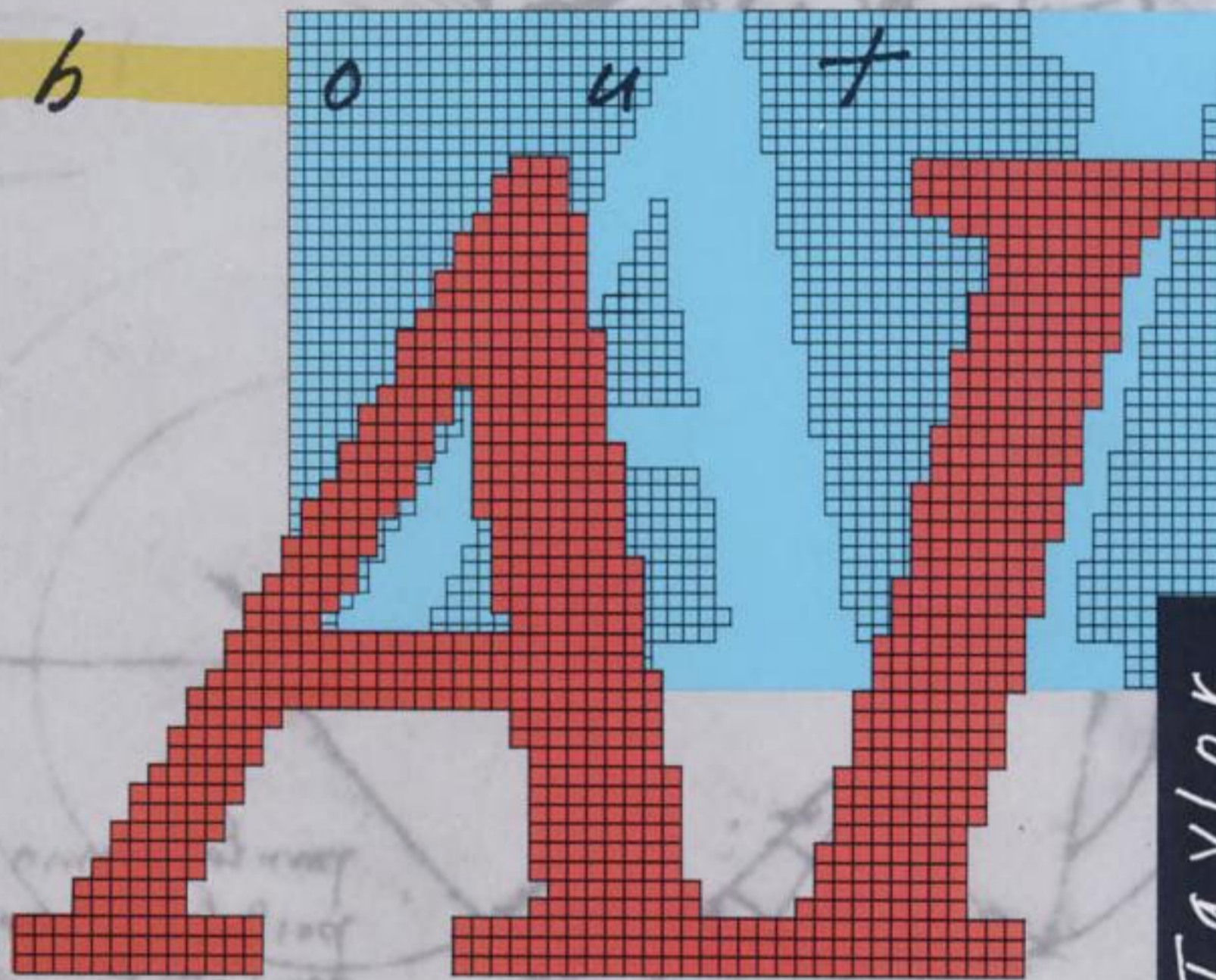


What Every

Engineer

Should Know

Abot



William A. Taylor

What Every Engineer Should Know About Artificial Intelligence

William A. Taylor

**The MIT Press
Cambridge, Massachusetts
London, England**

Copyright © 1988 Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form or by any electronic or mechanical means, (including photocopying, recording, or information storage and retrieval), without permission in writing from the publisher.

This book was printed and bound in the United States of America.

PUBLISHER'S NOTE

This format is intended to reduce the cost of publishing certain works in book form and to shorten the gap between editorial preparation and final publication. Detailed editing and composition have been avoided by photographing the pages of this book directly from the author's prepared copy.

Library of Congress Cataloging-in-Publication Data

Taylor, William A.

What every engineer should know about artificial intelligence/William A. Taylor

p. cm.

Bibliography: p.

Includes index.

ISBN 0-262-20069-4

1. Artificial intelligence. 2. Artificial intelligence--Data processing. I. Title.

Q335.T39 1988

006.3--dc19 87-35343

Contents

	Acknowledgments	xi
1	What Is Artificial Intelligence?	1
	Engineering Opportunity in AI	2
	The Promise of Artificial Intelligence	3
	Derivative Results of Artificial Intelligence	7
	A Definition of Artificial Intelligence	9
	Upward and Onward	9
2	Humans and Computers, Silicon Life and Carbon Life	11
	Theory and Practice	12
	Status Report on AI Research	13
	How Computers Work	14
	Handling Life's Little Problems	22
	Who Needs Intelligent Computers?	23
3	Results of AI Research	25
	Time Sharing	25
	Man-Machine Interfaces	26
	Natural Language Processing	27
	Expert Systems	29
	Logic Programming	30
	Object-Oriented Programming	30
	Spreadsheets	31
	AI Technologies That Do Not Work Well Yet	32
	Technologies That Will Not Work Soon	38
	The State of the Art	40
4	Applications of Artificial Intelligence	41
	Problems with Management	42
	Applications	44
	Lessons to Be Learned	65
5	The Lisp Programming Language	67
	Data, Rules, and Goals	67
	Commercial Success and Failure	69
	The Software Development Process	71
	Why Lisp Was Invented	73
	Review of Lisp	87
6	Using Lisp	89
	Using Properties: "Is" and "Has"	89
	Inferring New Facts from Old Facts	90
	Semantic Nets	93

Object-Oriented Programming	97
Memory Management in Lisp	99
List Structure	104
Summary of Lisp	108
Other Languages for AI	109
7 Software Tools for AI	111
Conventional Engineering Tools	112
The Software Development Process	113
Writing Programs Rapidly	117
Was It Worth the Cost?	126
What Next?	129
8 Programming Styles	131
Function-Based Programming	132
Object-Oriented Programming	133
Rule-Based Programming	134
Summary of Programming Styles	138
Conventional Device Independence	139
The Bottom Line on Programming	140
Goal-Oriented Factory Automation Software	141
Now What?	142
9 Expert Systems	143
Reasons for Developing Expert Systems	144
Rules in Expert Systems	144
Kinds of Expert Systems	145
Choosing Problems for Expert Systems	146
Nontechnical Considerations	149
Operation of Expert Systems	153
Mycin: A Medical Expert System	156
R1: A Forward-Chaining Expert System	160
Commercial Status of Expert Systems	163
Expert Systems and Logic Programming	164
10 Logic Programming	165
Pattern Matching	166
Rules Tell How to Infer New Facts	171
Unification	179
Summing up Mathematical Logic	182
11 User Interfaces to Logic Systems	185
The Importance of Asking Questions	185
A Simple Expert System	186
Problems with the User Interface	193
What Next with Rules?	193
12 Issues in Rule and Knowledge Management	195

Expert System Shells	195
The Role of Knowledge Management	204
Parts of Expert System Shells	206
Winning the Shell Game	213
13 Prolog	215
Prolog versus Lisp	216
Making Computers Compute	218
Prolog versus Logic Programming	218
Prolog Syntax	220
Control Structure	229
Who Needs Logic Programming?	244
The Japanese and the AI Market	245
14 Getting Started in AI	247
Hiring Outside Help	247
Training In-House Staff	248
Writing Expert Systems	249
Learning Lisp	251
Learning Object-Oriented Programming	254
Slogans for the AI Revolution	254
AI Is Only Another Technology	259
The Time for AI Is Now	260
15 The Future of Artificial Intelligence	263
Copying Human Intelligence	263
Commercialism	265
Advice Programs	265
New Kinds of Service	268
New Products	271
Overall Effects	273
Technology Trends	273
16 Japan and the Fifth Generation Project	275
Starting the Project	276
Why Do the Japanese Do It?	280
How Do the Japanese Do It?	282
What the Japanese Say about Japan	288
Japanese Research Projects	290
Impact of Artificial Intelligence	298
What to Do about Japanese Competition	298
The Real Battle	305
Markets and Technology	306
Bibliography	307
Artificial Intelligence Directories	308
A Dose of Realism	308

Philosophy of Innovation	309
Philosophy of Artificial Intelligence	310
Academic Artificial Intelligence	311
Commercial Artificial Intelligence	313
Getting Started in Lisp	313
Expert Systems	315
Prolog	317
Object-Oriented Programming	318
Japan	319
Other References	320
Index	323

Acknowledgments

I had a great deal of help from many people. Dick Morley provided a suitable environment. Bob Morley drew the pictures and demanded clarity. Art Habel passed a superbly experienced editorial eye over the text. Other people offered suggestions and, best of all, frank criticism.

