Machine Intelligence and Robotics: Report of the NASA Study Group

National Aeronautics and Space Administration
Washington, D.C.

March 1980

Machine Intelligence and Robotics: Report of the NASA Study Group

FINAL REPORT

March 1980

Preface

The NASA Office of Aeronautics and Space Technology (OAST) has established the goal of providing a technology base so that NASA can accomplish future missions with a several-orders-of-magnitude increase in mission effectiveness at reduced cost. To realize this goal, a highly focused program must be established advancing technologies that promise substantial increases in capability and/or substantial cost savings. The Study Group on Machine Intelligence and Robotics was established to assist NASA technology program planners to determine the potential in these areas. Thus, the Study Group had the following objectives:

(1) To identify opportunities for the application of machine intelligence and robotics in NASA missions and systems.

(2) To estimate the benefits of successful adoption of machine intelligence and robotics techniques and to prepare forecasts of their growth potential.

(3) To recommend program options for research, advanced development, and implementation of machine intelligence and robot technology for use in program planning.

(4) To broaden communication among NASA centers and universities and other research organizations currently engaged in machine intelligence and robotics research.

Foreword

This publication, complete with appendant documentation, is the final report of the NASA Study Group on Machine Intelligence and Robotics. As you will note in the Introduction, Section I, the report tells why the Study Group was gathered together; and what the Group felt and hoped to do. You will see that Section II is a timely tutorial on machine intelligence and robotics inasmuch as both fields may be really neoteric to a lot of assiduous readers.

NASA's needs and the applications of machine intelligence and robotics in the space program are discussed for you in Sections III and IV. Section V discusses the generic topic, Technological Opportunities, in two subsections, A, Trends in Technology, B, Relevant Technologies, and a third subsection, which is an Appendix on Relevant Technologies. (Don't skip any of these subsections, especially the third, because if you look there, you will find detailed discussions of the conclusions and recommendations which the Group made on each specific machine intelligence and robotics subject or topic.)

After 25 hundred man-hours, the Study Group and the workshop participants arrived at a few prenotions concerning the state of the art situation as it exists in NASA with regard to the machine intelligence and the robotics fields. The study members and workshop participants then conclude that four things may be better in NASA if four recommended items are adopted—as they so wrote in Section VI.

Appendix A tells who the Study Group people are, their organizations, interests, backgrounds, and some accomplishments. The appendix itemizes what the workshop subjects or topics were; and where and when the study actions were done at five locations in the United States. The people-participants (and what they talked about) are also listed for you in Appendix A. Appendixes B (Minsky, 1961), C (Newell, 1969), D (Nilsson, 1974), E (Feigenbaum, 1978), and F (Newell, 1970) are those references which the Group feels will provide support for their conclusions and recommendations.

The Study Group hopes you will read—and that you will find the report valuable and useful for the 1980s.

Carl Sagan, Chairman
Raj Reddy, Vice Chairman
Ewald Heer, Executive Secretary

Machine Intelligence and Robotics
Members of the NASA Study Group
June 1977–September 1979

Dr. Carl Sagan (Chairman)
David Duncan Professor of
Astronomy and Space Sciences
Cornell University

Dr. Raj Reddy (Vice Chairman)
Professor of Computer Science
Carnegie-Mellon University

Dr. Ewald Heer (Executive Secretary)
Program Manager for Autonomous Systems
and Space Mechanics
Jet Propulsion Laboratory

Dr. James S. Albus
Project Manager for Sensor and
Computer Control Technology
National Bureau of Standards

Dr. Robert M. Balzer, Project Leader and
Associate Professor of Computer Science
University of Southern California

Dr. Thomas O. Binford, Research Associate
Artificial Intelligence Laboratory
Department of Computer Science
Stanford University

Dr. Ralph C. Gonzalez
Professor of Electrical Engineering
and Computer Science
University of Tennessee

Dr. Peter E. Hart, Director
Artificial Intelligence Center
SRI International

Dr. John Hill, Staff Member
Artificial Intelligence Center
SRI International

B. Gentry Lee
Manager of Mission Operations
and Engineering, Galileo Project
Jet Propulsion Laboratory

Dr. Elliott C. Levinthal
Adjunct Professor of Genetics
Stanford University School of Medicine

Dr. Jack Minker, Department Chairman
and Professor of Computer Science
University of Maryland

Dr. Marvin Minsky
Donner Professor of Science
Massachusetts Institute of Technology

Dr. Donald A. Norman
Professor of Psychology
University of California at San Diego

Dr. Charles J. Rieger
Associate Professor of Computer Science
University of Maryland

Dr. Thomas B. Sheridan
Professor of Engineering
and Applied Psychology
Massachusetts Institute of Technology

Dr. William M. Whitney
Manager for Information Systems Research
Jet Propulsion Laboratory

Dr. Patrick H. Winston, Director
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Dr. Stephen Yerazunis
Associate Dean of Engineering
Rennsselaer Polytechnic Institute

Ex-Officio Members:

Dr. William B. Gevarter
Program Manager for Guidance and Control
NASA Headquarters

Stanley R. Sadin, Program Manager
Space Systems Studies and Planning
NASA Headquarters

Acknowledgments

The Study Group on Machine Intelligence and Robotics is grateful to a very large number of NASA personnel and other scientists and engineers for essential help in this study. The Study Group especially appreciates the contributions by the following individuals at the various workshop meetings.

Albee, Arden L.
Alsberg, Harold B.
Avizienis, Algirdas A.

Blanchard, David
Burke, James D.

Calio, Anthony J.
Carey, Ted
Casler, V.
Chapman, Robert D.
Claussen, B. A.
Clarke, Victor C.
Cohen, Danny
Cook, Henry
Crum, Earle M.
Cunningham, Robert
Cutts, James

des Jardins, Richard
Dobrotin, Boris M.
Dudley, Hugh J.

Ebersole, Michael
Elachi, Charles

Fero, Lester K.
Ferrell, William R.
French, James R.
Friedman, Leonard
Fu, King S.
Fuller, Samuel

Gilstad, Douglas A.
Goetz, Alexander F. H.
Greenblatt, Richard
Groth, Edward J.
Guttag, John V.

Haberman, Nico
Hall, E. L.
Hecht, Herbert
Henry, Richard C.
Herman, Daniel H.
Horn, Berthold
Hunt, B. R.

Ivie, Charles V.

Knight, Thomas
Kraft, C. C.

Loftus, Joseph P.
Louviere, Allen J.

Manson, Simon V.
McCandless, Samuel W.
McGhee, Robert B.
McReynolds, Stephen R.
Mead, Carver A.
Milgram, David
Mills, Harlan
Mitchell, Q. R.

Newell, Allen
Norris, Henry W.
Nudd, Graham

O'Leary, Brian

Paine, Garrett
Perlis, Alan
Popma, Dan C.
Porter, James
Powell, Robert V.

Quann, John J.

Ratner, Justin
Rasool, Ichtiaque S.
Rennels, David A.
Rose, Press
Rosenfeld, Azriel

Schaeffer, David
Selfridge, Oliver
Shaw, Mary
Smith, Brian
Smith, George W.
Smith, William L.
Sos, John Y.
Steurer, W. H.
Sutherland, Ivan
Swain, Philip H.
Swanlund, George

Teitelman, Warren
Tenenbaum, Jay M.
Textor, George P.

von Puttkamer, J. H.
von Tiesenhausen, George F.

Weinstein, Meir
Williams, Donald
Williams, R. J.

Young, A. Thomas

Zegalia, John B.

The Study Group on Machine Intelligence and Robotics also gratefully acknowledges the following authors and publishers for granting permission to use all, portions, or brief excerpts from their publications and books.

U.S. Office of Management and Budget. *Federal Data Processing Reorganization Study*. National Technical Information Service, Washington, DC.

Nils J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Company, Palo Alto, California. By permission of the author and publisher.

Herbert A. Simon. *The New Science of Management Decision*, revised edition, copyrighted 1977, pp 102-167. By permission of Prentice-Hall, Inc, Englewood Cliffs, New Jersey.

Ewald Heer. "New Luster for Space Robots and Automation," in *Astronautics & Aeronautics*, Volume 16, No 9, pp 48-60, September 1978, copyrighted by AIAA. By permission of the American Institute of Aeronautics and Astronautics, Inc, New York.

National Academy of Sciences, National Research Council, Committee on Planetary and Lunar Exploration, Space Science Board, Assembly of Mathematical and Physical Sciences. *Strategy for Exploration of the Inner Planets: 1977-1987*. Washington, DC, 1978.

Marvin Minsky. "Steps Toward Artificial Intelligence," pp 406-450, in *Computers and Thoughts*, edited by Edward A. Feigenbaum and Julian Feldman, copyrighted 1963 by McGraw-Hill, Inc. Reproduced by permission of the Institute of Electrical and Electronics Engineers, Inc., successor to the Institute of Radio Engineers, original copyright holder of the *Proceedings of the Institute of Radio Engineers*, January 1961, Vol 49, pp 8-30.

Allen Newell. "Heuristic Programming: Ill-Structured Problems," chapter 10, pp 361-414, in Publications in Operations Research, *Progress in Operations Research: Relationship Between Operations Research and the Computer*, Volume III, edited by Julius S. Aronofsky, of Operations Research Society of America, Publications in Operations Research Number 16, David B. Hertz, editor, copyrighted 1969. Reprinted by permission of John Wiley & Sons, Inc., New York.

Nils J. Nilsson. "Artificial Intelligence," pp 778-801, in Volume 4, *Information Processing 74*, Proceedings of IFIP Congress 74, organized by the International Federation for Information Processing, Stockholm, Sweden, August 5-10, 1974, Jack L. Rosenfeld, editor, copyrighted 1974. Reprinted by permission of North-Holland Publishing Company, Amsterdam, The Netherlands.

Edward A. Feigenbaum. "The art of artificial intelligence – Themes and case studies of knowledge engineering," pp 227-240, in the *Proceedings of the National Computer Conference – 1978*, copyrighted 1978. Reproduced by permission of AFIPS Press.

Allen Newell. "Remarks on the relationship between artificial intelligence and cognitive psychology," pp 363-400, in *Theoretical Approaches to Non-Numerical Problem Solving*, Part IV, edited by R. Banerji and M. D. Mesarovic, copyrighted 1970. Reprinted by permission of Springer-Verlag, New York.

Acronyms

ADA	automatic data acquisition
AI	artificial intelligence
AIAA	American Institute of Aeronautics and Astronautics
AP	automatic programming
ARPA	Advanced Research Projects Agency
ARPANET	ARPA network
BBN	Bolt, Beranek, and Newman, Inc.
CCD	charged-coupled device
CMU	Carnegie-Mellon University
DARPA	Defense Advanced Research Projects Agency
DMA	Defense Mapping Agency
DOD	Department of Defense
DODI	DOD instruction
ERDA	Energy Research and Development Administration
EROS	Earth resources observation satellite
FTSC	fault tolerant space computer
GPS	general problem solver
IBM	International Business Machine Corporation
IC	integrated circuit
ISI	Information Science Institute
JPL	Jet Propulsion Laboratory, Caltech
JSC	Johnson Space Center, NASA
KRL	knowledge representation language
LANDSAT	land resources satellite
LISP	list processor
LSI	large scale integrated
MDP	master data processor
MI	machine intelligence
MIT	Massachusetts Institute of Technology
MM64	Mariner Mars 1964
MM69	Mariner Mars 1969
MOS	metallic oxide semiconductor
MPP	massive parallel processor
MSFC	Marshall Space Flight Center, NASA
MVM	Mariner Venus Mars
MV62	Mariner Venus 1962
n	number
NASA	National Aeronautics and Space Administration
NBS	National Bureau of Standards
NCR	National Cash Register Company
NEEDS	NASA end-to-end data system
NSF	National Science Foundation
OAST	Office of Aeronautics and Space Technology, NASA
OCR	optical character recognition
ORB	orbiter
OTV	orbital transfer vehicle
POCCNET	payload operations computing cycle network
R&D	research and development
S/C	spacecraft

SEASAT	ocean resources satellite
RTOP	research and technology objectives and plans
SMU	systems machine unit
SRI	Stanford Research Institute
STDN	space tracking data network
USGS	US Geological Survey

Contents

Figures

Tables

Section I
Introduction

The NASA Study Group on Machine Intelligence and Robotics, composed of many of the leading researchers from almost all of the leading research groups in the fields of artificial intelligence, computer science, and autonomous systems in the United States, met to study the influence of these subjects on the full range of NASA activities and to make recommendations on how these subjects might in the future assist NASA in its mission. The Study Group, chaired by Carl Sagan, was organized by Ewald Heer, JPL, at the request of Stanley Sadin of NASA Headquarters. It included NASA personnel, scientists who have worked on previous NASA missions, and experts on computer science who had little or no prior contact with NASA. The Group devoted about 2500 man-hours to this study, meeting as a full working group or as subcommittees between June 1977 and December 1978.

A number of NASA Centers and facilities were visited during the study. In all cases, vigorous support was offered for accelerated development and use of machine intelligence in NASA systems, with particularly firm backing offered by the Director of the Johnson Space Center, which the Group considered especially significant because of JSC's central role in the development of manned spaceflight.

This report is the complete report of the Study Group. It includes the conclusions and recommendations with supporting documentation. The conclusions represent a group consensus, although occasionally there were dissenting opinions on individual conclusions or recommendations. While the report is critical of past NASA efforts in this field – and most often of the lack of such efforts – the criticisms are intended only as constructive. The problem is government-wide, as the Federal Data Processing Reorganization Project[1] has stressed, and NASA has probably been one of the least recalcitrant Federal agencies in accommodating to this new technology.

The Study Group believes that the effective utilization of existing opportunities in computer science, machine intelligence, and robotics, and their applications to NASA-specific problems will enhance significantly the cost-effectiveness and total information return from future NASA activities.

[1] U.S. Office of Management and Budget, *Federal Data Processing Reorganization Study*. Available from National Technical Information Service, Washington, D.C.

Section II
Machine Intelligence: An Introductory Tutorial[2]

Many human mental activities, such as writing computer programs, doing mathematics, engaging in common sense reasoning, understanding language, and even driving an automobile, are said to demand "intelligence." Over the past few decades, several computer systems have been built that can perform tasks such as these. Specifically, there are computer systems that can diagnose diseases, plan the synthesis of complex organic chemical compounds, solve differential equations in symbolic form, analyze electronic circuits, understand limited amounts of human speech and natural language text, and write small computer programs to meet formal specifications. We might say that such systems possess some degree of artificial intelligence.

