Documents, Document No. 123, 8, cited in Norberg and O'Neill, *A History of the Information Processing Techniques Office*, 35.

44. Robert S. Englemore, "AI Development: DARPA and ONR Viewpoints," in Bartee, *Expert Systems and Artificial Intelligence*, 215.

45. Barber Associates, *The Advanced Research Projects Agency 1958–1974*, V-49.

46. Richard H. Van Atta, Sidney G. Reed, and Seymour J. Deitchman, *DARPA Technical Accomplishments,* Vol. 2 (Alexandria, VA: Institute for Defense Analyses, 1991), 13–14.

47. J. C. R. Licklider, interviewed in John A. N. Lee and Robert Rosin, "The Project MAC Interviews," *IEEE Annals of the History of Computing*, Vol. 14, No. 2 (1992), 15.

48. Wildes and Lindgren, *A Century of Electrical Engineering and Computer Science at MIT*, 247.

49. Jerome Wiesner, cited in ibid., 265.

50. Ibid., 257.

51. Lee and Rosin, "The Project MAC Interviews," 17.

52. Norberg and O'Neill, *A History of the Information Processing Techniques Office*, 37–38.

53. J. C. R. Licklider, quoted in Rheingold, *Tools for Thought*, 140.

54. J. C. R. Licklider, quoted in Norberg and O'Neill, *A History of the Information Processing Techniques Office*, 39.

55. This and all subsequent quotations in this section are from J. C. R. Licklider, "Man-Computer Symbiosis," *IRE Transactions on Human Factors in Electronics*, Vol. HFE-1 (1960), 4–11.

56. Italics added.

57. Arden et al., *Report of the Long Range Computation Study Group*, 43.

58. See Sherry Turkle, *The Second Self* (New York: Simon & Schuster, 1984).

59. Licklider, "Man-Computer Symbiosis," 6.

60. Lee and Rosin, "The Project MAC Interviews," 18.

61. Barber Associates, *The Advanced Research Projects Agency 1958–1974*, V-50.

62. Van Atta, Reed, and Deitchman, *DARPA Technical Accomplishments*, Vol. 2, 13–14.

63. Ibid.

64. Honeywell had purchased the GE computer operation (ibid., 19–14).

65. J. C. R. Licklider, "The Early Years: Founding IPTO," in Bartee, *Expert Systems and Artificial Intelligence*, 220.

66. Robert Sproull, quoted in *IEEE Spectrum*, Vol. 19, No. 8 (1982), 72–73.

67. Kenneth Flamm, *Targeting the Computer* (Washington, DC: Brookings Institution, 1987), 53.

68. In industry, DARPA's funding share has been considerably lower. But until the 1980s, when expert systems began to find commercial applications, industry research in AI took a back seat to academic work.

69. Vannevar Bush, *Science: The Endless Frontier* (Washington, DC: U.S. Government Printing Office, 1945), 12.

Chapter 9

1. Fred Halliday, *The Making of the Second Cold War* (London: Verso, 1986). I will use a modified form of Halliday's periodization, marking the first Cold War as the period between 1947 (containment doctrine) and 1969 (the Nixon administration). Cold War II ran from 1979 (Carter's rightward shift) to 1989 (the collapse of East Germany). When referring to the Cold War *tout court*, I intend the entire period 1947–1989.