Debugging text is harder than debugging programs—a computer is willing to read and criticize a program after each change as if it had never seen it before, whereas finding human text critics is difficult. Steve Rowley is a premier AI practitioner who was kind enough to read the manuscript and mark it up. His critical pen improved the book immeasurably. Thanks, Steve.

Harold Abelson of MIT made helpful comments.

I also received help from AI vendors. Gold Hill, Symbolics, and Texas Instruments all gave me experience with different Lisp systems at their facilities and told me about their customers. Eloquent Systems provided AI combat stories, and Flavors Technology let me use their computers and typesetting equipment.

Worcester Polytechnic and Frost & Sullivan forced me to broaden my experience by signing me up to lecture about AI, and Gould sponsored my first effort at writing an expert system.

What Every Engineer Should Know About Artificial Intelligence

1

What Is Artificial Intelligence?

Artificial intelligence (AI) will have profound effects on engineering practice, but it is sometimes hard to see why. The popular definition of artificial intelligence research means designing computers that think as people do, and who needs that? There is no commercial reason to duplicate human thought because there is no market for electronic people, although it might be nice if everyone could have a maid and butler. There are plenty of organic people, and computer vendors can't compete with the modern low-cost technology used in making people.

Computers that think like people are far enough in the future that we can safely ignore them. From that point of view, AI research has failed because there are no computer programs anywhere that imitate human thought.

To be fair, AI is still very young. Atoms were thought to be indivisible for millennia after Aristotle invented atomic physics, and AI has moved faster than that. Research into how the human mind works forced the development of many new software tools and techniques that can greatly benefit the engineering profession.

For engineering purposes artificial intelligence is a few software ideas that work well enough for commercial use. These

accidental offshoots of AI research are slowly but surely changing the ground rules of engineering practice. In this book I explain AI so that engineers can understand what is happening and prepare for it.

Engineering Opportunity in AI

People buy computers because they are useful, not just for the thrill of owning an electronic gadget. Engineers should care deeply about artificial intelligence because it makes computers more useful. A computer without software is like a newborn infant. Human hardware is available at the moment of birth, but it cannot be used without knowledge and experience.¹ Humans become more useful as they mature and learn; computers become more useful as new programs are written. AI tools and techniques help write new and useful programs.

Limitations of Computers in Engineering

Computer-aided design (CAD) systems are really electronic drafting boards. They bear the same relationship to T-squares, pencils, and drafting tables that word processors bear to typewriters. A CAD system erases and redraws lines as needed to describe a design. The computer serves as a high-cost replacement for a sketch pad, a pencil, and an eraser. Especially the eraser—computers are superb at erasing.

CAD systems are misnamed because they do not aid the *design* process at all. Design requires understanding a problem, imagining a solution, trying it, then changing the solution until it works. Computers reduce the effort required for documentation. This helps designers to focus their thoughts and convey the design to others who have to implement it, but generating documentation is not design.²

The associations between seemingly unrelated facts that constitute the core of creative design happen between the

¹ The brain grows larger as a child grows, but most of the growth seems to occur in the synapses, which wire neurons together. That would be like shipping a personal computer with all the ICs piled in the bottom of the case and adding circuit board traces to establish connections over time. This idea is explained in *The Amazing Brain*, by R. Ornstein and R. Thompson (Boston, Mass.: Houghton Mifflin, 1984), p. 69. However, Ornstein and Thompson later assert that there are more connections in infant brains than in adult brains and that development consists of pruning unneeded connections rather than growing new ones (p. 166).

² There are a few software packages which help with the design process. The ICAD package is discussed in chapter 4. Symbolics, Inc. uses the NS software package to design custom integrated circuits.

engineers' ears, not on paper or on a computer terminal. Except for some aid in redrafting, computers are of little help in design. Computers cannot suggest changes because they understand neither the problem nor the solution. Without understanding of goals and means, there can be no design.

This is not to say that computers have no place in engineering. Engineers abandoned slide rules for pocket calculators and personal computers, and mainframe computers carry out tedious calculations to verify designs. The difficulty is that the engineer supplies all of the design knowledge. The computer grinds out numbers but has no idea what they mean. Only humans possess the insight to attach meaning to the numbers, so only humans can tell whether the numbers are correct.

This is true even when the computer seems to be helping with the design. Computer-aided engineering (CAE) workstations simulate electronic circuits to find errors. Electronic simulation programs are good enough that many products go directly from schematic to printed circuit boards, but calculating numbers according to a formula is not engineering. Circuit simulation serves essentially the same design verification function as stress analysis—the computer cranks out numbers showing voltage at any point in the circuit, but the engineer decides if the voltages are correct. The computer's task is purely clerical, albeit quite useful.

The Promise of Artificial Intelligence

AI promises that computers will be able to provide design help instead of being limited to clerical and numerical tasks. Researchers are beginning to put design rules into computers to help with the design process. Computers are beginning to ask, "Is that wire too long?" "Why not drill the hole bigger?" or "Are you sure the building won't sway during a hurricane?"

Computers that understand design rules will serve as electronic apprentices by taking over the simpler design tasks. Just as Michelangelo had his students paint backgrounds while he concentrated on major figures, computers will make suggestions and handle the simpler parts of the design.

Today's computers make many silly suggestions just like human novices, but as computers gain experience, computer programs will take over more and more of the mundane parts of engineering. Computers already check circuit design rules and verify manufacturability of printed circuit boards. They will soon calculate part routing in machine shops and devise assembly procedures.

Computers that know when to calculate stresses and that can make simple design changes will come with time. Once computers take over some of the clerical parts of engineering, humans will have more time for creative work. We will be able to spend more time proposing new solutions and less time checking old ones.

Practical Impact of Artificial Intelligence

The practical fruits of AI research help to make computers more useful and to make it possible to apply computers to new fields. Opportunities to use computers in engineering are multiplying. Just as computers called "engineering workstations" altered the draftsman's job, using computers to help with design will change the engineering profession.¹

Whenever computers are applied to a new problem, great changes occur. Anyone involved in computerizing accounting and financial management remembers that it was an unholy mess. The balance sheet did not balance. Paychecks were not distributed on time. Invoices were not sent out. After much hair pulling, computers were taught to keep accounts and things settled down. *But nobody saved any money!* Accounting departments replaced armies of low-paid ledger clerks with platoons of high-paid programmers dwelling in air-conditioned splendor.² After accounting software matured, computerized accounting became cheaper than manual bookkeeping, but getting there took time.

Using computers in the design process will cause anguish and pain at first, but once the dues are paid, engineers will emerge from the fray with a whole new bag of tricks. The profession will improve beyond recognition.

The Intellectual Revolution

Artificial intelligence techniques are bringing about a revolution in the way people handle information. Engineers are essentially information processors. We wrinkle paper and make telephone lines hot; we do not shovel sand. We are paid for telling others

¹ Architects still argue about when it is best to use a pencil and when to use a computer. There are engine plants in Detroit where engineers redraw any computer-generated prints they receive because computers do not follow their notation conventions and the shop floor people prefer familiar drawings.

² *Fortune* magazine states that employment in the accounting departments of the 1,000 largest firms stayed constant in terms of bodies per dollar of annual sales throughout the computer revolution.

Computers long ago replaced humans for clerical tasks such as accounting. By the mid-1970s, computers could have replaced humans at some engineering tasks but were too expensive to make it worthwhile. Now that costs have dropped, entrepreneurs see new opportunities and are developing new engineering support tools based on AI.

Technologies for Human Emulation

Emulating human behavior requires many different technologies. Being human is a complex task,¹ so researchers divided their efforts into specialties: vision, speech recognition, natural language processing, locomotion, expert systems, planning, and automatic learning.

Vision is required for humanoid robots and would be helpful in factories because visual gauges do not touch the workpiece and do not wear. Vision systems have begun to creep into factories but are still very difficult for factory people to use.

Speech recognition is hearing a stream of human speech and identifying all the words. People do not really hear all the words during conversation. We deduce many words by understanding their context.

Natural language processing (NLP) means analyzing a string of words to decide what they mean. Conversation requires speech recognition to identify the words being said, NLP to understand what they mean, and speech synthesis to formulate replies. Speech synthesis is too simple to be a core part of AI.

Robotics researchers are trying to design machines that walk around like C3PO in *Star Wars*. Studying locomotion is not strictly part of the research into how the human mind works; it grew out of efforts to understand the parts of the brain that control motion. Two-legged walking has turned out to be extremely difficult to duplicate.

Expert systems encode knowledge in if/then rules. Computer programs embody human knowledge. All programs are the result of a human telling a computer how to do a task. Rules provide a new way of coding knowledge that saves time and money in some situations.

Planning is necessary for intelligent behavior. Building a successful robot requires computerized planning because the

¹ The more I study artificial intelligence, the more I appreciate the genuine article. Psalm 139:14 sums it up — "I will praise thee; for I am fearfully and wonderfully made: marvellous are thy works, and that my soul knoweth right well." It has been said that artificial intelligence is better than none. This is true, but only barely.

big as mouse ears," or "Wine is a mocker, and strong drink a raging, and he who is deceived by them is not wise," or "Anything that can go wrong, will go wrong," or even " $F = ma$."