Most of the work on building these kinds of systems has taken place in the field called Artificial Intelligence (AI)[3]. This

[2] This section is based on copyrighted material in Nils J. Nilsson's book *Principles of Artificial Intelligence* available from Tioga Publishing Company, Palo Alto, California. The Study Group wishes to thank Nilsson for his permission for use of this material.

[3] In this report the terms Machine Intelligence and Artificial Intelligence are used synonymously.

work has had largely an empirical and engineering orientation. Drawing from a loosely structured but growing body of computational techniques, AI systems are developed, undergo experimentation, and are improved. This process has produced and refined several general AI principles of wide applicability.

AI has also embraced the larger scientific goal of constructing an information-processing theory of intelligence. If such a science of intelligence could be developed, it could guide the design of intelligent machines as well as explicate intelligent behavior as it occurs in humans and other animals. Since the development of such a general theory is still very much a goal rather than an accomplishment of AI, we limit our attention here to those principles that are relevant to the engineering goal of building intelligent machines. Even with this more limited outlook, discussion of AI ideas might well be of interest to cognitive psychologists and others attempting to understand natural intelligence.

In the rest of this section, we will provide a broad overview of several different problem areas in which AI methods and techniques have been applied.

A. Robotics

The problem of controlling the physical actions of a mobile robot might not seem to require much intelligence. Even small children are able to navigate successfully through their environment and to manipulate items, such as light switches, toy blocks, eating utensils, etc. However these same tasks, performed almost unconsciously by humans, when performed by a machine require many of the same abilities used in solving more intellectually demanding problems.

Research on robots or robotics has helped to develop many AI ideas. It has led to several techniques for modeling world states and for describing the process of change from one world state to another. It has led to a better understanding of how to generate plans for action sequences and how to monitor the execution of these plans. Complex robot control problems have forced us to develop methods for planning first at a high level of abstraction, ignoring details, and then at lower and lower levels, where details become important. Nilsson's book contains several examples of robot problem solving which illustrate important ideas in AI.

B. Perception Problems

Attempts have been made to fit computer systems with television inputs to enable them to "see" their surroundings or to fit them with microphone inputs to enable them to "hear" speaking voices. From these experiments, it has been learned that useful processing of complex input data requires "understanding" and that understanding requires a large base of knowledge about the things being perceived.

The process of perception studied in artificial intelligence usually involves a set of operations. A visual scene, say, is encoded by sensors and represented as a matrix of intensity values. These are processed by detectors that search for primitive picture components such as line segments, simple curves, corners, etc. These, in turn, are processed to infer information about the three-dimensional character of the scene in terms of its surfaces and shapes. The ultimate goal is to represent the scene by some appropriate model. This model might consist of a high-level description such as "A hill with a tree on top with cattle grazing."

The point of the whole perception process is to produce a condensed representation to substitute for the unmanageably immense, raw input data. Obviously, the nature and quality of the final representation depend on the goals of the perceiving system. If colors are important, they must be noticed; if spatial relationships and measurements are important, they must be judged accurately. Different systems have different goals, but all must reduce the tremendous amount of sensory data at the input to a manageable and meaningful description.

The main difficulty in perceiving a scene is the enormous number of possible candidate descriptions in which the system might be interested. If it were not for this fact, one could conceivably build a number of detectors to decide the category of a scene. The scene's category could then serve as its description. For example, perhaps a detector could be built that could test a scene to see if it belonged to the category "A hill with a tree on top with cattle grazing." But why should this detector be selected instead of the countless others that might have been used?

The strategy of making hypotheses about various levels of description and then testing these hypotheses seems to offer an approach to this problem. Systems have been constructed that process suitable representations of a scene to develop hypotheses about the components of a description. These hypotheses are then tested by detectors that are specialized to the component descriptions. The outcomes of these tests, in turn, are used to develop better hypotheses, etc.

This hypothesize-and-test paradigm is applied at many levels of the perception process. Several aligned segments suggest a straight line; a line detector can be employed to test it. Adjacent rectangles suggest the faces of a solid prismatic object; an object detector can be employed to test it.

The process of hypothesis formation requires a large amount of knowledge about the expected scenes. Some Artificial Intelligence researchers have suggested that this knowledge be organized in a special structure called a frame or schema. For example, when a robot enters a room through a doorway, it activates a room schema, which loads into a working memory a number of expectations about what might be seen next. Suppose the robot perceives a rectangular form. This form, in the context of a room schema, might suggest a window. The window schema might contain the knowledge that windows typically do not touch the floor. A special detector, applied to the scene, confirms this expectation, thus raising confidence in the window hypothesis. Nilsson's book discusses various fundamental ideas underlying frame-structured representations and inference processes.

C. Combinatorial and Scheduling Problems

An interesting class of problems is concerned with specifying optimal schedules or combinations. Many of these problems can be attacked by the methods of AI. A classical example is the traveling salesman's problem. The problem here is to find a minimum distance tour, starting at one of several cities, visiting each city precisely once, and returning to the starting city. The problem generalizes to one of finding a minimum cost path over the edges of a graph containing n nodes such that the path visits each of the n nodes precisely once.

Many puzzles have this same general character. Another example is the eight-queens problem. The problem is to place eight queens on a standard chessboard in such a way that no queen can capture any of the others; that is, there can be no more than one queen in any row, column, or diagonal. In most problems of this type, the domain of possible combinations or sequences from which to choose an answer is very large. Routine attempts at solving these types of problems soon generate a combinatorial explosion of possibilities that exhaust even the capacities of large computers.

Several of these problems (including the traveling salesman problem) are members of a class that computational theorists call NP-complete. Computational theorists rank the difficulty of various problems on how the worst case for the time taken (or number of steps taken) to solve them by the theoretically best method grows with some measure of the problem size. (For example, the number of cities would be a measure of the size of a traveling salesman problem.) Thus, problem difficulty may grow linearly, polynomially, or exponentially, for example, with problem size.

The time taken by the best methods currently known for solving NP-complete problems grows exponentially with problem size. It is not yet known whether faster methods (involving only polynomial time, say) exist; but it has been proven that if a faster method exists for one of the NP-complete problems, then this method can be converted to similarly faster methods for all the rest of the NP-complete problems. In the meantime, we must make do with exponential-time methods.

AI researchers have worked on methods for solving several types of combinatorial problems. Their efforts have been directed at making the time-versus-problem-size curve grow as slowly as possible, even when it must grow exponentially. Several methods have been developed for delaying and moderating the inevitable combinatorial explosion. Again, knowledge about the problem domain is the key to more efficient solution methods. Many of the methods developed to deal with combinatorial problems are also useful on other, less combinatorially severe problems.

D. Automatic Programming

The task of writing a computer program is also related to other areas of AI. Much of the basis research in automatic programming, theorem proving, and robot problem solving overlaps. In a sense, existing compilers already do "automatic programming." They take in a complete source code specification of what a program is to accomplish; they write an object code program to do it. What we mean here by automatic programming might be described as a "super-compiler" or a program that could take in a very high-level description of what the program is to accomplish and from it produce a program. The high-level description might be a precise statement in a formal language, such as the predicate calculus, or it might be a loose description, say, in English, that would require further dialogue between the system and the user in order to resolve ambiguities.

The task of automatically writing a program to achieve a stated result is closely related to the task of proving that a given program achieves a stated result. The latter is called program verification. Many automatic programming systems produce a verification of the output program as an added benefit.

One of the important contributions of research in automatic programming has been the notion of debugging as a problem-solving strategy. It has been found that it is often much more efficient to produce an errorful solution to a programming or robot control problem cheaply and then

modify it (to make it work correctly), than to insist on a first solution completely free of defects.

E. Expert Consulting Systems

AI methods have also been employed in the development of automatic consulting systems. These systems provide human users with expert conclusions about specialized subject areas. Automatic consulting systems have been built that can diagnose diseases, evaluate potential ore deposits, suggest structures for complex organic chemicals, and even provide advice about how to use other computer systems.

A key problem in the development of expert consulting systems is how to represent and use the knowledge that human experts in these subjects obviously possess and use. This problem is made more difficult by the fact that the expert knowledge in many important fields is often imprecise, uncertain, or anecdotal (though human experts use such knowledge to arrive at useful conclusions).

Many expert consulting systems employ the AI technique of rule-based deduction. In such systems, expert knowledge is represented as a large set of simple rules, and these rules are used to guide the dialogue between the system and the user and to deduce conclusions. Rule-based deduction is one of the major topics in Nilsson's book.

F. Natural Language Processing

When humans communicate with each other using language, they employ, almost effortlessly, extremely complex and still little understood processes. It has been very difficult to develop computer systems capable of generating and "understanding" even fragments of a natural language, such as English. One source of the difficulty is that language has evolved as a communication medium between intelligent beings. Its primary purpose is to transmit a bit of "mental structure" from one brain to another under circumstances in which each brain possesses large, highly similar surrounding mental structures that serve as a common context. Furthermore, part of these similar, contextual mental structures allows each brain to know that the other brain also possesses this common structure and that the other brain can and will perform certain processes using it during its communication "acts." The evolution of language use has apparently exploited the opportunity for each brain to use its considerable computational resources and shared knowledge to generate and understand highly condensed and streamlined messages: A word to the wise from the wise is sufficient. Thus, generating and understanding language is an encoding and decoding problem of fantastic complexity.

A computer system capable of understanding a message in natural language would seem, then, to require (no less than would a human) both the contextual knowledge and the processes for making the inferences (from this contextual knowledge and from the message) assumed by the message generator. Some progress has been made toward computer systems of this sort, for understanding spoken and written fragments of language. Fundamental to the development of such systems are certain AI ideas about structures for representing contextual knowledge and certain techniques for making inferences from that knowledge. Although Nilsson's book does not treat the language-processing problem in detail, it does describe some important methods for knowledge representation and processing that do find application in language-processing systems.

G. Intelligent Retrieval From Databases

Database systems are computer systems that store a large body of facts about some subject in such a way that they can be used to answer user's questions about that subject. To take a specific example, suppose the facts are the personnel records of a large corporation. Example items in such a database might be representations for such facts as "Joe Smith works in the Purchasing Department," "Joe Smith was hired on October 8, 1976," "The Purchasing Department has 17 employees," "John Jones is the manager of the Purchasing Department," etc.

The design of database systems is an active subspecialty of computer science, and many techniques have been developed to enable the efficient representation, storage, and retrieval of large numbers of facts. From our point of view, the subject becomes interesting when we want to retrieve answers that require deductive reasoning with the facts in the database.

There are several problems that confront the designer of such an intelligent information retrieval system. First, there is the immense problem of building a system that can understand queries stated in a natural language like English. Second, even if the language-understanding problem is dodged by specifying some formal, machine-understandable query language, the problem remains of how to deduce answers from stored facts. Third, understanding the query and deducing an answer may require knowledge beyond that explicitly represented in the subject domain database. Common knowledge (typically omitted in the subject domain database) is often required. For example, from the personnel facts mentioned above, an

intelligent system ought to be able to deduce the answer "John Jones" to the query "Who is Joe Smith's boss?" Such a system would have to know somehow that the manager of a department is the boss of the people who work in that department. How common knowledge should be represented and used is one of the system design problems that invites the methods of Artificial Intelligence.

H. Theorem Proving

Finding a proof (or disproof) for a conjectured theorem in mathematics can certainly be regarded as an intellectual task. Not only does it require the ability to make deductions from hypotheses but it also demands intuitive skills such as guessing about which lemmas should be proved first in order to help prove the main theorem. A skilled mathematician uses what he might call judgment (based on a large amount of specialized knowledge) to guess accurately about which previously proven theorems in a subject area will be useful in the present proof and to break his main problem down into subproblems to work on independently. Several automatic theorem proving programs have been developed that possess some of these same skills to a limited degree.

The study of theorem proving has been of significant value in the development of AI methods. The formalization of the deductive process using the language of predicate logic, for example, helps us to understand more clearly some of the components of reasoning. Many informal tasks, including medical diagnosis and information retrieval, can be formalized as theorem-proving problems. For these reasons, theorem proving is an extremely important topic in the study of AI methods.

I. Social Impact[4]

The impact of computers, machine intelligence, and robotics must be examined in the broader context of their impact on society as a whole, rather than the narrower focus based on NASA needs and applications. The impact of information processing technology (and machine intelligence and robotics) on society has been considered in detail by Simon. Here we present the conclusions derived by him. The reader is referred to Simon's book for details of the reasoning and evidence that led to the conclusions presented here.

[4] This subsection is based on material presented in *The New Science of Management Decision*, revised edition, by Herbert A. Simon, Prentice-Hall, Englewood Cliffs, N.J., 1977. The Study Group would like to thank Professor Simon and Prentice-Hall for their kind permission for the use of the material. The reader is referred to Chapters 3 and 5 of the book for detailed discussions that lead to the conclusions presented here.

1. The Dehumanization – Alienation Hypothesis

There has been a great deal of nervousness, and some prophetic gloom, about human work in highly automated organizations. An examination of such empirical evidence, and an analysis of the arguments that have been advanced for a major impact of automation upon the nature of work has led us to a largely negative result.

There is little evidence for the thesis that job satisfaction has declined in recent years, or that the alienation of workers has increased. Hence, such trends, being nonexistent, cannot be attributed to automation, past or prospective. Trends toward lower trust in government and other social institutions flow from quite different causes.

An examination of the actual changes that have taken place in clerical jobs as the result of introducing computers indicates that these changes have been modest in magnitude and mixed in direction. The surest consequence of factory and office automation is that it is shifting the composition of the labor force away from those occupations in which average job satisfaction has been lowest, toward occupations in which it has been higher.

The argument that organizations are becoming more authoritarian and are stifling human creativity flies in the face of long-term trends in our society toward the weakening of authority relations. Moreover, the psychological premises on which the argument rests are suspect. Far more plausible is the thesis that human beings perform best, most creatively, and with greatest comfort in environments that provide them with some immediate amount of structure, including the structure that derives from involvement in authority relations. Just where the golden mean lies is hard to say, but there is no evidence that we are drifting farther from it.