Such concepts are almost impossible to handle with conventional programming languages. Before AI research could really get underway, researchers needed a language that could manipulate symbols and the associations between them.

"Lisp" stands for "List processing," and is the name of a programming language invented in the early 1960s by John McCarthy at MIT. McCarthy designed Lisp to keep track of relationships between different kinds of information. Information is represented by symbols that stand for ideas, just as the symbols "dog" and "cat" stand for the idea of friendly furry domestic animals. Programming in terms of relationships between symbols is called "symbolic programming" and has been the key to efforts to understand human thought.

Purely by accident Lisp turned out to be commercially useful as well as academically interesting.

Software Development Tools

AI researchers developed special computer programs to help write other programs. Software developers produce far more function per unit of effort in the AI programming environment than in any other environment.

Powerful tools were desperately needed because early AI programs were the largest that had been written at the time. Researchers quickly saturated the biggest computers MIT could buy. Some computers had as many as 256,000 words, but programs outgrew memories that small by the mid-1960s.¹ Keeping track of all the parts of such large programs is a clerical task for which computers are well suited.

Commercial firms that adopt the software development tools and procedures pioneered in AI research centers eventually find that their software costs drop. When products such as microwave ovens, toasters, and washing machines are controlled by computers, software becomes important to more and more products. Firms that write software in less time or with fewer people than their competitors will have a significant advantage in the era of electronic appliances and software-based products.

The cost of the software in a microwave oven is not particularly high, especially when spread over thousands of units.

¹ Patrick Winston, director of the MIT AI Lab, said at a press conference at AAAI '87 that his portable personal computer has enough memory to have cost \$3 million in 1967, when he began his AI work.

The advantage of having better software tools is that new software can be developed more quickly. The faster a company can develop new software and produce newer models, the easier it is to keep ahead.

One factor that contributed to the Japanese victory in the American electronics market was their ability to produce new models quickly. Sony introduces new models of the Walkman cassette player several times per year. Developing products this rapidly is not possible without superb engineering tools.

A Definition of Artificial Intelligence

Donald Knuth once said, "The difference between art and science is that science is what we can program into a computer. All else is art." Artificial intelligence covers so much intellectual ground that it can be difficult to define it much more precisely than that.

To me, AI is two unrelated things—human emulation that does not work and a few software techniques that are ready for use. I define AI as a programming style, where programs operate on data according to rules in order to accomplish goals.¹ To a chess-playing program, the data are the positions of the men on the board. Rules are the moves permitted in the game of chess and other rules that define strategy. The goal is to win the game.

This programming style corresponds to engineering practice. Data are supplied in the statement of the problem. Rules are the properties of materials, engineering lore, design experience, contents of technical manuals, the accumulated wisdom of the profession. The goal is to solve a problem—build a dam, bridge a river, make a circuit do something a customer wants badly enough to allocate resources to acquire it.

Upward and Onward

As AI technology matures, it will become easier and easier to encode rules of engineering practice in computers. Watching computers take over more and more engineering tasks will be absolutely fascinating to those who stay on top of the new techniques and extremely threatening to those who fall behind.

AI has moved out of the academic and intellectual stage and is showing signs of commercial worth. Welcome to the world of the artificial intelligentsia.

¹ This is a narrow definition of AI because it excludes everything AI researchers are doing except building bigger and better expert systems. This part of AI seems likely to have commercial potential sooner than other areas.

?

2 Humans and Computers, Silicon Life and Carbon Life

Computers are not "thinking machines" or "giant brains" or any of the other terms commonly used to describe them in the early days. In this chapter I discuss some of the reasons why it is so difficult to program computers to exhibit intelligent behavior.

Life based on carbon compounds is made in both intelligent and unintelligent varieties, but the entire product line is built out of cells. Cells are made of a membrane wrapped around a bit of organic goo wrapped around a nucleus. The membrane collects raw materials from the environment and passes waste products out of the cell, the goo manufactures chemicals the cell needs to survive, and the nucleus contains the information needed to make more cells.

Silicon life is not manufactured in as many models as carbon life, and none of them is intelligent. Silicon life is based on transistors, which were invented in 1948. All that a transistor in a computer can do is switch current on and off. Transistors have no metabolism, cannot manufacture chemicals, and cannot reproduce themselves.

On a purely functional basis, cells seem to be much better components with which to build intelligence, but transistors have a major advantage. A transistor processes information

much faster than a cell. If we knew how to connect up transistors to emulate human thought, silicon life would think *fast*.

Theory and Practice

In theory, the thinking performance of silicon life ought to be comparable to carbon life. A human brain has about 10^{10} neurons, and they seem to be able to change state by switching from on to off at peak rates of about 1,000 times per second. If all the neurons in a brain switch at once, the maximum compute power is about 10^{13} state changes per second.

The fastest supercomputer made has about 10^9 transistors, but transistors can change state 10^6 times as fast as neurons—about 10^9 times per second. If all the transistors are switching at once, the supercomputer has a compute capacity of 10^{18} transitions per second, or 5 orders of magnitude faster than a human brain.¹

It should be possible to build computers that think faster than human minds, but computers cannot be made to think at all. Part of the reason is that only a small fraction of 1% of the transistors in a computer are switching at a time. This is because most computers are organized to do only one thing at a time, unlike the human brain, which does many things at once.

New computers incorporating many processors are being developed to do many different things at once. Although it is too early to tell, there is hope that these research efforts will lead to computers that are more intelligent and more useful than the computers of today.²

¹ Saying that a neuron switches 1,000 times per second may overestimate the switching capacity of the brain. Only the fastest neurons in the eye switch at 1,000 times per second, and most are much slower. The real rate does not matter much for the purposes of this comparison.

Neurons may transmit information using frequency modulation rather than amplitude modulation. That is, the rate at which a neuron switches is the signal, not the individual pulses themselves. If this is so, 1,000 transitions per second is the carrier frequency and not the data rate, and neurons generate information far slower than 1,000 transitions per second. The overall human neural system seems to be able to make a simple decision, such as jumping away from danger, in about 0.2 second, or five urgent decisions per second.

No one is sure what a neuron really does. Bernard Woodrow of Stanford University proposed a neuron model in 1964 in which a neuron could learn to play blackjack. If playing blackjack really takes only one neuron, it is not surprising that humans can work marvels given that we have so many.

² Putting many small computers in the same box and having them work on the same problem has been a research area for more than twenty years. It has always been much cheaper to buy many small computers than to buy one big one. A microprocessor costing about \$10,000 executes 4 or 5 million instructions per

Status Report on AI Research

Setting the goal of emulating human thought was all very well, but researchers needed to know when the goal had been achieved. "Intelligence" is difficult to define rigorously, so AI researchers accepted a standard for computer intelligence known as the *Turing test*.

The Turing test is simple. An untutored human interrogator is alone in a room with two computer terminals. One terminal is connected to a computer, and the other is linked to another terminal with a human operator. The interrogator may type *anything* on either terminal—questions, statements, discussions of any topic. The operator wins if the interrogator correctly identifies the human. If the interrogator cannot differentiate between the human and the computer, the computer passes the test and is intelligent *by definition*.¹

There are some nonobvious difficulties in passing the test because the computer must introduce typing errors and type slowly in order to seem human. The computer must emulate both human intelligence and human fallibility in order to be mistaken for a human. The Turing test does not measure pure intelligence any more than human IQ tests do but demands a mixture of abilities, including imitating human typing patterns. This mixture of skills is defined as intelligence for the purposes of the test.

Like IQ tests, the Turing test has the virtue of not requiring a definition of intelligence. Humans behave intelligently, so any computer that mimics human behavior is intelligent by definition. This reminds me of the method once used to weigh hogs in the Appalachian mountains. Nobody had scales. Farmers put the hog on one end of a balance pole and piled rocks

second. The world's fastest supercomputers run only 100 times faster but cost 2,000 times as much. Economies of scale do not apply to large computers.

One solution is to put many microprocessors in a box and run them in parallel. Nearly 30 vendors offer parallel computers but they have not made significant commercial impact yet. Parallel processing is subject to difficult software problems and requires new computer architectures. One of the most unusual parallel computers is described in *The Connection Machine*, by W. Daniel Hillis (Cambridge, Mass.: MIT Press, 1985).

¹ There is another definition of the Turing test. The human is a man pretending to be a woman; the computer pretends to be a woman. The computer is intelligent if the computer is chosen as the "woman." The details of the definition do not matter because no computer could imitate either a man or a woman today.

The processor is often abbreviated "CPU," which stands for central processing unit. The CPU is responsible for all changes that happen to information while it is in memory. The CPU also decides when to write old data out of memory and when to read new information in.

Data are transferred between memory and devices such as disks, magnetic tapes, terminals, printers, card readers and other peripheral devices which are collectively called "I/O." Whenever the CPU decides to move data, it sends commands to an I/O device, telling it which data to move and where to put it.

The CPU and memory in a computer are usually put close together because information must pass between them quickly. Having the CPU too far from the memory would slow the computer down. I/O devices can be located further away because data move more slowly between memory and I/O devices. Terminals hundreds of miles from the computer can transfer data over telephone lines, but telephone lines cannot transfer data as quickly as the connections inside the computer.