Finally, while we certainly live in a world that is subject to continuing change, there is reason to believe that the changes we are undergoing are psychologically no more stressful, and perhaps even less, stressful, than those that our parents and grandparents experienced. It appears that the human consequences we may expect from factory and office automation are relatively modest in magnitude, that they will come about gradually, and that they will bring us both disadvantages and advantages – with the latter possibly outweighing the former.

2. The Potential for Increased Unemployment

Simon presents evidence that any level of technology and productivity is compatible with any level of employment,

including full employment. He suggests that the problems we face today will not cause us to retreat from high technology – for such a retreat would not be consistent with meeting the needs of the world's population – but that they will bring about a substantial qualitative shift in the nature of our continuing technological progress. For future increases in human productivity, we will look more to the information-processing technologies than to the energy technologies. Because of resource limitations and because of shifting patterns of demand with rising real incomes, a larger fraction of the labor force than at present will be engaged in producing services, and a smaller fraction will be engaged in producing goods. But there is no reason to believe that we will experience satiety of either goods or services at full employment levels.

3. The Impact on Resources and Environment

Technology is knowledge and information-processing technology is knowledge of how to produce and use knowledge more effectively. Modern instruments – those, for example, that allow us to detect trace quantities of contaminants in air, water, and food – inform us about consequences of our actions of which we were previously ignorant. Computers applied to the modeling of our energy and environmental systems trace out for us the indirect effects of actions taken in one part of our society upon other parts. Information-processing technology is causing all of us to take account of the consequences of our actions over spans of time and space that seldom concerned us in the past. It is placing on us – perhaps forcing on us – the responsibilities of protecting future generations as well as our own. In this way, the new technology, the new knowledge, is helping to redefine the requirements of morality in human affairs.

J. Conclusion

In this section we have attempted to provide a broad introductory tutorial to AI. Detailed discussion of the methods and techniques of AI and the wide range of problem domains in which they have been applied is given in various survey articles by Minsky (1963), Newell (1969), Nilsson (1974), and Feigenbaum (1978) all of which appear as Appendixes B to E of this report. Appendix F (Newell, 1970) discusses the relationship between artificial intelligence and cognitive psychology. (The book, *Introduction to Artificial Intelligence* by Patrick H. Winston, also provides an excellent introduction to the field.)

Section III
NASA Needs

NASA is, to a significant degree, an agency devoted to the acquisition, processing, and analysis of information – about the Earth, the solar system, the stars, and the universe. The principal goal of NASA's booster and space vehicle commitment is to acquire such scientific information for the benefit of the human species. As the years have passed and NASA has mustered an impressive array of successful missions, the complexity of each mission has increased as the instrumentation and scientific objectives have become more sophisticated; and the amount of data returned has also increased dramatically. The Mariner 4 mission to Mars in 1965 was considered a striking success when it returned a few million bits of information. The Viking mission to Mars, launched a decade later, acquired almost ten thousand times more information. Comparable advances have been made in Earth resources and meteorological satellites, and across the full range of NASA activities. At the present time, the amount of data made available by NASA missions is larger than scientists can comfortably sift through. This is true, for example, of Landsat and other Earth resources technology satellite missions. A typical information acquisition rate in the 1980s is about 10^{12} bits per day for all NASA systems. In two years, this is roughly the total non-pictorial information content of the Library of Congress. The problem is clearly getting much worse. We have reached a severe limitation in the traditional way of acquiring and analyzing data.

A recent study at JPL estimates that NASA could save 1.5 billion dollars per year by A.D. 2000 through serious implementation of machine intelligence. Given different assumptions, the saving might be several times less or several times more. It is clear, however, that the efficiency of NASA activities in bits of information per dollar and in new data acquisition opportunities would be very high were NASA to utilize the full range of modern computer science in its missions. Because of the enormous current and expected advances in machine intelligence and computer science, it seems possible that NASA could achieve orders-of-magnitude improvement in mission effectiveness at reduced cost by the 1990s.

Modern computer systems, if appropriately adapted, are expected to be fully capable of extracting relevant data either onboard the spacecraft or on the ground in user-compatible format. Thus, the desired output might be a direct graphic display of snow cover, or crop health, or global albedo, or mineral resources, or storm system development, or hydrologic cycle. With machine intelligence and modern computer graphics, an immense amount of data can be analyzed and reduced to present the scientific or technological results directly in a convenient form. This sort of data-winnowing and content analysis is becoming possible, using the developing techniques of machine intelligence. But it is likely to remain unavailable unless considerably more relevant research and systems development is undertaken by NASA.

The cost of ground operations of spacecraft missions and the number of operations per command uplinked from ground to spacecraft are increasing dramatically (Figures 3-1 and 3-2). Further development of automation can, at the same time, dramatically decrease the operations costs of complex missions and dramatically increase the number and kinds of tasks performed, and therefore, the significance of the data returned. Figures 3-3 and 3-4 illustrate schematically how improved automation can produce a significant decline in the cost of mission operations. The projected reallocation of responsibility during mission operations between ground-based humans and spacecraft computer processing is shown in Figure 3-5. There are many simple or repetitive tasks which existing machine intelligence technology is fully capable of dealing with more reliably and less expensively than if human beings were in the loop. This, in turn, frees human experts for more difficult judgmental tasks. In addition, existing and projected advances in robot technology would largely supplant the need for manned missions, with a substantial reduction in cost.

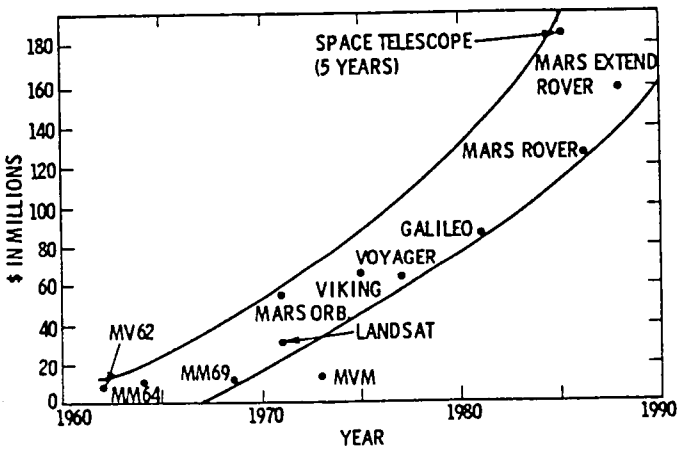


Figure 3-1. Trend of mission ground operations costs. Increasing mission complexity and duration contribute to the ground operation costs.

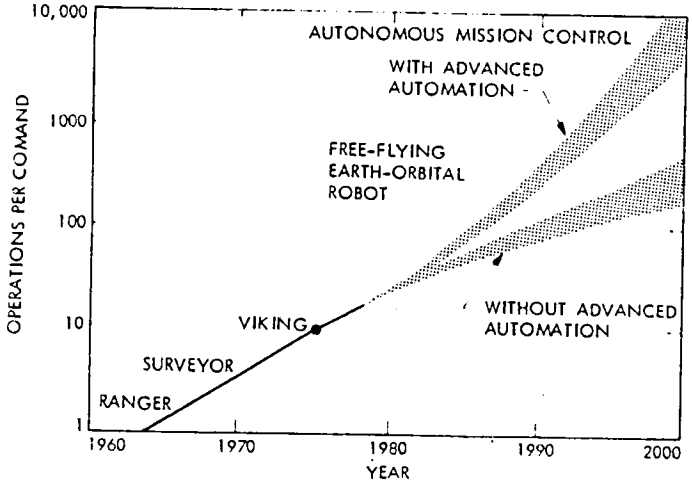


Figure 3-2. Trend of spacecraft automation. As a relative indicator, the level of automation is measured by the different elementary functions the spacecraft can perform in an unpredictable environment between ground commands. A 100-fold improvement through advanced automation is projected by the year 2000.

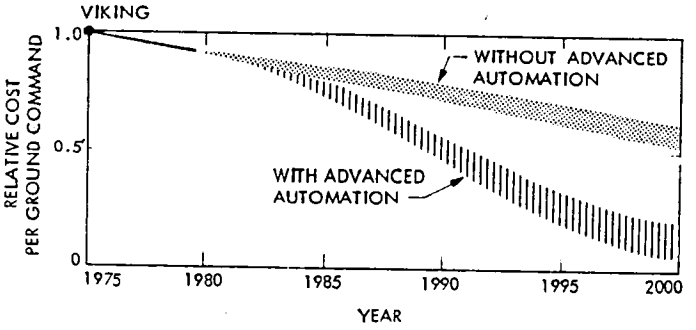


Figure 3-3. Trend of cost to generate ground commands. A four-fold improvement through advanced automation is projected by the year 2000 through (1) performing more ground functions on the spacecraft, and (2) automating the remaining functions on the ground.

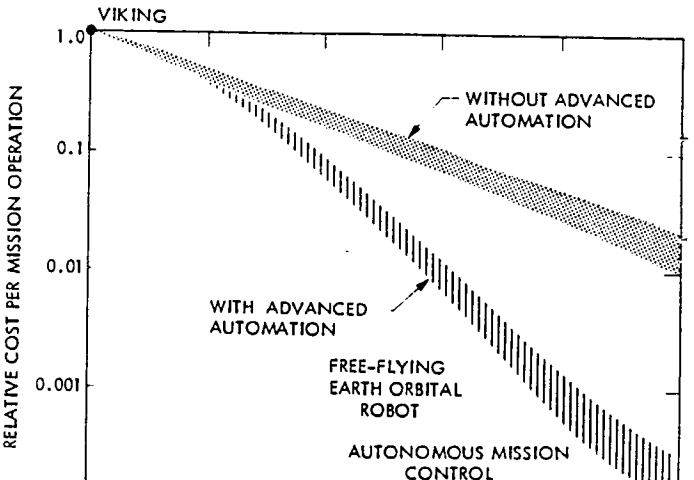


Figure 3-4. Trend of cost per mission operation. A 100- to 1000-fold improvement through advanced automation is projected by the year 2000 for executing a typical mission operation.

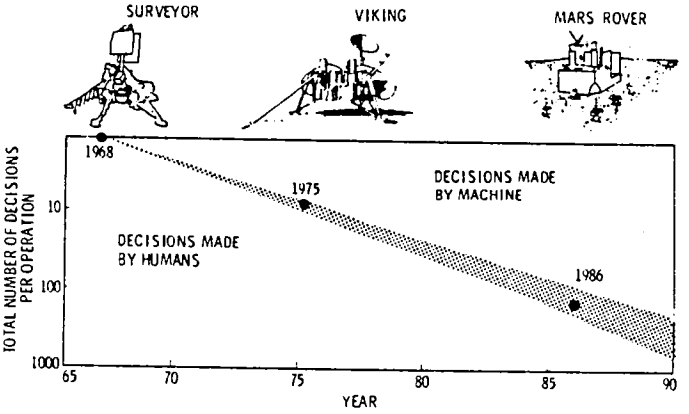


Figure 3-5. Trend of function (decision) allocation between humans and spacecraft. For the same typical operation, the machine takes over an increasing number of elementary functions, leaving high-level decisions to human beings.

Section IV
Applications of Machine Intelligence and Robotics in the Space Program[5]

A. Introduction

The space program is at the threshold of a new era that may be distinguished by a highly capable space transportation system. In the 1980s, the Space Shuttle and its adjuncts will enable increased activities in the scientific exploration of the universe and a broadened approach to global service undertakings in space. The first steps toward utilizing the space environment for industrial and commercial ventures will become possible and can trigger requirements for more advanced space transportation systems in the 1990s. This will enable expanded space industrial activities and, by the end of this century, could lead to Satellite Power Systems for solar energy production, to lunar or asteroidal bases for extracting and processing material resources, and to manned space stations for commercial processing and manufacturing in space. A major objective for NASA is to develop the enabling technology and to reduce the costs for operating such large-scale systems during the next two decades. On examining potential NASA missions in this time frame we expect that machine intelligence and robotics technology will be a vital contributor to the cost-effective implementation and operation of the required systems. In some areas, it will make the system feasible, not only for technological reasons, but also in terms of commercial acceptability and affordability.

During the next two decades, the space program will shift at least some emphasis from exploration to utilization of the space environment. It is expected that this shift will be accompanied by a large increase in requirements for system operations in space and on the ground, calling for general-purpose automation (robotics) and specialized automation. What operations, tasks, and functions must be automated, and to what degree, to accomplish the NASA objectives with the most cost-effective systems?

B. Robots and Automation in NASA Planning

Whereas mechanical power provides physical amplification and computers provide intellectual amplification, telecommunication provides amplification of the space accessible to humans. By means of telecommunication, humans can activate and control systems at remote places. They can perform tasks even as far away as the planets. During the 1960s, this became known as teleoperation. Teleoperators are man-machine systems that augment and extend human sensory, manipulative, and cognitive abilities to remote places. In this context, the term robot can then be applied to the remote system of a teleoperator, if it has at least some degree of autonomous sensing, decision-making, and/or action capability. The concept of teleoperation has profound significance in the space program. Because of the large distances involved, almost all space missions fall within the teleoperator definition; and, because of the resultant communication delay for many missions, the remote system requires autonomous capabilities for effective operation. The savings of operations time for deep space missions can become tremendous, if the remote system is able to accomplish its tasks with minimum ground support. For example, it has been estimated that a Mars roving vehicle would be operative only 4 percent of the time in a so-called move-and-wait mode of operation. With adequate robot technology, it should be operative at least 80 percent of the time.

NASA saw the need to examine the civilian role of the U.S. space program during the last quarter of this century. A series of planning studies and workshops was initiated with the Outlook for Space Study in 1974, which included a comprehensive forecast of space technology for 1980-2000. In a subsequent NASA/OAST Space Theme Workshop, the technology forecasts were applied to three broad mission themes: space exploration, global services, and space industrialization. Based on the derived requirements for cost-effective space mission operations, five new directions were identified for developments in computer systems, machine intelligence and robotics: (1) automated operations aimed at a tenfold reduction in mission support costs; (2) precision pointing and control; (3) efficient data acquisition to permit a tenfold increase in information collection needed for global coverage; (4) real-time data management; and (5) low-cost data distribution to allow a thousand-fold increase in information availability and space-systems effectiveness. The machine intelligence and automation technologies for data acquisition, data processing, information extraction, and decision making emerge here as the major drivers in each area and call for their systematic development. In addition, for certain areas such as automated operations in space, the mechanical technologies directed at

[5]Excerpted from *New Luster for Space Robots and Automation* by Ewald Heer, Astronautics & Aeronautics, Volume 16, No 9, pp 48-60, September 1978.

materials and objects acquisition, handling, and assembly must also be further developed; robots doing construction work in Earth orbit or on the lunar surface will need manipulative and locomotion devices to perform the necessary transport and handling operations.

C. Future Applications

In space applications, robots may take on many forms. None looks like the popular science fiction conception of a mechanical man. Their appearance follows strictly functional lines, satisfying the requirements of the mission objectives to be accomplished. The discussion which follows briefly presents mission categories, mission objectives, and system characteristics pertinent to space robot and automation technology. Estimates of technology development efforts to automate system functions are given in Table 4-1.

1. Space Exploration

Space exploration robots may be exploring space from Earth orbit as orbiting telescopes, or they may be planetary flyby and/or orbiting spacecraft like the Mariner and Pioneer families. They may be stationary landers with or without manipulators like the Surveyor and the Viking spacecraft, or they may be wheeled like the Lunakhod and the proposed Mars rovers. Others may be penetrators, flyers, or balloons, and some may bring science samples back to Earth (Figures 4-1 – 4-3). All can acquire scientific and engineering data

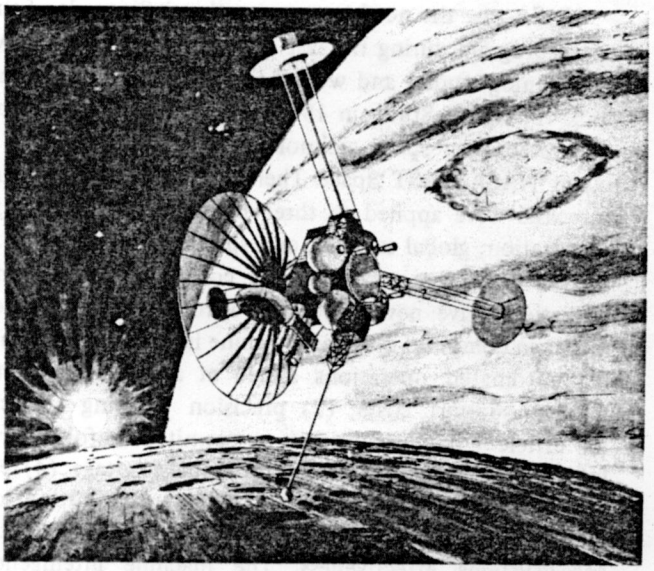


Figure 4-1. Galileo spacecraft navigates between Jupiter and Galilean satellites in rendering. After sending a probe into the jovian atmosphere, the robot spacecraft will perform complex maneuvers at various inclinations with repeated close encounters with the satellites.

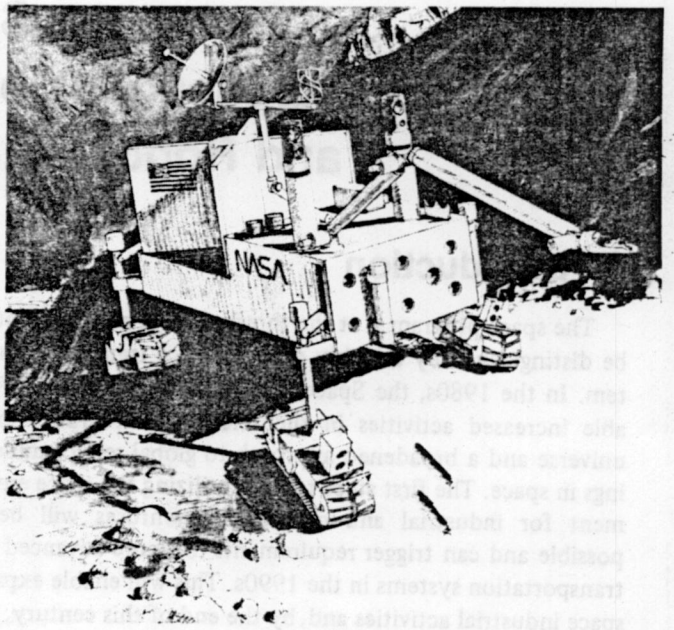


Figure 4-2. Mars surface robot will operate for 2 years and travel about 1000 km performing experiments automatically and sending the scientific information back to Earth.

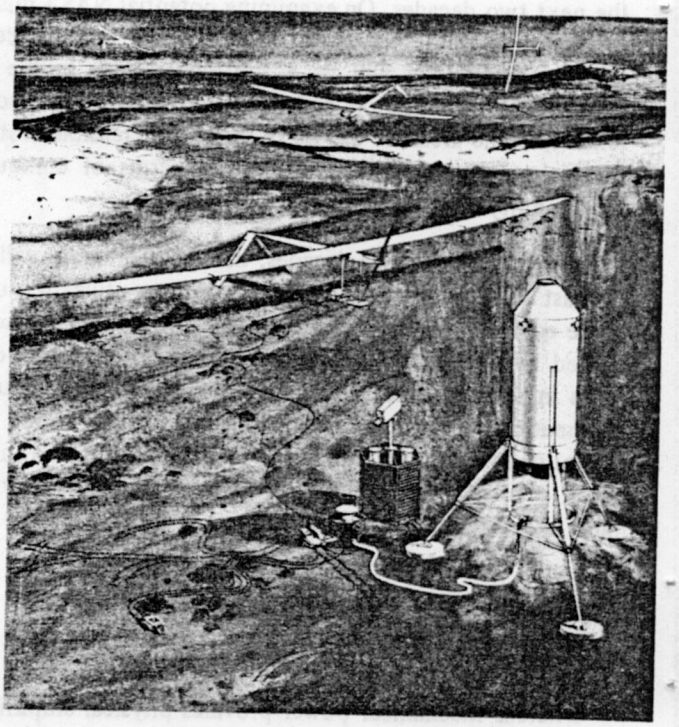


Figure 4-3. Artist's concept of a Mars surface scientific processing and sample return facility. Airplanes transport samples into the vicinity of the processing station. Tethered small rovers then bring the samples to the station for appropriate analysis and return to Earth.

Table 4-1. Estimates of the technology development efforts to automate system functions

ESTIMATES OF TECHNOLOGY DEVELOPMENT EFFORTS TO AUTOMATE SYSTEM FUNCTIONS

GROUND OPERATIONS

Mission Categories	Mission	Year	Data Interpretation	Data Distribution	Archiving	Sequencing	Site Selection	Spacecraft Monitoring	Simulation
Exploration of Space	GALILEO-JUPITER ORBITER PROBE	1982	X	/	X	X	/	X	X
	SOLAR POLAR MISSION	1983	/	/	X	X		X	X
	VENUS ORBITAL IMAGING RADAR	1983	▒	/	▒	X		X	X
	SPACELAB INSTRUMENT PROGRAM	1984	X	X	▒			X	
	X-RAY OBSERVATORY	1986	▒	X	▒			X	
	SATURN ORBITER DUAL PROBE	1987	▒	X	▒	▩	▩	▒	▒
	MARS SAMPLE RETURN	1988	▒	X	▒	▩	▩	▒	▩
	MOBILE LUNAR SURFACE SURVEY[1]	1989	▒	X	▒	▒	▒	▒	▒
	EARTH ORB. SOLAR OBSERVATORY	1992	X	X	X			X	
	ASTROPH. SPACE LABORATORY	1993						X	
	ATMOSPHERIC PHYSICS LAB.	1994						X	
	SPACE-BASED RADIO TELESCOPE	1995	▒	X	▒			▒	
	AUTOMATED PLANETARY STATION	2000	▒	X	▒	▩	▩	▩	▩
Global Services	SEASAT FOLLOW-ON	1982	X	X	▩			X	
	TIROS-O	1984	X	X	X			X	
	SOIL MOISTURE SATELLITE	1985	X	X	▩			X	
	STORMSAT	1985	X	X	X			X	
	GLOBAL COMMUNICATIONS	1987						X	
	GLOBAL CROP FORECASTING	1988	▩	▒	▒			X	
	HIGH RESOLUTION SEA SURVEY	1988	X	X	X			X	
	DISASTER WARNING SATELLITE	1989						X	
	EARTH ENERGY BUDGET MONITOR	1990						X	
	LGE-SCALE ALL-WEATHER SURVEY	1993	▩	▒	▒			X	
	GEOLOGICAL MAPPING	1994						X	
	GLOBAL NAVIGATION	1996						X	
Utilization of Space	SPACE MANUFACTURING MODULE	1985						▒	
	SPACE HEALTH CARE SYSTEM	1986						▒	
	LUNAR PRECURSOR PROCESSOR	1990							
	NUCLEAR WASTE DISPOSAL	1995							
	TELEOPERATOR VEHICLE SYSTEM	1996							
	ROBOT VEHICLE EARTH ORBIT	1997						X	
	LUNAR BASE	1998							
	SPACE STATION	2000							
	SATELLITE POWER SYSTEM	2000						X	
Transportation Systems	ORBITAL TRANSFER VEHICLE	1990						X	
	HIGH ENERGY OTV (PLANET)	1992						X	
	PRIORITY LAUNCH VEHICLE	1994							
	HEAVY-LIFT LAUNCH VEHICLE	1998						X	

ON-BOARD SPACECRAFT OPERATIONS

Mission	Year	Pointing and Controlling	Guidance and Navigation	Rendezvous and Station Keep.	Docking	Landing and Ascent	Surface Traversing	Science Data Collecting	Service Data Collecting	Data Processing	Planning/Deciding	Resource Control	Fault Detection, Repair	Shape Control
GALILEO-JUPITER ORBITER PROBE	1982	/	X					/		X	X	/	X	
SOLAR POLAR MISSION	1983	/	X					/		/	/	/	X	
VENUS ORBITAL IMAGING RADAR	1983	/	X					X		▒	▒	X	X	
SPACELAB INSTRUMENT PROGRAM	1984	X								▒	X			
X-RAY OBSERVATORY	1986	X						X		▒	▒	X	X	
SATURN ORBITER DUAL PROBE	1987	X	▒			■		▩		▩	■	▒	▩	
MARS SAMPLE RETURN	1988	X	X	▩	▩	▩	▩	▩		▩	▩	▩	▩	
MOBILE LUNAR SURFACE SURVEY[1]	1989					X	▩	▩		▩	▩	▒	▩	
EARTH ORB. SOLAR OBSERVATORY	1992	▒						▒		▩	▩	▒	▩	
ASTROPH. SPACE LABORATORY	1993	▒						▒		▩	▩	▒	▩	
ATMOSPHERIC PHYSICS LAB.	1994	▒						▒		▩	▩	▒	▩	
SPACE-BASED RADIO TELESCOPE	1995	▒						▒		■	■	▒	▩	▩
AUTOMATED PLANETARY STATION	2000	▒	▩	▩	▩	■	■	▩		■	■	▒	■	
SEASAT FOLLOW-ON	1982	/							X	▒	▒	/	X	
TIROS-O	1984	/							X	▒	▒	/	X	
SOIL MOISTURE SATELLITE	1985	/							X	▒	▒	/	X	
STORMSAT	1985	/							X	▒	▒	/	X	
GLOBAL COMMUNICATIONS	1987	▒										/	X	▩
GLOBAL CROP FORECASTING	1988	X							▒	▩	▩	/	X	
HIGH RESOLUTION SEA SURVEY	1988	▒							X	▒	▒	/	X	
DISASTER WARNING SATELLITE	1989	▒							▒	■	■	/	X	
EARTH ENERGY BUDGET MONITOR	1990	▒							X	▒	▒	/	X	
LGE-SCALE ALL-WEATHER SURVEY	1993	▒							▒	▩	▩	/	X	
GEOLOGICAL MAPPING	1994	▒							▒	▩	▩	/	X	▩
GLOBAL NAVIGATION	1996	▒										/	X	▩
SPACE MANUFACTURING MODULE	1985			▒	▒					▩	▩	▒	▒	
SPACE HEALTH CARE SYSTEM	1986									▒		▒	▩	
LUNAR PRECURSOR PROCESSOR	1990					▩				▒		▒	▩	
NUCLEAR WASTE DISPOSAL	1995			▒	▒									
TELEOPERATOR VEHICLE SYSTEM	1996	▒	▒	▒	▒							X		
ROBOT VEHICLE EARTH ORBIT	1997	▒	▒	▒	▒					▒	■	▒	▒	
LUNAR BASE	1998									▩	▩	▒	▩	
SPACE STATION	2000	▒								▩	▩	▒	▩	
SATELLITE POWER SYSTEM	2000	▩										▒	X	▩
ORBITAL TRANSFER VEHICLE	1990	▒	▒	▩	▩					▩	■	▒	▩	
HIGH ENERGY OTV (PLANET)	1992	▒	▒							▩	▩	▒	▩	
PRIORITY LAUNCH VEHICLE	1994			▩	▩	▩				▒	▒	▒	▩	
HEAVY-LIFT LAUNCH VEHICLE	1998	▒	▒	▩	▩	▩				▩	▩	▒	▩	

IN-SPACE HANDLING[2]

Mission	Year	Deployment	Transfer	Assembly and Joining	Inspection	Servicing	Maintenance	Repair	Monitoring	Retrieving	Rescue	Processing	Manufacturing	Lunar Mining
MOBILE LUNAR SURFACE SURVEY[1]	1989	▩	▩	▩	▩	▩	▩	▩	▩	▩	▩			▩
SPACE MANUFACTURING MODULE	1985	▒	▒		▒	▒	▒	▒	▒			▩	▩	
SPACE HEALTH CARE SYSTEM	1986		▒		▒	▒	▒	▒	▒					
LUNAR PRECURSOR PROCESSOR	1990		▒						▒			▩		
NUCLEAR WASTE DISPOSAL	1995		▒											
TELEOPERATOR VEHICLE SYSTEM	1996	▒	▒	▒	▒	▒	▒	▒	▒	▒	▒			
ROBOT VEHICLE EARTH ORBIT	1997	▩	▩	▩	▩	▩	▩	▩	▩	▩	▩			
LUNAR BASE	1998								▩			▩	▩	▩
SPACE STATION	2000								▩			▩	▩	
SATELLITE POWER SYSTEM	2000												▩	
ORBITAL TRANSFER VEHICLE	1990	▩	▩	▩	▩	▩	▩	▩	▩	▩	▩			
HIGH ENERGY OTV (PLANET)	1992		▩	▩	▩			▩						

KEY: THE AUTOMATION OF THE IDENTIFIED SYSTEM FUNCTIONS REQUIRES:

/ INTEGRATION OF EXISTING TECHNOLOGY

X MODERATE ADDITIONAL DEVELOPMENTS

▒ EXTENSIVE TECHNOLOGY DEVELOPMENTS

▩ MAJOR TECHNOLOGY DEVELOPMENTS

■ MAJOR TECHNOLOGY DEVELOPMENTS WITH UNCERTAIN OUTCOME.