Humans Operate Differently from Computers

There is an attractive similarity between computers and humans. It is almost impossible to resist the temptation to compare a CPU and memory to the human brain and I/O devices to our senses. Information flows into our memory through sight, sound, touch, taste, and smell. Our brain remembers the information, decides to take action, and sends commands to our muscles so that we speak or move around. This analogy is the origin of the term "electronic brain."

Assuming that things are alike because they look alike is a common error. In this case, although there are similarities in structure, computers and humans operate in fundamentally different ways.

Computers Require Explicit Programming

Computers are sensitive to events such as an operator pressing a key on a terminal, a printer finishing a line of print, or an interrupt telling the computer that enough time has passed that it is time to update its clock. Whenever a computer notices an event, it interrupts what it was doing to handle the event.

The first step in handling interrupts is bringing information about the event into the computer's memory. This tells the computer what happened—the event processor finds out the name of the key the operator pressed, tells which printer has finished printing and needs more text to print, or that the clock needs to be changed.

Once the computer knows what happened, it carries out instructions that tell it how to process the event. The event rattles around inside the computer for a while, finally generating some output. Each step along the way is handled by explicit instructions previously entered by a human programmer. The instructions tell the computer exactly what to do under every conceivable set of circumstances. Computers are the ultimate bureaucrats—they have no choice except to handle events according to previously defined procedures because they cannot make up new procedures on their own.

Handling Independent Tasks

Explicit programming suffices for independent tasks such as airline reservations. Although hundreds of seat requests may be processed at the same time, each transaction is essentially independent. The fact that Jones wants to fly between Boston and New York on June 7 has little to do with the fact that Smith wants a seat on the same flight.



Nocturnal Airlines



Computers are unbelievably literal-minded. Tell them, "Give out seats," and that is what they do.

Process engineers would rather give computers general guidelines about manufacturing practice and let them figure out how to handle events as they arise. Factory automation people are interested in AI because computerized factories *must* be rulebased in order to work at all.

Humans Operate with General Rules

In contrast to computers, humans do not operate according to explicit programming. Human memory is not well suited to remembering detailed procedures. When details are important, humans use checklists to make sure nothing is omitted. When details are vital, one human reads the checklist and another makes sure the first does not skip anything. Humans are poor at details, but they are superb at remembering overall rules of behavior and adapting rules to new situations as they arise.

Humans managing complex activities such as space shuttle launches and manufacturing automobiles do not remember every single detail. They learn broad patterns of how the situation ought to operate and adapt to events as they occur.

It takes a great many rules to operate well in a complicated domain. Chess grandmasters memorize on the order of 50,000 rules of chess strategy such as "Keep the knights away from the edge of the board."¹ Computers can store 50,000 rules, but cannot search through them rapidly enough to plan chess moves fast enough to win.

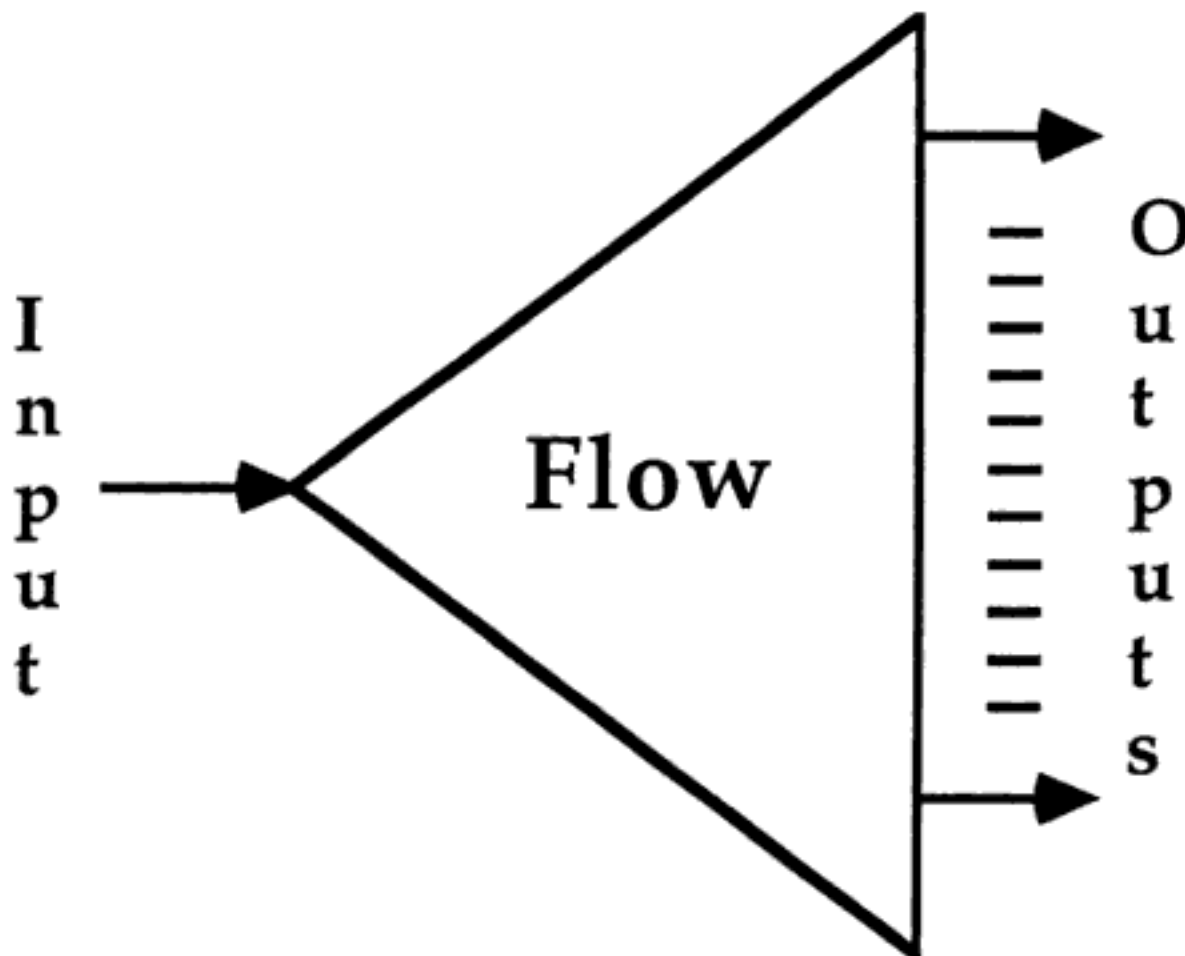
Human inability to remember details leads to mistakes. Space shuttle engineers knew that cold weather could endanger the seals on the booster rockets, but political considerations kept this information from being presented to the launch directors who made the decision to lift off. Human frailty with respect to detail caused the accident.

If computers could accept a little more rule-based programming, the combination of human ability to recognize overall patterns and the computer's ability to keep track of details would be an unbeatable combination.

¹ *The Sciences of the Artificial*, by Herbert A. Simon (Cambridge Mass.: MIT Press 1982), p. 106. Given a brief glance at a chessboard, chess grandmasters can remember 100% of the board position, masters about 90%; novices remember only five or six pieces. If the positions of the pieces are random instead of drawn from a valid game, grandmasters do about as well as novices in remembering the board layout. Masters seem to break a board down into "standard" chunks containing only a few pieces. See also *Thought and Choice in Chess* by A. DeGroot (Hawthorne, N.Y.: Mouton, 1965).

Computers Use Divergent Logic

When a computer accepts an event and it starts rattling around inside, the computer usually produces much more data than it received. A terminal operator at John Hancock enters a policy number and the computer prints the entire insurance history. The billing operator at the telephone company types a telephone number and the computer displays the last six months' bills. Kids push a button on an arcade game to fire a missile and the entire screen changes. The IRS puts in a social security number and out comes a tax history. Computers produce more output than input—they are all mouth and no ears.



Information enters a computer, rattles around inside, and produces vast quantities of output.