NOTE: EACH ENTRY REPRESENTS THE RELATIVE COLLECTIVE LEVEL OF EFFORT TO ACCOMPLISH THE FUNCTION FOR THE MISSIONS AS DESCRIBED IN THE NASA/OAST SPACE SYSTEMS TECHNOLOGY MODEL, 22 MARCH 1978.

1) THE LUNAR ROVERS OF THIS PROGRAM WILL BE DEVELOPED WITH IN-SPACE HANDLING CAPABILITIES AND WILL SUPPORT THE LUNAR PRECURSOR PROCESSOR (1990) AND THE LUNAR BASE (1998).

2) HANDLING FUNCTIONS ARE GENERALLY ASSOCIATED WITH MOBILITY UNITS, MANIPULATIVE DEVICES OR TOOLS REQUIRING CONTROL OF ACTUATORS

using their sensors, process the data with their computers, plan and make decisions, and send some of the data back to Earth. Some robots are, in addition, able to propel themselves safely to different places and to use actuators, manipulators, and tools to acquire samples, prepare them, experiment *in situ* with them, or bring them back to Earth.

Exploratory robots are required to send back most of the collected scientific data, unless they become repetitive. The unknown space environment accessible to the sensors is translated into a different, still uninterpreted environment, in the form of computer data banks on Earth. These data banks are then accessible for scientific investigation long after the space mission is over.

Projections into the future lead one to speculate on the possibility of highly autonomous exploratory robots in space. Such exploratory robots would communicate to Earth only when contacted or when a significant event occurs and requires immediate attention on Earth. Otherwise, they would collect the data, make appropriate decisions, archive them, and store them onboard. The robots would serve as a data bank, and their computers would be remotely operated by accessing and programming them from Earth whenever the communication link to the robot spacecraft is open. Scientists would be able to interact with the robot by remote terminal. Indeed, the concept of distributed computer systems, presently under investigation in many places, could provide to each instrument its own microcomputer, and scientists could communicate with their respective instruments. They could perform special data processing onboard and request the data to be communicated to them in the form desired. Alternatively, they could retrieve particular segments of raw data and perform the required manipulations in their own facilities on Earth.

Prime elements in this link between scientists and distant exploratory robots would be large antenna relay stations in geosynchronous orbit. These stations would also provide data handling and archiving services, especially for inaccessible exploratory robots, e.g., those leaving the solar system.

2. Global Services

Global service robots orbit the Earth. They differ from exploratory robots primarily in the intended application of the collected data. They collect data for public service use on soil conditions, sea states, global crop conditions, weather, geology, disasters, etc. These robots generally acquire and process an immense amount of data. However, only a fraction of the data is of interest to the ultimate user. At the same time, the user often likes to have the information shortly after it has been obtained by the spacecraft. For instance, the value of weather information is short-lived except for possible historical reasons. The value of information of disasters such as forest fires is of comparably short duration. The demand for high-volume onboard data processing and pertinent automated information extraction is therefore great.

The usual purpose of global service robots is to collect time-dependent data in the Earth's environment, whose static properties are well-known. The data are used to determine specific patterns or classes of characteristics and translate these into useful information. For instance, for Landsat and Seasat (Figure 4-4), the data are currently sent to the ground, where they are processed, reduced, annotated, analyzed, and distributed to the user. This process requires up to 3 months for a fully processed satellite image and costs several thousand dollars. The image must then be interpreted by the receiver; i.e., the information must still be extracted by the user.

Figure 4-4. Seasat. The oceanographic satellite's high-data-rate Synthetic Aperture Radar imaging device has provided data on ocean waves, coastal regions, and sea ice.

Present developments in artificial intelligence, machine intelligence, and robotics suggest that, in the future, the ground-based data processing and information extraction functions will be performed onboard the robot spacecraft. Only the useful information would be sent to the ground and distributed to the users, while most of the collected data could be discarded immediately. This would require the robot to be able to decide what data must be retained and how they were to be processed to provide the user with the desired information. For instance, the robot could have a large number of pattern classification templates stored in its memory or introduced by a user with a particular purpose in mind. These templates would represent the characteristics of objects and/or features of interest. The computer would compare the scanned patterns with those stored in its memory. As soon as something of interest appeared, it would examine it with higher resolution, comparing it to a progressively narrower class of templates until recognition had been established to a sufficient degree of confidence. The robot would then contact the appropriate ground station and report its findings and, if required, provide the user with an annotated printout or image. The user would be able to interact with the robot, indeed with his particular instrument, by remote terminal much the same as with a central computer and, depending on intermediate results, modify subsequent processing.

For space exploration and global services, the ground-based mission operations can become extremely complex. A recent example of a planetary exploration mission, and perhaps the most complex to date, is Viking. At times there were several hundred people involved in science data analysis, mission planning, spacecraft monitoring, command sequence generation, data archiving, data distribution, and simulation. Although for earlier space missions sequencing had been determined in advance, on Viking this was done adaptively during the mission. The operational system was designed so that major changes in the mission needed to be defined about 16 days before the spacecraft activity. Minor changes could be made as late as 12 hours before sending a command. The turnaround time of about 16 days and the number of people involved contributes, of course, to sharply increased operational costs. The Viking operations costs (Figure 3-1) are for a 3-month mission. The planned Mars surface rover mission is expected to last 2 years, covering many new sites on the Martian surface. Considering that this mission would be more complex and eight times as long, ground operations would have to be at least ten times as efficient to stay within, or close to, the same relative costs as for Viking.

During the Viking mission, about 75,000 reels of image data tapes were collected and stored in many separate locations. The images are now identifiable only by the time when and the location where they were taken. No indication regarding image information content is provided, and the user will have to scan catalogs of pictures to find what he or she wants. For such reasons, it is expected that most of the data will not be used again.

The ground operations for Earth orbital missions suffer from problems similar to those of planetary missions. The overall data stream is usually much higher for Earth orbital missions, images are still very costly, and they take up to several months to reach the user.

These considerations strongly suggest that technology must be developed so that most ground operation activities can be performed as close as possible to the sensors where the data is collected, namely by the robot in space. However, examining the various ground operations in detail, we conclude that most of those that must remain on the ground could also be automated with advanced machine intelligence techniques. The expected benefits derived from this would be a cost reduction for ground operations of at least an order of magnitude and up to three orders of magnitude for user-ready image information.

3. Utilization of Space Systems

Space industrialization requires a broader spectrum of robotics and automation capabilities than those identified for space exploration and global services. The multitude of systems and widely varying activities envisioned in space until the end of this century will require the development of space robot and automation technology on a broad scale. It is here that robot and automation technology will have its greatest economic impact. The systems under consideration range from large antennas and processing and manufacturing stations in Earth orbit to lunar bases, to manned space stations, to satellite power systems of up to 100 km^2. These systems are not matched in size by anything on Earth. Their construction and subsequent maintenance will require technologies not yet in use for similar operations on Earth.

Space processing requires a sophisticated technology. First it must be developed and perfected, and then it must be transferred into the commercial arena. Basic types of processes currently envisioned include solidification of melts without convection or sedimentation, processing of molten samples without containers, diffusion in liquids and vapors, and electrophoretic separation of biological substances. It is expected that specialized automated instrumentation will be developed for remote control once the particulars of these processes are worked out and the pressure of commercial requirements becomes noticeable.

Large-area systems such as large space antennas, satellite power systems, and space stations require large-scale and complex construction facilities in space (Figures 4-5 and 4-6). Relatively small systems, up to 100 m in extent, may be deployable and can be transported into orbit with one Shuttle load. For intermediate systems of several hundred meters in extent, it becomes practical to shuttle the structural elements into space and assemble them on site (Figure 4-7).

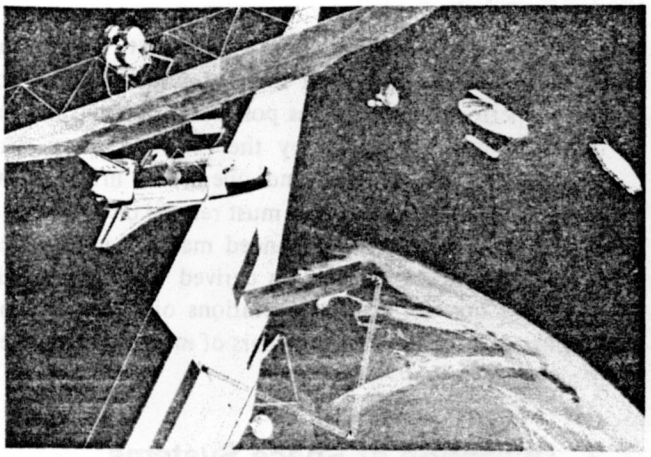


Figure 4-5. Large space systems require robot and automation technology for fabrication, assembly, and construction in space.

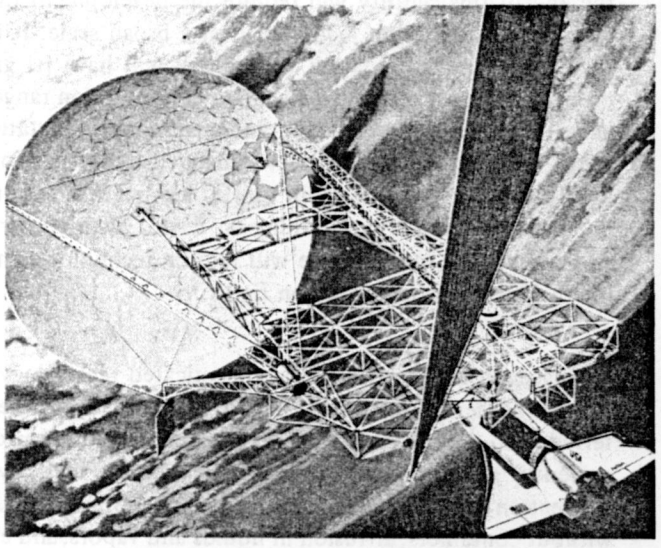


Figure 4-6. Large space antennas are erected with the help of a space-based construction platform. The Shuttle brings the structural elements to the platform, where automatic manipulator modules under remote control perform the assembly.

Very large systems require heavy-lift launch vehicles which will bring bulk material to a construction platform (Figure 4-8), where the structural components are manufactured using specialized automated machines.

The structural elements can be handled by teleoperated or self-actuating cranes and manipulators which bring the components into place and join them (Figure 4-9). Free-flying robots will transport the structural entities between the Shuttle or the fabrication site and their final destination and connect them. These operations require a sophisticated general-purpose

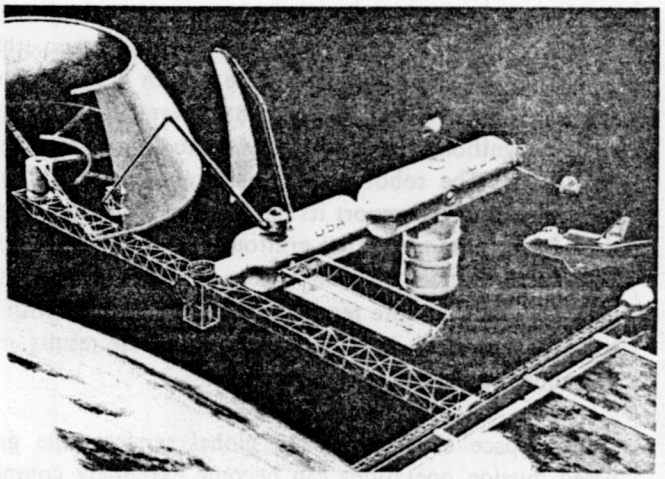


Figure 4-7. Construction of a space station. Bulk material is brought by the Shuttle. Structural elements are fabricated at the construction facility and then assembled by remotely controlled manipulators.

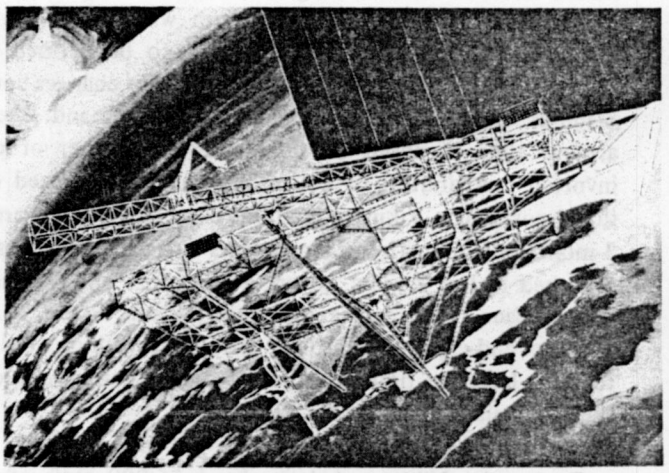


Figure 4-8. Complex construction facility in space with automatic beam builders, cranes, manipulators, etc., is served by the Shuttle.

handling capability. In addition to transporting structural elements, the robot must have manipulators to handle them, and work with them and on them. Large structural subsystems must be moved from place to place and attached to each other. This usually requires rendezvous, stationkeeping, and docking operations at several points simultaneously and with high precision – a problem area still not investigated for zero gravity. Automated "smart" tools would also be required by astronauts to perform specialized local tasks.

These robot systems could be controlled remotely as teleoperator devices, or they could be under *supervisory control* with intermittent human operator involvement. Astronauts in space or human operators on Earth will need the tools to accomplish the envisioned programs. The technology for in-space assembly and construction will provide the foundation for the development of these space-age tools.

After the system has been constructed, its subsequent operation will require service functions that should be performed by free-flying robots or by robots attached to the structure. The functions which such a robot should be able to perform include calibration, checkout, data retrieval, resupply, maintenance, repair, replacement of parts, cargo and crew transfer, and recovery of spacecraft.

During and after construction, there should be a robot on standby for rescue operations. An astronaut drifting into space could be brought back by a free-flying robot. Such devices could also be on stand-by alert on the ground. The delivery systems for these rescue robots need not be man-rated. They can deliver expendable life support systems or encapsulate the astronaut in a life support environment for return to a shuttle, space station, or Earth. They could also perform first-aid functions.

Another phase of space industrialization calls for a lunar or asteroidal base. After a surface site survey with robot (rover) vehicles, an automated precursor processor system could be placed on the Moon or the asteroid. This system would collect solar energy and use it in experimental, automated physical/chemical processes for extracting volatiles, oxygen, metals, and glass from lunar soil delivered by automated rovers (Figure 4-10). The products would be stored, slowly building up stockpiles in preparation for construction. The lunar or asteroidal base would be built using automated equipment and robots as in Earth orbit. After construction, general-purpose robot devices would be necessary for maintenance and repair operations. In addition, the base would use industrial automation (qualified for operation in space) or a sort generally similar to those employed on Earth for similar tasks.

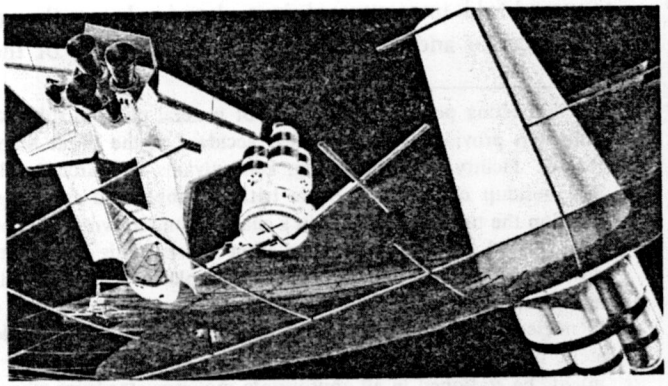


Figure 4-9. Space construction of large antenna systems with automated tools, teleoperated manipulators, and free-flying robots.

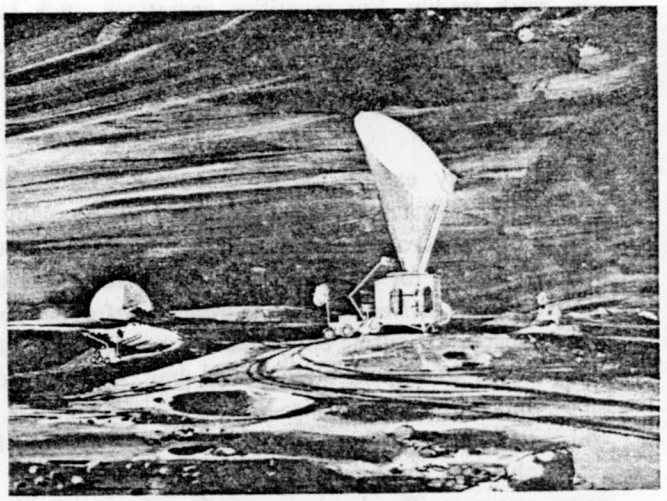


Figure 4-10. Automated material processors on the lunar surface are serviced by robot vehicles with raw lunar soil.

Section V
Technological Opportunities

A. Trends in Technology

Machine intelligence and robotics are not only relevant but essential to the entire range of future NASA activities. Content analysis of Earth orbital and planetary spacecraft results is merely one application. Other applications exist: in mission operations, in spacecraft crisis management, in large constructions in Earth orbit or on the Moon, and in mining in the lunar or asteroidal environments. These last applications are probably at least a decade into the future, but some essential preparations for them would seem prudent. These preparations might include the development of teleoperators, manipulative devices which are connected via a radio feedback loop with a human being, so that, for example, when the human on the Earth stretches out his hand, the mechanical hand of the teleoperator in Earth orbit extends likewise; or when the human turns his head to the left, the teleoperator's cameras turn to the left so that the human controller can see the corresponding field of view. Where the light travel times are on the order of a tenth of a second or less, the teleoperator mode can work readily. For repetitive operations, such as girder construction and quality control in large space structures, automation and machine intelligence will play a major role in any efficient and cost-effective design.

In planetary exploration in the outer solar system, the light-travel times range from tens of minutes to many hours. As a result, it is often useless for a spacecraft in trouble to radio the Earth for instructions. In many cases, the instructions will have arrived too late to avoid catastrophe. Thus, the Viking spacecraft during entry had to be able to monitor and adjust angle of attack, atmospheric drag, parachute deployment, and retro-rocket firing. Roving vehicles on Mars, Titan, and the Galilean satellites of Jupiter will have to know how to avoid obstacles during terrain traverses and how not to fall down crevasses. The development of modern scientific spacecraft necessarily involves pushing back the frontiers of machine intelligence.

In our opinion, machine intelligence and robotics is one of the few areas where spinoff justifications for NASA activities are valid. In most such arguments, socially useful applications, such as cardiac pacemakers, are used to justify very large NASA expenditures directed toward quite different objectives. But it is easy to see that the direct development of the application, in this case the pacemaker, could have been accomplished at a tiny fraction of the cost of the activity which it is used to justify – the Apollo program, say. However, because there is so little development in machine intelligence and robotics elsewhere in the government (or in the private sector), spinoff arguments for NASA involvement in such activities seem to have some substantial validity. In the long term, practical terrestrial applications might include undersea mineral prospecting and mining, conventional mining (of coal, for example), automated assembly of devices, microsurgery and robotics prosthetic devices, the safe operation of nuclear power plants[6] or other industries which have side effects potentially dangerous for human health, and household robots. A further discussion of future NASA applications of machine intelligence and robotics, and possible spinoff of these activities, is given in the supporting documentation.

With the development of integrated circuits, microprocessors, and silicon chip technology, the capabilities of computers have been growing at an astonishing rate. Figures 5-1 through 5-4 provide estimates of recent past and projected future developments. By such criteria as memory storage, power efficiency, size and cost, the figures of merit of computer systems have been doubling approximately every year. This implies a thousand-fold improvement in a decade. In another decade the processor and memory (four million words) of the IBM

[6] An interesting possible application of general purpose robotics technology is provided by the nuclear accident at the Three Mile Island reactor facility near Harrisburg, Pennsylvania in March/April 1979. The buildup of a high pressure tritium bubble had as one possible solution the turning of a valve in a chamber under two meters of water impregnated with very high radiation fluxes. This is an extremely difficult environment for humans, but a plausible one for advanced multipurpose robots. The stationing of such robots as safety devices in nuclear power plants is one conceivable objective of the development of robotics technology. Generally, such multipurpose robots might be stationed in all appropriate industrial facilities where significant hazards to employee or public health or to the facility itself exists.

Shortly after the Three Mile Island reactor accident the operating company began recruiting "jumpers," individuals of short stature willing, for comparatively high wages, to subject themselves to high radiation doses thought inappropriate for permanent reactor technicians (*New York Times*, July 16, 1979, page 1). The functions are often no more difficult than turning a bolt, but in a radiation environment of tens of rems per hour. There would appear to be strong humanitarian reasons for employing small multipurpose self-propelled robots for this function, as well as to redesign nuclear power plants to make much fuller use of the capabilities of machine intelligence. The competent use of machine intelligence and robotics is an important component of all recently proposed additional energy sources – for example, mining and processing shale and coal.

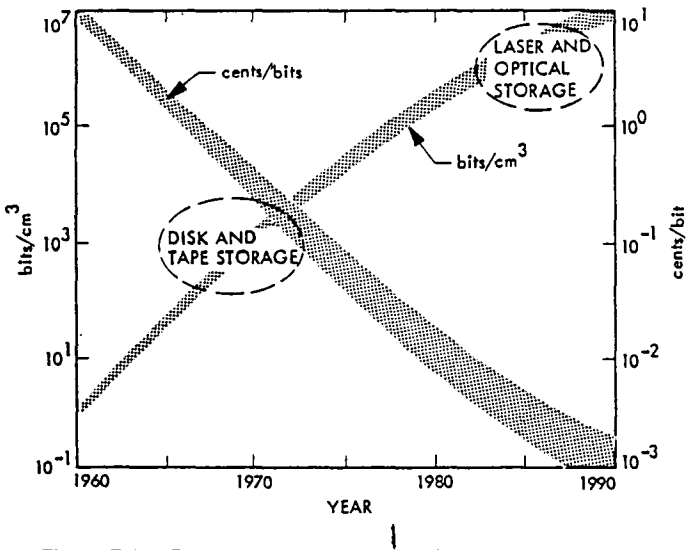


Figure 5-1. Data storage technology. The storage capacity is doubling every 1-1/2 years, whereas the cost of random access memory is halving every 2-1/2 years. In 1960, the equivalent of 1 m^3 stored a 15-page pamphlet; in 1980, the same space will accommodate a 2000-book library and in 1990, the entire Library of Congress.

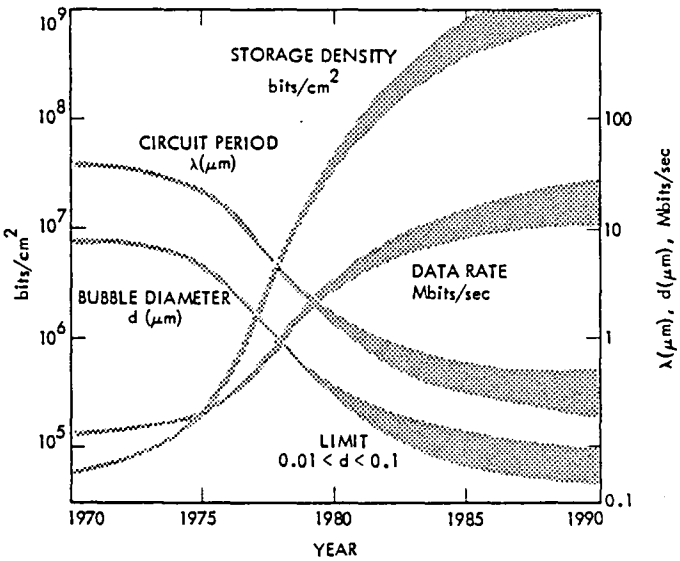


Figure 5-3. Bubble memory technology. About 4 x 10^8 bits/cm^2 would be reached in 1985. This implies a bubble diameter of 10^{-5} cm, which is ten times greater than the theoretical limit. (Adapted from A.H. Bobeck, Bell Laboratory, ELECTRO '77, N. Y.).

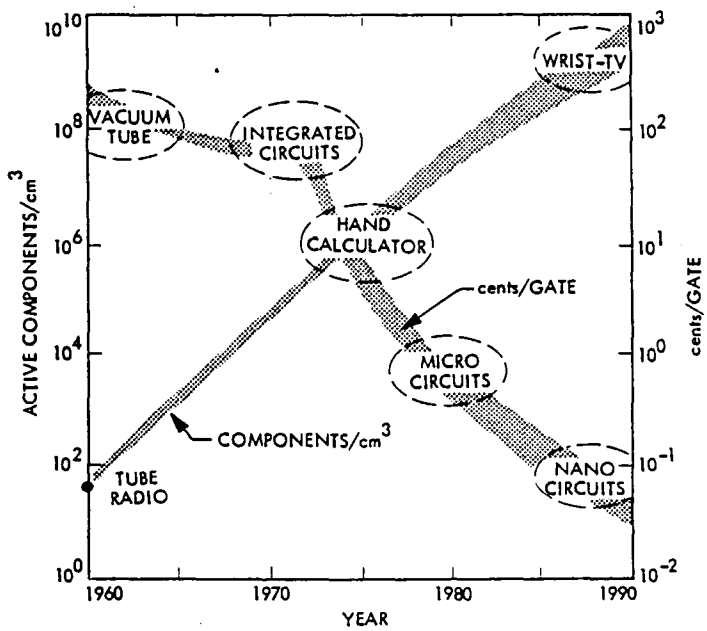


Figure 5-2. Active devices technology. The number of active components per cubic centimeter is doubling every 1-1/8 years, whereas the average cost per logic gate is halving every 2-1/2 years.

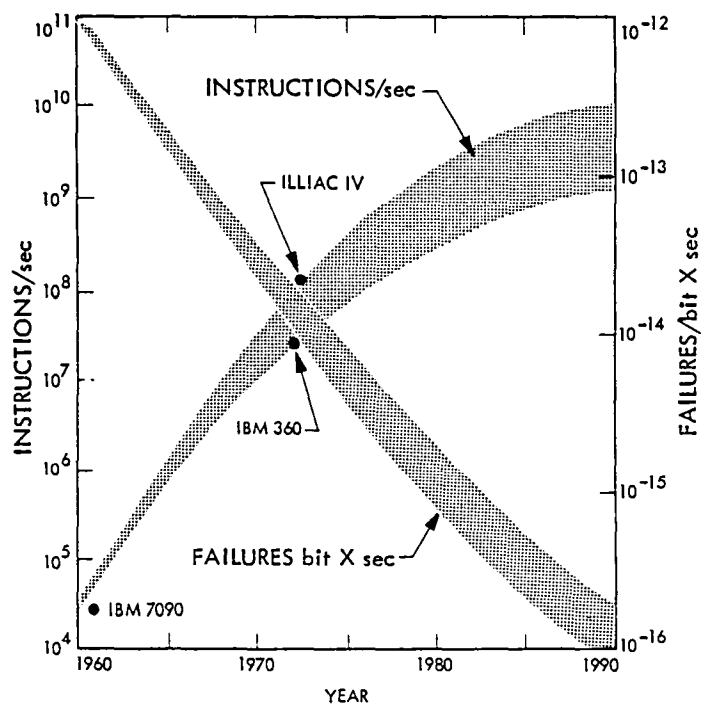


Figure 5-4. Computer systems technology. The average increase of computer speed is doubling every 1-1/2 years, whereas the failure rate is halving every 2-3/4 years.