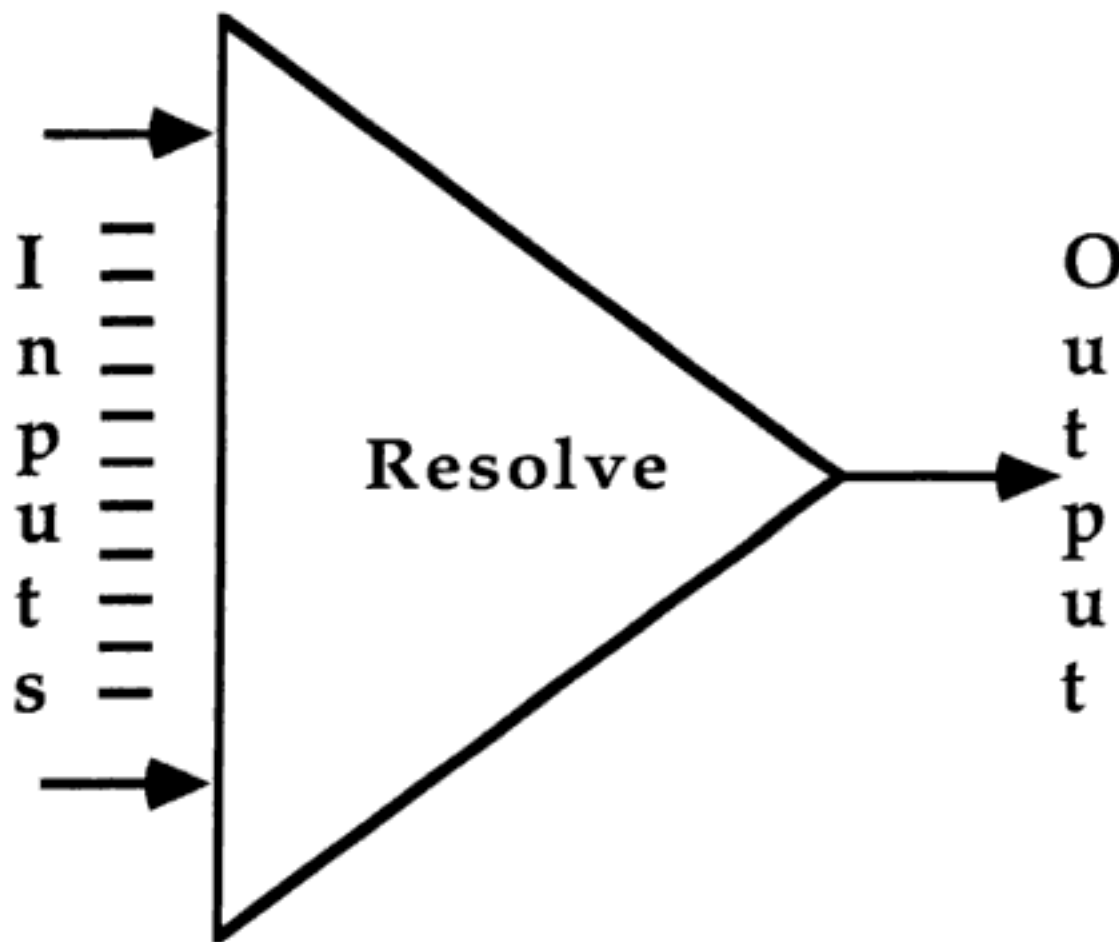
This is partly because manufacturers know how to build better computer output devices than input devices. A television monitor that generates thirty pictures per second costs a few hundred dollars. Attaching a TV camera to a computer costs thousands of dollars, and the computer cannot make much sense of the picture anyway. Entry clerks type data at peak rates of ten characters per second, whereas line printers type data at twenty lines per second. Lack of good ways to get sensory information into computers has held back efforts to write programs that process real-world information effectively.

Humans Use Convergent Logic

Humans have much higher capacity input devices than output devices. Our eyes can resolve ten separate images per second. There are more than one million nerve fibers in the optic nerve

running from the back of each eye to the brain, and each nerve transmits one point in the picture to the brain. Each eye sends the brain ten *million* picture elements per second, and the brain has no trouble processing this flood of information. Visual images are compared with memory, and decisions of what to do are made quickly enough to avoid threats.

When sending information out, the brain can call on only 600 muscles in the entire body. Some muscles are used for overhead functions such as breathing, digestion, and heartbeat. Some are really used for input because they move the ears and swivel the eyes. Having so few muscles for output, the brain can send out only a tiny fraction of the information it receives. Humans are all ears and no mouth.



Humans have tremendous input bandwidth and limited output capability.

The earliest known human writing dates back to at least 3000 B.C. No one knows when speech was invented, but there are indications that people did not use speech very long before inventing writing. The fact that children can be taught to read within a year or two of learning to talk is supporting evidence for the idea that reading is a natural human activity in that it uses neural pathways that occur naturally in the brain. Skills such as algebra, geometry, and solving differential equations, all of which are taught relatively late in the education process, seem not to be natural human activities.

- Computers do one thing at a time, humans worry about many things at once. Software techniques called "multiprogramming" or "multitasking" make computers seem as if they are doing many things at once, but that is only a clever fake. The computer spends a little time on one task, then drops it for the next task. By switching between tasks quickly enough, the computer gives the illusion of working on all the tasks. Humans, in contrast, really do pay attention to many things at once. Engineers can get involved in a job and forget to eat, but no matter how hard we think, people react to sudden noises and know when to visit the toilet. Some people can read and converse at the same time, and Julius Caesar could dictate seven letters at once.

- Computers do things that are hard for humans; humans do things that are hard for computers. There seems to be a relationship between how young humans are when they learn a skill and how difficult it is to program a computer to do it. Infants learn to recognize their mothers early on. Despite a lot of research, no one has programmed a computer to recognize faces well. Toddlers can converse, but no computer comes close to their ability to understand speech. Children 10 or 12 years old can learn to drive cars, and a major research effort is slowly leading to an autonomous computerized vehicle that drives itself on highways. On the other hand, it takes a college education to do calculus or to learn accounting, and computers do both these tasks fairly well. The later humans learn a skill, the easier it is to teach computers.

Handling Life's Little Problems

Computers and humans are structured differently because they are designed to solve different problems. Suppose that I am bounding across the desert. I peek out from behind a rock and encounter an object. There is a question I must answer *very* quickly, and I had better answer it correctly: "Must I flee?" Get the wrong answer, and I probably never have to worry about anything else again, ever.

Most animals are biased toward flight. If a cat hears a sudden noise, it starts to run and looks back to see if it is safe to stop. Cats have been domesticated for a long time and are seldom threatened with dismemberment, but the flight reflex remains.

Assuming that I need not flee, I consider the question "Must I fight?" These two questions are so closely related that behavioral psychologists talk of the "fight or flight" reflex, but

there is more to life than fleeing and fighting. Having decided not to fight, I ask, "May I feed?"

Having found an object that I need not flee or fight and on which I cannot feed, there is only one more possibility—"May I fondle?" If I need not flee or fight and cannot feed or fondle, I do the only sensible thing and forget it.

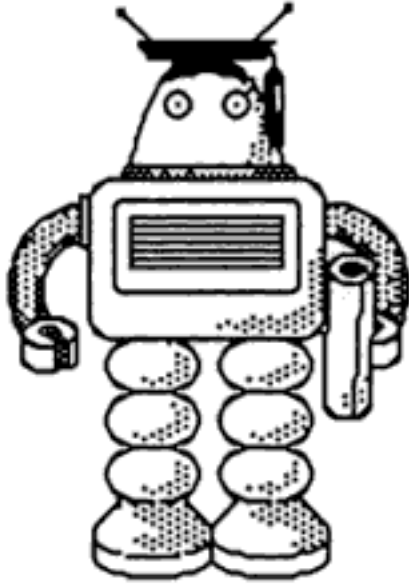
The human senses, nervous system, and memory are designed to answer these four questions quickly and accurately. Our eyes constantly flicker to and fro, scanning for threats. No matter how dull our lives, we listen continually for threatening noises.¹

Human hardware is designed to answer these four vital questions quickly and accurately. All else is culture. Living creatures have no inherent interest in whether an engine has the right number of pistons, or whether airplane seats are assigned correctly, or if the balance sheet balances—these are cultural matters.

Who Needs Intelligent Computers?

These functional differences are the reason why we do not think there will be much demand for intelligent computers as long as computer intelligence is like human intelligence. People have

¹ Human male vision systems seem to have special cells that recognize women although women seem not to have special hardware to recognize men. See *The Amazing Brain*, by R. Ornstein and R. Thompson (Boston, Mass.: Houghton Mifflin 1984), pp. 170-171, for a discussion of some of the more obvious differences between male and female brains.



3

Results of AI Research

Ideas have flowed out of AI research labs in a steady stream since the field was founded. Some ideas found immediate application, some seem about to become useful, and many are unlikely to become valuable in the foreseeable future. In this chapter I discuss some results of AI research and explore their impact.

Human thought is such a complicated and mysterious process that even partial human emulation is a long way off. Despite this unfortunate delay, however, there are many results of AI research in commercial use.

Time Sharing

Early computers were so expensive that an entire university had to make do with only one, even with a government grant. Researchers tired of waiting for their turn on the computer and developed software to let many people share a computer at once. Most of the early work was done by project MAC at MIT. The name stood either for "multiple access computing" or for "machine-aided cognition" depending on whether professors were seeking grants for AI or for time sharing. Cynics stated that it stood for "man against computer."

Man-Machine Interfaces

One of the factors limiting computer use is difficult user interfaces. Hackers who appreciate computers for their own sake memorize arcane commands, but people who are more interested in getting the job done than in playing with the tool demand computers that can be used with little training.

People who do not use computers for a living usually ignore their computers except when they need them. If the commands are too complicated to remember between work sessions, the computer is essentially useless.

AI researchers are not always computer experts. They have backgrounds in biology, physiology, cognitive psychology, anthropology, and physics. Researchers are less tolerant of inconvenient computer tools than commercial users because they get paid for results, not for putting in their time. Not only that, programs to mimic human behavior were so large that even hackers could not cope unaided. In order to make progress, it was necessary to make research computers easy to use.

Making computers easy to use turned out to be so difficult that an entire subdiscipline grew up to address user convenience. Xerox's Palo Alto Research Center pioneered "windowing," a technique of dividing a computer screen among several tasks, as shown in figure 3.1.