370/168 will probably be houseable in a cube about five centimeters on a side (although computer architecture different from that of the IBM 370/168 will probably be considered desirable). It is difficult to think of another area of recent technology which has undergone so many spectacular improvements in so short a period of time.

This steep rate of change in computer technology is one major factor in the obsolescence of NASA computer systems. New systems are being developed so fast that project scientists and engineers, mission directors, and other NASA officials have difficulty discovering what the latest advances are, much less incorporating them into spacecraft-mission or ground-operations design.

Another problem is the competition between short-term and long-term objectives in the light of the NASA budget cycle. Major funding is given for specific missions. There is a high premium on the success of individual missions. The safest course always seems to be to use a computer system which has already been tested successfully in some previous mission. But most missions have five- to ten-year lead times. The net result is that the same obsolete systems may be flown for a decade or more. This trend can be seen in areas other than computer technology, as, for example, in the NASA reliance in lunar and planetary exploration for 15 years on vidicon technology, well into a period when commercial manufacturers were no longer producing the vidicon systems and NASA was relying on previously stockpiled devices. This has been the case since 1962. Only with the Galileo mission, in 1984, will more advanced and photometrically accurate charged-coupled device systems be employed. The problem is much more severe when it applies to a field undergoing such dramatic advances as computer technology. The management dynamics can be understood, but it is nevertheless distressing to discover that an agency as dependent on high technology as NASA, an organization identified in the public eye with effective use of computer technology, has been so sluggish in adopting advances made more than a decade earlier, and even slower in promoting or encouraging new advances in robotics and machine intelligence.

The general technological practice of adopting for long periods of time the first system which works at all rather than developing the optimal, most cost-effective system has been amply documented.[7] This phenomenon is by no means restricted to NASA. The need to handle radioactive substances led many years ago to the development of rudimentary teleoperators. At first progress was rapid, with force reflecting, two-fingered models appearing in the early 1950s. But this development all but stopped when progress was sufficient to make the handling of radioactive materials possible – rather than easy, or economical, or completely safe. This occurred in part because the nuclear industry, like NASA, became mission-oriented at this time. Since then, the development of computer-controlled manipulators has proceeded slowly on relatively sparse funding, and there has been little drive to understand in a general and scientific way the nature of manipulation. Major advances seem similarly stalled and likewise entirely feasible in such areas as locomotion research, automated assembly, self-programming, obstacle avoidance during planetary landfall, and the development of spacecraft crisis analysis systems.

[7] Simon, Herbert A., *The New Science of Management Decision*, revised edition, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1977.

B. Relevant Technologies

The principal activity of the Study Group during its existence was to identify machine intelligence and robotics technologies that are highly relevant to NASA and the success of its future programs. Each Study Group workshop had one or more of these topics as the foci of interest. Appendix A gives a complete list of topics covered at each of the workshops. In this section we provide a summary of the discussions of the topics considered by the Study Group.

1. Robotics Technology

Robotics and machine intelligence have in the past played surprisingly small roles in NASA space programs and research and development. Yet these areas will become increasingly more important as the emphasis shifts from exploration missions to missions involving space utilization and industrialization and the fabrication and assembly of space structures. The high cost of placing people in space suggests that the use of robots might be the method of choice long before robotics become practical on Earth.

The uses of robotics can be broadly grouped into manipulators and intelligent planetary explorers. There already exist automatic vision and manipulation techniques that could be developed into practical systems for automatic inspection and assembly of components. Parts could be marked to allow simple visual tracking programs to roughly position them, while force-sensing manipulators could mate the components. Where large structures, designed from standard sets of component parts and assembled in regular patterns are concerned, manipulators could perform reliable, accurate, repetitive operations which would be difficult for a human, in space, to do. Intelligent robot explorers will become imperative, if sophisticated large-scale interplanetary exploration is to become a reality. The round-trip communication delay time

which ranges from a minimum of nine minutes to a maximum of forty minutes for Mars, and the limited "windows" during which information can be transmitted, precludes direct control from Earth. Thus the human role must be that of a supervisor and periodic program-updater of the computer. A robot Mars explorer must be equipped with sensors and appropriate computing capability which maximizes both efficient mobility and intelligent data gathering.

The Study Group recommends that NASA take an active role in developing the necessary robotics technology, including rovers and manipulators, rather than expecting this technology to be transfered from other sources.

2. Smart Sensor Technology

There are several additional areas within NASA applications and mission programs which would benefit from advances in machine visual perception. These areas include remote sensing and crop survey, cartography and meteorology, teleoperators, and intelligent robot explorers.

Current crop census systems do not seem to meet the expectations of lowered cost and increased repeatability from automated classification. It also appears that the 80-meter resolution per pixel of LANDSAT imagery is insufficient for current structural pattern recognition and scene analysis techniques. What is needed is an emphasis on sensors whose resolution is between 2 meters and 2 centimeters per pixel. Coarse sampling (2 meters) would separate field boundaries, while finer resolution (2 centimeters) could be used to perform structural analysis on limited parts of the fields.

Much work is yet to be done in computer stereo vision. Such systems will find applications in the automation of cartographic processes. While research into stereo vision have produced systems which work in a research environment, support is needed for newer high performance systems. Teleoperators and manipulators for fabrication and assembly of materials in space will require a vision system containing smart sensors which provide stereo presentations and the ability to generate multiple views. The quality of visual components in a teleoperator system will determine its utility as much as its mechanical sophistication. Intelligent robot explorers will rely on smart sensor visual systems in order to navigate and recognize interesting features to sample. Laser ranging devices offer minimal navigational aid due to their limited range capability. Stereo vision systems based on motion parallax offer superior capabilities by navigating with respect to distant landmarks. It would thus be possible to avoid difficult terrain and to return to locations of interest.

The Study Group recommends that NASA expand and diversify its image processing research to include knowledge guided interpretation systems and initiate development of LSI-based smart sensors capable of both signal-based and symbolic interpretation.

3. Mission Operations Technology

It appears that significant cost-effective performance can also be realized by the application of machine intelligence techniques to mission planning and sequencing operations. These operations tend to be time-critical during space missions and require many repetitive and routine decision-making roles currently performed by human operators. Mission planning and control facilities dealing with data collection, experimentation scheduling, and monitoring should be automated to a much larger degree. Various missions may share many common requirements which could be served by a software facility providing for mission-independent aspects of data collection and allowing embedding of mission-specific, task-oriented software.

The Study Group recommends that NASA begin the development of a reusable, modular intelligent mission control center with the goal of increasing the mechanization and standardization of sequencing, data handling and delivery, and related protocols.

4. Spacecraft Computer Technology

Digital computers have been playing an ever increasing role in NASA space missions as the need to control and coordinate sophisticated sensors and effectors grows. They are destined to play a dominant role in future space missions. There are several issues, to which NASA should address itself, which bear on the ability of current space-qualified computers to support robotic devices requiring large central processors and memory.

Specifically, fault tolerant designs, large scale integrated circuits, and computer architectures should receive attention by NASA. Fault tolerance implies that expected computer system behavior should continue after faults have occurred. Fault tolerance is essential to space missions since it is impossible to adequately test each component of the total system. Techniques for building reliable systems should include the ability to isolate the effect of a fault to a single module and to detect the fault so automatic recovery algorithms can be invoked to "repair" the fault.

LSI technology holds the promise of more powerful, sophisticated computers with smaller power and weight requirements. However, since technology is rapidly advancing,

the effective use of LSI systems may be severely blunted by the time requirements of space qualification. NASA must avoid committing to architectures prematurely. The adoption of a family of space-qualified computers would allow software to be developed and hardware decisions to be deferred allowing for more cost-effective and powerful technologies. There are many architectural alternatives for space computers: distributed, centralized, and network implementations. A distributed processor system is attractive from a management point of view since it provides separation of functions. In situations where there are special timing requirements for intelligent devices or sensors, the dedication of processors to these devices may be appropriate. However, in order to support robotic devices, much larger centralized computer systems, possibly with peripheral memories, will be required. This is an important area for study since spacecraft computer technology will to a large part determine the sophistication and success of future missions.

The Study Group recommends that NASA plan to test and space-qualify LSI circuits in-house to reduce the apparent factor of 5 or 10 increase in cost of industry supplied space-qualified microprocessors and memories. Further, the Study Group believes that NASA should play an active role in encouraging the development of flexible computer architectures for use in spacecraft.

5. Computer Systems Technology

Current trends in the use of computer technology throughout NASA seriously impede NASA utilization of machine intelligence. Distributed processing techniques being adopted by NASA takes advantage of microcomputer technology to develop intelligent sensors and controllers of instruments. While microprocessors are well suited for simple sensing and controlling functions, many of the essential functions involving the use of machine intelligence and robotics technique require much larger processors. A flexible spacecraft computer architecture, within which both microprocessors and larger systems can coexist and communicate and cooperate with each other, seems to be a highly desirable goal for NASA.

The standardization of computer hardware which is intended to reduce costs by avoiding new hardware development and space qualification may result in the use of obsolete hardware. This will limit the resources available for a machine intelligence system, and possible preclude any effective implementations. NASA should look at developing techniques for software portability, or, equivalently, hardware compatibility in a family of machines. The desire to minimize software complexity may unnecessarily restrict experimental machine intelligence systems. Part of the problem rests with the issues of protection and reliability. NASA should reevaluate its hardware systems in light of recent techniques for providing resource sharing and protection in centralized systems.

The Study Group recommends a "software-first" approach to computer systems development within NASA so that hardware can be supplied as late as possible in order to take advantage of the latest technological advances.

6. Software Technology

The method of software development within NASA is in striking contrast to program development environments that exists in several laboratories working on machine intelligence. Compared with other users of computer technology, such as military and commercial organizations, NASA appears to be merely a state-of-the-art user. But compared with software development environments found in universities and research institutes there is a significant technological lag. The technology lag represented by this gap is not NASA's responsibility alone. The gap is indicative that an effective technology transfer mechanism does not yet exist within the computer field.

Software developed within NASA is often done in a batch environment using punched cards, resulting in a turnaround time of hours or even days. In contrast, the machine intelligence laboratories are characterized by being totally on-line and interactive. While debugging in a batch environment is a purely manual operation, requiring modification of the source program via statements to display internal values and intermediate results, many more programming aids are available in an interactive laboratory environment. Changes to programs are automatically marked on reformatted listings, the author and date of the changes are recorded, and the correspondence between source and object modules is maintained. In addition, extensive debugging and tracing facilities exist including interactive changing the programs data and restarting it from arbitrary checkpoints. The investment made to substitute computer processing for many manual activities of programmers should ultimately result in improved software quality and programmer productivity.

It should be emphasized that improved software development facilities can be created within NASA through the transfer and utilization of existing computer science technology. However, further improvements necessitate advances in the field of automatic programming which is an area of machine intelligence where programming knowledge (i.e., knowledge about how programs are constructed) is embedded

within a computer tool that utilizes this knowledge to automate some of the steps which would otherwise have to be manually performed. This is an area which deserves attention by NASA, perhaps towards developing specialized automatic programming systems tailored to NASA's needs.

The Study Group recommends immediate creation of an interactive programming environment within NASA and the adoption of a plan to use a modern data-encapsulation language (of the DOD ADA variety) as a basis of this facility. The Study Group also believes that NASA should initiate research towards the creation of automatic tools for software development.

7. Data Management Systems Technology

There are several data mangement issues where artificial intelligence techniques could be brought to bear. These areas range from the control of data acquisition and transmission, data reduction and analysis, and methods for dissemination to users. For example, onboard computers should perform data reduction and selective data transmission. This will minimize the amount of data transmitted and conserve communication channels and bandwidth. This requires an advanced computer capable of various types of data analysis. Once the data reaches a ground collection site, there are three types of data management functions required to make the data accessible and usable to researchers. First, the data must be archived. This is the simplest type of management which does not involve analysis of the data itself. For example, "Retrieve all data for the fifth orbit of the Viking mission." Secondly, access to specific portions or collections of the data, locating predetermined criteria such as "all infrared images centered over Pittsburgh taken between June and September of 1978" must be provided. Both archival and criteria selection management systems are well within current technology, and to some extent are available in systems similar to those at the EROS data center in Sioux Falls. However, the third type of database management function, the ability to access data by its content does not yet exist, and requires specific artificial intelligence support. It would utilize a knowledge base containing specific facts about the data, general rules concerning the relationships between data elements, and world models into which complex requests can be evaluated. This knowledge base would guide the system in locating data containing the desired attributes utilizing a predefined indexing criteria and the relationship of the desired attributes to the indexing attributes.

The Study Group recommends reexamination and evaluation of the NASA end-to-end data management system and the establishment of a systems engineering group consisting of computer scientists and hardware experts to achieve an effective system design and implementation.

8. Man-Machine Systems Technology

For both ground- and space-based NASA systems we would like to have the best integration of human intelligence and machine intelligence; but we lack an understanding of how best to combine these natural and artificial components. For example, to be more effective in the use of teleoperators, NASA needs to redress a basic lack of knowledge: there now is no satisfactory theory of manipulation on the basis of which to improve design and control of manipulators. The relative assignment of roles to man and computer and the design of the related interfaces require much better understanding than now exists.

In view of potential long-range payoff and the fact that such related research as exists within NASA has been ad hoc and mission-oriented, *the Study Group recommends support of significantly more basic research on man-computer cooperation, and, more generally, on man-machine communication and control.* NASA organizational entities representing life sciences and the technological disciplines of computers and control should develop better cooperative mechanisms and more coherent programs to avoid man-machine research "falling between the cracks," as has been the case. Future NASA missions can have the advantages of human intelligence in space, without the risks and life support costs for astronauts, by developing teleoperators with machine intelligence, with human operators on Earth monitoring sensed information and controlling the lower-level robotic intelligence in supervisory fashion.

9. Digital Communication Technology

Computer based communication systems have been used by the artificial intelligence community since the inception of the ARPANET network which is now used under NSF support to link approximately 500 non-computer scientists in about eight different research communities. These systems provide electronic mail (using distribution lists) and communication, and are used to give notices and reminders of meetings and reports. Online documentation of programs with instant availability to updated versions allow users access to information and programs at a variety of research sites. In addition, document preparation services including text editing systems, spelling correctors, and formatting programs are in common use. NASA would do well to adopt a computer based communication system since it would offer opportunities for improvements in management, planning, and mission implementation. If the system were a copy of existing systems at research sites on the ARPANET, software could be taken directly from those systems.