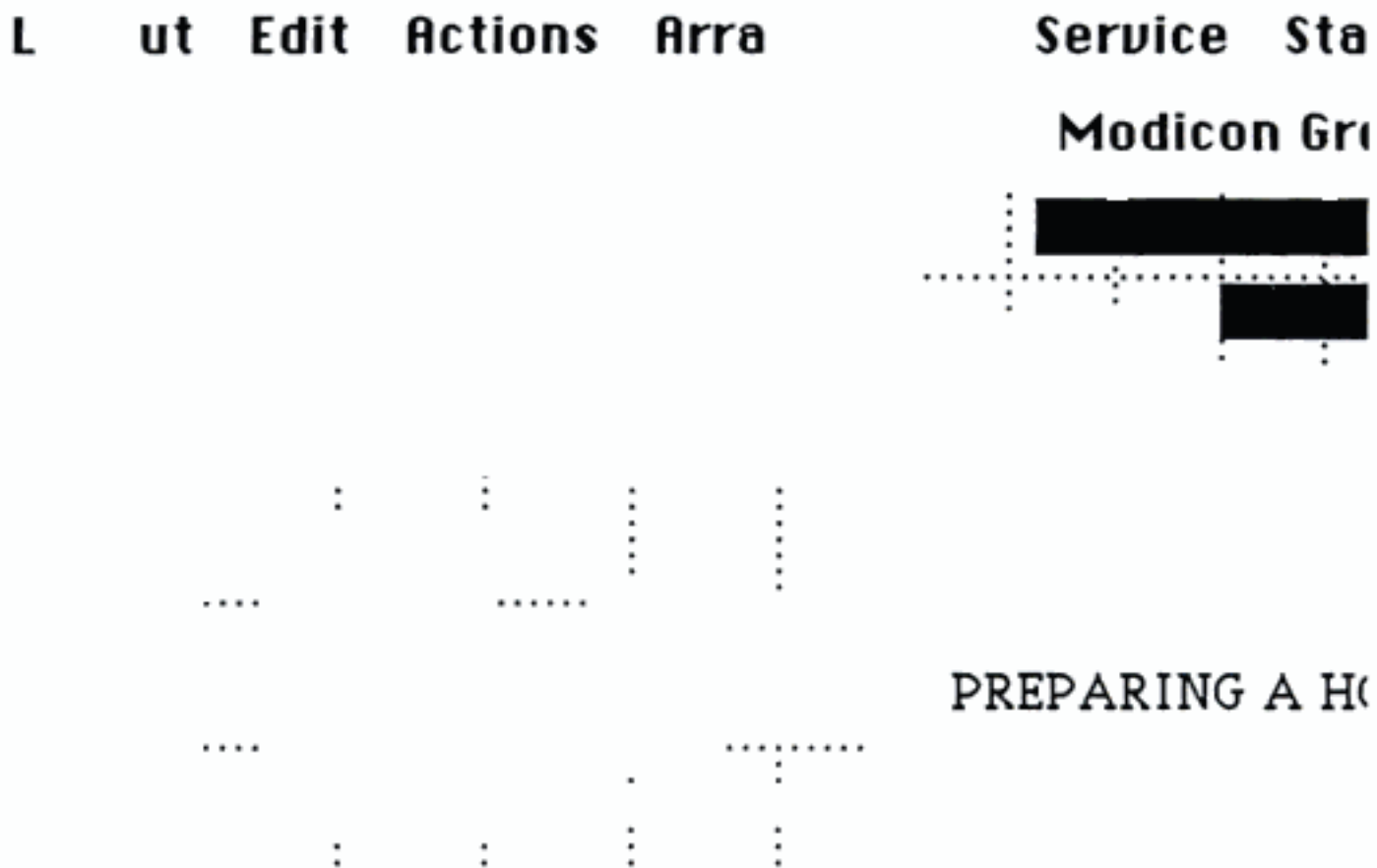


Figure 3.1
 A portion of a window display. Output from many different programs can be viewed on the screen at the same time. Keyboard input goes to only one program at a time, of course.

Shared displays seem to be far too busy until users become accustomed to them, then most people wonder how they got along without them. Users edit programs in one part of the screen, look at input to their program in another, and see output from the program in a third. Displaying many tasks at once is like spreading many different sheets of paper on a desk—it helps people keep track of what they are doing.

Windowing spread from Xerox to Apple Computer, where it became the basis for the Apple Macintosh. Developing the software to manage overlapping windows smoothly is difficult and required many insights to get it right the first time. Even now, there are very few programming groups who can develop acceptable windowing systems.

Window systems are beginning to appear on engineering workstations and personal computers, but it has taken years of trial and error to get the software to work correctly. Even IBM fans concede that the Macintosh is easier to use. This ease of use is rooted in AI research.¹

Natural Language Processing

Researchers have struggled to write programs that understand human languages since the late 1960s. It was originally thought that computers would soon translate freely from English into any other language. Software gurus spoke confidently of holding telephone conversations while a computer translated. Alas, computers *still* cannot translate human languages; translation evidently requires more knowledge about the meanings of words than anyone knows how to program into a computer.

Computers can understand human language *if the domain of discourse is simple enough*. The first program that demonstrated this was called ETAOIN SHRDLU², or SHRDLU for short, and was written by Terry Winograd in 1972.

¹ Even though Xerox did the fundamental research on how window systems ought to operate, Apple reaped the commercial rewards. The original idea was that the computer display should emulate a desk top. Papers hide one another on a real desk, so Xerox designed overlapping windows that hid one another. Now that Apple sells half a billion dollars worth of computers with overlapping windows every year, Xerox has decided that the fact that papers hide one another on a desk is a bug, not a feature. Instead of one window covering another so that the lower window is hidden, the lower window should shrink to make way for the upper window.

² "ETAOIN SHRDLU" is the alphabet ordered by letter frequency in English prose. The most common letter is "e" followed by "t" and so on. This order was the basis of the keyboard layout for the Linotype system. Typesetters would

In this case, Intellec™ finds "New York" in both the city column and the state column and asks the user to resolve the ambiguity. When analyzing the query, "Which of the New York employees live in Buffalo?" Intellec™ knows that cities are located in states and that "New York" refers to the state.

Human languages are riddled with ambiguities. Children learn to deal with ambiguous statements over a period of years, but computer scientists have a great deal of work to do before computers can progress much beyond data retrieval.

Users find that Intellec™ increases the volume of database requests by a factor of 10 or more. This is not surprising. Whenever anything becomes easier, people do more of it whether it is worth doing or not. Increasing the volume of retrieval requests has a predictable result—the first \$10 million in Intellec™ sales generated about \$30 million worth of additional business for IBM. Artificial intelligence is a *great* way to sell more computers.

Drawings of the "blocks world" are common in AI publications. Writing a program to move a picture of a robot arm is much easier than writing a program for natural language processing.

Expert Systems

Expert systems get a disproportionate share of media attention. Reporters seem to feel there is something magical about knowledge-based systems, as if all prior efforts were based on

ignorance. Reporters' interest in expert systems may be justified, because some expert systems are beginning to make money.

The central idea behind expert systems is expressing computer knowledge in the form of rules such as

if the AC light and the DC light on a power supply are both off
then check to make sure that the device is plugged in

Rules are a handy way to express certain kinds of human expertise, just as equations are suitable for expressing other kinds of information. In chapter 10 I explain that there is a great deal more to expert systems than rules.

Logic Programming

Expert systems are based on the rules of mathematical logic, just as conventional programs are based on the rules of arithmetic. Prolog, which stands for "programming in logic," was invented to express mathematical logic in computer notation. This is necessary in order to use rules in computer programs. The result is a style of computer usage called "logic programming" or "rule-based programming."¹

Prolog is a primitive language just as Fortran was when first introduced, but the introduction of logic notation to computers was as important a development as the first use of algebraic programming in the late 1950s. Expert systems should become more intelligent and easier to use as logic languages develop.

In chapter 8 I explain the basics of rule-oriented programming in order to prepare for a discussion of expert systems in chapter 9. Logic programming is discussed in more detail in chapter 10 and Prolog in chapter 13.

Object-Oriented Programming

Object-oriented programming is a way of structuring programs so that a particular type of data and the parts of a program that process that type of data are combined. Data and the functions that process them are collectively called an "object." Objects are manipulated as a unit; code and data cannot be separated.²

¹ Prolog was not the first language to use rules. Planner seems to have been the first rule-based language. As with most innovations, the roots are difficult to unravel because so many people contributed to the idea.

² Xerox developed the first commercial object-oriented programming system. Their Smalltalk language reached a reasonably stable state by 1972. Smalltalk systems are available for the IBM PC, the Apple Macintosh, and for a line of AI workstations marketed by Tektronix Inc. No one has figured out

Whenever a user changes a number in an input cell, all output formulas that refer to that number are recalculated. When these calculations are completed, formulas that refer to the changed output cells are recalculated, and so on until all affected numbers have been updated. The formulas apply constraints on the values displayed in their cells. These constraints are propagated through the entire spreadsheet whenever an input is changed.

Constraint propagation was an old idea when two entrepreneurs, Dan Bricklin and Bob Frankston, introduced the first commercial spreadsheet program in 1979. Their innovation was to package the idea of constraint propagation behind a matrix of cells that looked like multicolumn accounting paper. Financial people who made their living manipulating sheets of numbers found that Visi-Calc saved considerable time and bought enough copies to found an industry.¹ There are probably many ideas of equal commercial merit² lurking in AI labs just waiting for someone to market them properly.

AI Technologies That Do Not Work Well Yet

There are many areas of AI research that seem to be about to produce profitable results. It makes business sense for a technical company to be ahead of the competition, but not too far ahead. If a product is too advanced, the market is not ready to accept it, and it costs too much to educate customers enough to get them to buy. Taking technical risks to enter new markets is

¹ After enough copies of Visi-Calc had been sold to demonstrate that there was a market for spreadsheet software, an employee of the inventors named Mitch Kapor suggested several improvements. The inventors were busy trying to control their explosive growth and told Mr. Kapor to get back to work. He quit, raised money, and founded Lotus Development Corp to sell a rival spreadsheet called Lotus 1,2,3 which came to dominate the spreadsheet market. It is common for technical innovators to be eclipsed by others who understand the market implications of the invention better than the original entrepreneurs.

² In the sense that a natural language is anything that many people know, spreadsheets can be thought of as natural language computer interfaces. There is nothing spreadsheet users do that cannot be done by writing a program in Basic or C or Lisp, but most people are unwilling to learn these languages. By turning computer programming from arcane mumbo-jumbo into the process of manipulating familiar columns of numbers, spreadsheets made computers accessible to millions of new users, earning millions of dollars in the process.

There are probably other opportunities to computerize "languages" that are already known to potential customers. The statement "knit one, purl two decreasing every row" is perfectly intelligible to millions of knitters. There may be a market for a computer that understands "knitting language."

often worthwhile. Most old products are difficult to sell except in special circumstances. Ivory Soap goes on forever, but technical products become obsolete rapidly.

The ideal position is *just slightly* ahead of the competition—advanced enough to command a premium price, yet not so far ahead as to make customers nervous or make life difficult for component suppliers. Using yesterday's technology to solve tomorrow's problems seems to pay off well.

Investors and entrepreneurs sift technology for commercial opportunities. Sometimes a company tries to develop a new product even if they think the project is likely to fail because the potential rewards are so great. That is why so many companies have developed industrial robots—the present market is small, but sales would be immense if robots could be made practical.

At other times, companies develop products even if they are not sure who will buy them because the technology is so fascinating. Once in a great while, a product developed for the sake of technology turns out to be successful, but technology-driven products often end in commercial failure.

Failure is more common than technical innovators like to admit. Engineers enjoy developing new features, but users only pay for benefits. Product developers often forget that users do not buy technology, they pay to have their problems solved. If technology is absolutely necessary to solve the problem, customers will tolerate it, but only until a less technical solution is available.

Robotics and the Factory of the Future

Robots are in the "barely viable" stage. Despite tremendous interest in robots and many experiments using robots, industrial robot sales are only a bit more than \$300 million per year.¹ In economic terms, annual robot sales are about as important as the movie industry and come to about one month's computer sales for Digital Equipment Corporation.

The Japanese claim to have many more robots working in their factories than Americans do. This is partly a matter of definition. The Japanese define a robot as any machine that picks up parts and moves them around. Americans think of "pick-and-place" devices as machines, not robots, and argue that Americans have as many pick-and-place robots as the Japanese.

Regardless of which country is using robots more effectively, the fact remains that programmable controllers, which are an

¹ Industrial robot sales should reach \$370 million in 1987 and \$1 billion by the mid-1990s according to *Fortune* magazine (September 14, 1987, p. 82).

older form of factory automation, sell at least \$500 million per year in the United States. For all the interest in robotics, robots have had a minimal impact on most factories. The robot market is limited because robots are difficult to use under realistic factory conditions.

The popular idea behind factory robots is C3PO in a hard hat slaving away on the assembly line, but this is unrealistic. No one expects robots to have a major impact on high-volume manufacturing. Manufacturing more than 1,000 parts per day usually justifies the cost of custom-made machines. A machine designed to do just one job costs less than a robot that can do many different jobs. Machines used in high-volume production usually wear out before the part becomes obsolete, so it makes no sense to pay extra for a flexible robot.

Low-volume manufacturers would like to buy one machine and make many different parts with it. Most parts are produced in batches of fifty or fewer, and this is the natural market for robots. The problem is that factories are full of machines that were designed to be operated by humans, and parts designers are accustomed to designing parts to be made by such machines. A computer-operated machine has different capabilities from a human-operated machine. Before robotic machines can be used to full advantage, parts designers need to learn new techniques to design parts so that computers can make them easily.

Difficulties Controlling Robots Computers which control robots are extremely difficult to program. Making each part requires a separate computer program to tell the robot how the part should be made. Like human machinists, robot machinists need instructions which they can understand.

There simply are not enough part programs available to make robots useful. Just as personal computers could not achieve wide sales until there were enough programs to make people want to buy them, robots will find limited application until there is enough high-quality software available.

It costs so much to program a robot to make a part that unless a factory can spread the programming cost over many parts, it is usually cheaper to make them manually. Unfortunately for robot vendors, if a customer needs enough parts to justify the cost of programming a robot, engineers can often justify the cost of special tooling. If the order is too small for special tooling, it is usually cheaper to make the parts by hand.

Sensor Problems The main difficulty in programming robots is that computer programs must be precise. Every possibility must

will continue to use organic robots. Replacing human factory workers with machines is a gradual process that is not happening nearly as fast as pundits claim.

Speech Recognition

Devices that can recognize a few hundred spoken words have been on the market for several years. Most speech recognition systems are trained separately to recognize each user's voice. Limited vocabularies are useful in materials handling applications because a warehouse clerk can tell the computer what is in a box while moving the box. The operator can do the job and tell the computer about it at the same time.

The biggest problem is deciding where one word stops and the next begins because people do not pause between words when speaking. Some researchers speculate that the speech recognition problem cannot be solved until computers understand what they are hearing. Humans seem to recognize words from context as much as from hearing them precisely.

There would be an immense market for a reasonably priced device that could recognize 2,000 to 3,000 words—executives could tell the computer to "take a letter."¹

Computer Vision

It is easy to describe how computer vision should work: Attach a TV camera to a computer, point the camera at a scene, and have the computer list the objects in the camera's field of view. Vision is far easier done than said. People see without difficulty, but no one can say how vision works. There is a saying: "There is more to vision than meets the eye."

The earliest efforts were directed at "feature extraction," which finds significant parts of a scene and tries to make sense of them. Researchers are not exactly sure what features our eyes extract from a visual image but have shown that the eye is sensitive to edges, color contrast, texture, and certain kinds of shading.

Having located features in a visual scene, the problem is to match the features with objects in memory to see what is going on. Even though a Nobel prize was awarded for research in how simple creatures such as snails store information, we do not

¹ There are many speech recognition systems on the market. Most of them have vocabularies limited to less than 100 words and have to be trained to recognize each individual speaker. IBM has a prototype that recognizes thousands of words, but it has to be trained for each speaker and is too expensive for general use.

fully understand how features are stored in memory or how stored features are matched against features found by the eyes.

Feature extractors are good enough that vision systems can be used for certain kinds of quality inspection and measurement and can read letters written clearly in several different fonts. Computer vision is at a stage similar to robots: If it worked well in factories there would be an enormous market, but it does not.

Like robots, vision systems are difficult to use. Camera placement is critical, light levels have to be adjusted precisely, focus must be maintained, lenses must be kept clean, and many other finicky details must be attended to. When vision systems work, they work well, but keeping them working seems to be beyond the capabilities of most factory maintenance people.

Automatic Programming

Programming is the act of telling a computer how to do a task. Human programmers must first understand the task, then explain how to do it in a language the computer understands. Computers always do exactly what they are told. The difficulty is telling them exactly what we mean.

Computers are unbearably literal minded. When a dog is on the other side of a fence and I tell it, "Come here," the dog usually goes away and runs around the fence in order to carry out the command. If a dog were no more intelligent than a computer, it would run straight at me until it hit the fence, then keep pawing away trying to "Come here" until its legs wore off.

The computer obeys the letter of the command and either comes straight to me or busts a gut trying. The dog does not do *exactly* as it is told—it starts out by going away when told to come—but owners prefer that dogs obey the spirit of the command rather than the letter. Teaching children and subordinates to do what I mean rather than what I say is difficult and frustrating, so it should come as no surprise to learn that teaching computers to do what users mean instead of what they say is difficult and frustrating.

There are signs that computers will be able to generate certain kinds of programs automatically. Some mechanical design systems generate tapes to guide computer-controlled machines to cut out parts. Programs called "application generators" translate pseudo-English task descriptions into computer programs. Application generators do not quite go directly from a memo

stupendous feats, computers cannot figure out the steps to move from point A to point B.

Factory managers must plan how to move equipment and materials. Large items are rotated and occasionally moved backward in order to get to the proper place. People do such things easily, but computer programs for planning are still a major research area.

Learning from Experience

A few game-playing programs record victories and defeats and attempt to draw on the records for hints of how to play. Expert systems learn, but only by having humans fix bugs in rules or write new ones. No computers learn automatically, even from carefully chosen examples.

Humans are born knowing how to learn, and that seems to be enough to get started. Our learning is extremely flexible, but it is conditioned by what we must learn. People who grow to maturity in jungles where the horizon is not visible have difficulty seeing horizontal lines. Eskimos raised in the Arctic have difficulty seeing vertical lines. Adults find it hard to learn the sounds of a new language, but children find it easy.

Analogies

People understand analogies well—apt analogies are an effective way to explain unfamiliar concepts. Computers cannot generalize their abilities from one task to the next. Suppose that a robot is programmed to paint doors in an automobile factory. If the design changes, engineers would like to say, "Just like the one last week, except a little more paint along the bottom," but this does not work. Programming starts from scratch as if the robot had never seen a door before.¹

Creativity

Human creativity seems to be based on unforeseen combinations of existing ideas. Henry Ford combined the idea of the automobile with assembly line techniques developed by Sears Roebuck, and the mass automobile industry emerged. Automobiles had existed for a long time and Sears had been

¹ Patrick Winston, director of the MIT AI Lab, has written a program that draws analogies between situations in different Shakespearean plays. This work is described in *Learning and Reasoning by Analogy* (AI Lab Memo 520, April 1979) and *Learning by Augmenting Rules and Accumulating Sensors* (AI Lab Memo 678, May 1982).

using assembly line methods to fill catalog orders for years, but no one had combined these ideas before Ford.

Thomas Edison developed the electric light bulb by running electricity through a carbon thread enclosed in a vacuum to keep it from burning up. Carbon arc lights were well known but the electrodes burned too fast to be practical for homes. Vacuums had been investigated for a hundred years before Edison, but he was the first to put the two together.

Computers with unlimited disk storage ought to be able to remember more knowledge than humans and therefore have more to draw on when making creative associations. The difficulty is that the number of possible associations grows rapidly as the amount of knowledge increases. It takes human insight to know which combinations to try.

Nobody knows how people limit the number of possibilities to a manageable number. When researchers figure that out, it may be possible to generate creativity mechanically. Until then, we have to make do with human creativity, erratic as it may be.¹

The State of the Art

To speak of the "march of science" is to speak nonsense. Science does not march; it crawls on its belly, tripping over every leaf and twig in its path. AI researchers originally thought that they could knock off vision and speech recognition over a summer or two, then get on to the *real* problems. It has not turned out that way; vision and speech recognition remain mysterious. Human input/output devices are complex enough to baffle the most intense scrutiny, to say nothing of the CPU itself.

AI techniques and ideas work well enough for a number of commercial applications, however, and some of them are discussed in the next chapter.

¹ Doug Lenat developed a program called "AM," which makes mathematical discoveries. AM rediscovered many truths already known to human mathematicians but has not come up with anything people did not already know. See *Knowledge-Based Systems in Artificial Intelligence*, by R. Davis and D. Lenat (New York: McGraw Hill, 1982). Lenat also wrote a program called "Eurisko," which could learn to play certain games well enough to defeat human players.

languages. The Lisp development tools described in chapter 6 effectively doubled programmer productivity during the project. This is an example of using the indirect results of AI research to increase productivity on a conventional project.

GigaMos Systems Inc.: Process Control

GigaMos Systems Inc. (GSI) designed a specialized expert system called Picon to monitor and control real-time processes. GSI labored to make Picon easy to use. Customers can develop the rules for their own applications within weeks of installing a Picon system.

Picon is one of the few commercial expert systems that was designed for real-time process control. Most expert systems are not fast enough to handle data that change as rapidly as process variables in a chemical plant.

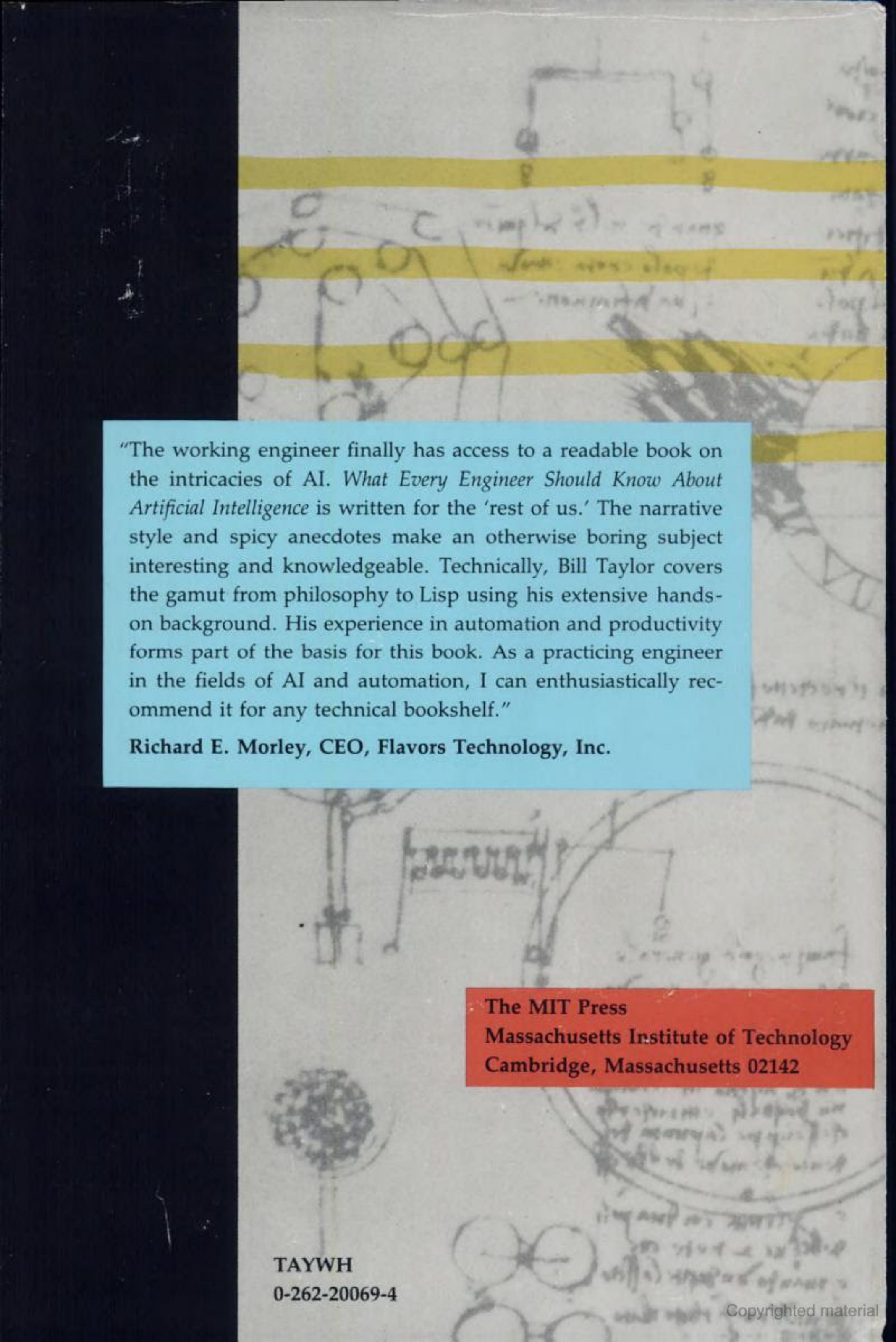
A chemical plant has as many as 20,000 variables. Alarm monitors look at most variables and notify people if a variable moves out of a specified range. Operators must decide what to do about each alarm. They can ignore an alarm because they feel that the sensor has malfunctioned or because the process will settle down by itself, or they can decide to intervene.

Power plants and chemical refineries are collections of large subassemblies that interact in complicated ways. When something goes wrong in a subsection of the plant, its variables drift out of tolerance and alarms are signaled for those variables. Fixing the problem would be easy if the rest of the plant continued to work properly, but the problem usually spreads beyond its point of origin. It is quite common for the first alarm to come from a part of the plant that is actually OK but that has been disturbed by a problem somewhere else.

Alarm processing gets exciting—a nuclear power plant can generate as many as 800 alarms within 2 minutes of a major disturbance. The control panel has as many as 3,000 lights on it, and within 5 minutes, essentially all the lights turn red.

Without paying close attention to the exact sequence in which red lights come on, there is no way to figure out what caused the problem quickly enough to do anything about it. Picon remembers the order in which alarms occur and uses its process knowledge to advise the operator what to do.

Picon cannot scan 20,000 variables fast enough to analyze all the alarms, so it uses a process called "focusing" to decide which data to collect. Focusing rules look at a few variables that characterize major subsystems and ignore other variables. When the major variables indicate that a part of the plant is not behaving



"The working engineer finally has access to a readable book on the intricacies of AI. *What Every Engineer Should Know About Artificial Intelligence* is written for the 'rest of us.' The narrative style and spicy anecdotes make an otherwise boring subject interesting and knowledgeable. Technically, Bill Taylor covers the gamut from philosophy to Lisp using his extensive hands-on background. His experience in automation and productivity forms part of the basis for this book. As a practicing engineer in the fields of AI and automation, I can enthusiastically recommend it for any technical bookshelf."

Richard E. Morley, CEO, Flavors Technology, Inc.

**The MIT Press
Massachusetts Institute of Technology
Cambridge, Massachusetts 02142**

**TAYWH
0-262-20069-4**