Appendix on Relevant Technologies

The principal activity of the Study Group during its existence was to identify information processing technologies that are highly relevant to NASA and to the success of its future programs. Each workshop had one or more of these topics as the foci of interest. Appendix A gives a complete list of topics covered at each of the workshops. In this section we provide detailed discussions of those topics which are considered by the Study Group to be of high priority for NASA.

1. Robotics Technology

This section discusses the need for advanced development of intelligent manipulators and sensors. The application areas for these devices range from the assembly of space structures to planetary rovers capable of autonomous execution of highly sophisticated operations. Research in the areas of robotics and artificial intelligence is necessary to ensure that future missions will be both cost-effective and scientifically valuable. In addition, results in robotics and artificial intelligence are directly applicable in the areas of automatic assembly, mining, and exploration and material handling in hazardous environments.

1.1 Need for Robotics Within NASA

Robotics and artificial intelligence have played surprisingly small roles in the space program. This is unfortunate because there are a number of important functions they could serve. These include, very broadly:

1. To enable missions that would otherwise be out of the question because of cost, safety, or feasibility for other reasons. Example: At rather low cost, we could have had a remotely-manned lunar explorer in progress for the past decade.

2. To enable the kinds of popular and valuable features that might rekindle public interest in the exploitation and exploration of space. Example: In the past decade, the hypothetical lunar explorer just mentioned would have been operating for 1,000,000 five-minute intervals. In this period, a vast number of influential public visitors could have operated some of the Explorer's controls, remotely, from NASA visitor centers. Imagine the education and enthusiasm that could come from such a direct public participation in space!

3. To achieve general cost reductions from efficient automation. Example: The Skylab Rescue Mission would have been a routine exercise, if a space-qualified teleoperator had been developed in the past decade. It would have been a comparatively routine mission to launch it on a military rocket if the Shuttle project encountered delays.

These things have not been done, in part, because NASA has little strength at present in the necessary technical areas. In our view the future prospects seem poor unless there is a change. We see several obstacles:

In-House Competence. NASA's current strength in artificial intelligence is particularly low. NASA's in-house resources are comparatively weak, as well, in computer science on the whole, especially in areas such as higher-level languages and modern debugging and multiprocessing methods.

Self-Assessment. Even more serious, NASA administrators seem to believe that the agency is outstanding in computation science and engineering. This is far from true. The unawareness of weakness seems due to poor contact of the agency's consultants and advisors with the rest of the computational research world.

Superconservative Tradition. NASA has become committed to adhere to the concept of very conservative, fail-safe systems. This is eminently sound in the days of Apollo, when (i) each successful launch was a miracle of advanced technology and (ii) the lives of human passengers were at stake. But today, we feel, that strategy has become self-defeating, leading to unnecessarily expensive and unambitious projects.

Fear of Complexity. On a similar note, we perceive a broad distrust of complicated automatic machinery in mission planning and design. This distrust was based on wise decisions made in the early days of manned space exploration, but it is no longer appropriate in thinking about modern computation. Instead of avoiding sophisticated computation, NASA should become masterful at managing and exploiting it. Large computers are fundamentally just as reliable as small computers.

Fear of Failure. Many NASA people have confided to the Study Group that the agency is afraid that any mission failures at all may jeopardize the whole space program, so that they "cannot take chances" in advanced design. Again, this attitude was sound in the Apollo era, but probably is not sound when we consider the smaller, multiple, and individually inexpensive missions of today.

What Are the Alternatives? We feel that NASA should begin to consider new styles of missions which are, at the same time, more adventurous and less expensive. Left as it is, NASA's thinking will continue to evolve in ways that will become suffocatingly pedestrian. To get out of this situation, it will be necessary to spend money, but the amount needed to learn to do exciting things like using powerful computers and semi-intelligent robots will be small compared to the money needed in the past for developing propulsion systems. "Getting there" is no longer all the fun; it is time to think about how to do sophisticated things after the mission arrives there.

Space Programs and Intelligent Systems. It is extremely expensive to support personnel in space for long periods. Such costs will render impossible many otherwise exciting uses of space technology. Yet, our Study Group found relatively little serious consideration of using autonomous or semi-autonomous robots to do things in space that might otherwise involve large numbers of people. In many cases, the use of artificial intelligence had not been considered at all, or not considered in reaching conclusions about what computer resources will be needed, or prematurely dismissed on the basis of conversations with the wrong people. In other cases, it was recognized that such things were possible in principle, but out of the question because of NASA's mission-oriented – as opposed to technology-oriented – way of planning for the future.

Two examples come to mind as obvious illustrations of cases where we found the views expressed to be particularly myopic:

(1) **Building Large Space Structures.** Large-scale constructions usually involves two activities. First, basic building blocks must be *fabricated* from stock material. Second, the building blocks must be *assembled*. Space fabrication seems necessary becuase of difficulty in launching large prefabricated sections. We applaud the work that NASA has done already toward creating machines that continuously convert sheet metal into beams. We are less happy with the lack of justification for automatic inspection and assembly of such beams. There are existing automatic vision and manipulation techniques that could be developed into practical systems for these tasks. The beams could be marked, during fabrication, so that descendants of today's visual tracking programs could do rough positioning. And, force-sensing manipulators could mate things together, once roughly positioned. Where large structures are concerned, in fact, these are areas in which reliable, accurate, repetitive human performances would be very hard to maintain.

(2) **Mining.** An ability to build structures is probably a prerequisite to doing useful, economically justified mining on the Moon, the planets, and the asteroids. But the ability to build is only a beginning. The vision and manipulation problems that plague the robot miner or assembler are different. Rocks do not have fiduciary marks, and forces encountered in digging and shoring are less constrained than those involved in screwing two parts together. On the other hand, less precision is required, and even interplanetary distances do not prevent the exchange of occasional questions and return suggestions with Earth-based supervisors.

1.2 The State of the Art

At this point, we turn to some specific areas, both to draw attention to NASA's special needs and to tie those needs to the state of the art.

Basic Computer Needs. A first step toward enabling the use of artificial intelligence and other advanced technologies is to use more sophisticated computer systems. We conjecture that the various benefits that would follow from this approach could reduce the cost of spacecraft and ground-based operations enough to make several missions possible for the present cost of one.

We want to emphasize this point strongly, for we note a trend within NASA to do just the opposite! In our Study Group meetings with NASA projects over the year, time and time again we were shown "distributed" systems designed to avoid concentrating the bulk of a mission's complexity within one computer system. However, we feel that this is just the wrong direction for NASA to take today because computer scientists have learned much about how to design large computer systems whose parts do *not* interact in uncontrollably unpredictable ways. For example, in a good, modern "time-sharing system" the programs of one user – however badly full of bugs – do not interfere either with the programs of other users or with the operation of the overall "system program." Thus, because we have learned how to prevent the effects of bugs from propagating from one part to another, there is no longer any basic reason to prefer the decentralized,

"distributed" systems that became the tradition in the "fail-safe" era of engineering.

However, because NASA has not absorbed these techniques, it still distrusts centralization of computation. We argue elsewhere that this leads to very large and unnecessary costs of many different kinds.

The Development of Sophisticated Manipulators. We feel that NASA has not adequately exploited the possibilities of even simple man-controlled remote manipulators. The Skylab sunshade episode might well have been easily handled by an onboard device of this sort, and we think it likely that it would have paid for itself in payload by replacing some variety of other special-purpose actuators.

The need to handle radioactive substances led to the development of rudimentary teleoperators many years ago. At first progress was rapid, with force-reflecting, two-fingered models appearing in early 1950s. But, strangely, this development all but stopped when progress was sufficient to make the handling of nuclear materials possible, rather than easy, economical, and completely safe. We believe that this happened because the nuclear industry, like NASA, became at this time mission-oriented rather than technology oriented -- so that places like Argonne National Laboratory lost their basic research and long-view funding.

Consequently, today manipulators differ little from their 1950s ancestors. They are still two-fingered and they still leave their operators fatigued after a half-hour or so of use. Even today, there is no generally available and reliable mobile and dexterous manipulator suitable for either emergency or preventive maintenance of nuclear plants -- this is still done by people working under extremely hazardous conditions. Concerns within a nuclear plant about storage safety, detection of faults, and adequacy of emergency systems are perhaps best handled using a mobile and dexterous robot.

If such devices had been developed -- and space-qualified versions produced -- NASA could have exploited them, both for teleoperator (human-controlled) and for fully autonomous (robot) use. Indeed, we feel, NASA's needs in this area are quite as critical as those in the nuclear industry. Nevertheless, NASA has not given enough attention to work in the area. Perhaps a dozen or more clumsy two-fingered systems have been developed, but all of these would be museum pieces had the work gone at proper speed.

It therefore makes sense for NASA to enter into a partnership with ERDA to reverse the neglect of manipulator technology. A good start would be to sponsor the development of a tendon-operated arm with a multifingered hand, both heavily instrumented with imaginative force, touch sensors, and proximity vision systems. Besides the obvious value -- in space -- of separating the man and his life-support problems from the workspace, there are many obvious spinoffs in general manufacturing, mining, undersea exploitation, medicine (micro-teleoperators), and so forth.

Controlling a Manipulator: Still a Research Problem. Dynamic control of the trajectory of a many-jointed manipulator seems to require large calculations, if the motion is to be done at any speed. It takes six joints to put a hand at an arbitrary place at an arbitrary orientation, and the six degrees of freedom have interactions that complicate the dynamics of arm control. The equations are too complex for straightforward real-time control with a low-capacity computer. The problem can be simplified by placing constraints on manipulator design, for example by designing the axes of rotation of the last three joints to intersect, but even the simplified problem is not yet solved.

In any case, the most obvious approach -- to put an independent feedback control loop around each joint -- fails because constant feedback loop gains cannot manage (at high speeds) the configuration-dependent inertia terms or the velocity interaction terms. On the other hand, it seems clear that such problems can be solved by combinations of "table look-up" for sample situations with correctional computations. In any case the control computer will need a central memory that is large by today's space standards.

Rover Mobility, Locomotion, and Guidance Research. Although much knowledge regarding several of the solar system planets has been gained through missions employing remote sensors, and more can be obtained in the future in this manner, many of the critical scientific questions require detailed surface experiments and measurements such as those conducted by the Viking landers on Mars. Despite the historic achievement represented by the soft landing of the Vikings and the effectiveness of the onboard experimental systems, more new important questions were raised. For these to be answered, an extensive surface exploration should be undertaken. A surface trajectory involving hundreds of kilometers, and desirably over 1000 kilometers, would be required to explore a sufficient number of the science sites on Mars to gain an adequate coverage of the planet.

The round-trip communications delay time, which ranges from a minimum of nine minutes to a maximum of forty minutes for Mars, and the limited "windows" during which information can be transmitted precludes direct control of the rover from Earth. Accordingly, a rover on Mars or another planet must be equipped with sensors and appropriate computing capability and procedures to proceed autonomously

along Earth-specified trajectories. The intelligence of this path selection system, together with the basic mobility characteristics of a rover, determine whether scientific sites of specific interest can be reached, given the characteristics of the approach terrain and the distances between sites. It follows that a low-mobility rover equipped with a high-quality path selection system will not be able to reach particular sites nor could it undertake an extensive mission. It also follows that a high-mobility rover guided by a low-quality path selection system would be limited in a similar fashion. Therefore, systematic research programs aimed at maximizing both the rover mobility and the intelligence in path selection systems consistently should be undertaken to provide a sound basis for the planning and execution of surface exploration of solar system bodies.

The term "mobility" includes several characteristics which when taken together describe collectively the capability of the rover to deal with specific classes of terrain.

1. The stability of the rover in terms of the in-path and cross-path (i.e., pitch and roll) which the rover can handle without the hazard of overturning. This characteristic is not only important in terms of the general slope characteristic of the terrain surface, but especially in connection with boulders and trenches on which individual propulsion elements may find temporary purchase (foothold).

2. The maneuverability of the rover, i.e., the turning radius and dynamical characteristics, will determine what terrains in the large sense will be open for exploration. Unless the rover is able to execute strong turning trajectories and maneuver in close quarters, many areas will be forbidden.

3. Clearance of the payload above the propulsion units will have a direct effect on the available paths. A rover whose clearance is adjustable will not only offer prospects for recovery should the rover become hung-up but may also offer additional scientific capabilities. Finally, an adjustable clearance would allow for the rover's center of gravity to be reduced temporarily in situations where the critical pitch/roll conditions are approached to increase safety or to permit the rover to follow a normally unsafe terrain.

4. The rover's speed capabilities will have a direct effect on the scope of the time required for the traverse between specified science sites.

5. Locomotion is a very major factor since it exerts a primary limit as to what terrains can be handled. The three major alternatives available, wheels, tracks and legs, not only offer varied propulsive and maneuverability capabilities as well as potential sensor information for guidance, but also pose unique as well as general control problems.

With respect to the propulsive and maneuverability factors, wheels and tracked units can be designed to achieve the required footprint pressures and traction required to deal with soft, loose materials such as ultrafine sand as well as hard coherent terrain forms such as boulders. Wheels have the advantage of being able to change direction with a minimum of scuffing and to tolerate small obstacles in lateral motion. The tracked units have the advantage of being able to bridge larger trenches but offer potential problems in turning on irregular terrain.

Neither the potential nor the limitations of such concepts have been firmly established and a systematic research and development program would appear to be in order. Such a program should be aimed at developing maximum carry load-to-wheel weight ratios consistent with reliability, footprint pressure, turning capabilities, and dimension.

A legged vehicle, which makes use of six or eight legs of varying joint complexity, would appear to offer decided advantages over wheeled or tracked vehicles in extremely rugged and irregular terrain. Depending on the number of segments and their lengths and the degrees of freedom provided by the connecting joints, a rover capable of dealing with extraordinarily irregular terrain and possessing exceptional climbing ability is potentially feasible. Maneuverability and stability potential of such a rover could exceed that of wheeled or tracked rovers. However, the feet of such a device may raise a serious problem. Rather large feet would be required to provide a sufficiently low footprint pressure on soft or loose terrain. On the other hand, such big broad feet might seriously limit the rover's capability in gaining a firm purchase on very irregular terrain. Research on legged vehicles has been very limited in the United States. At the present time, McGee at Ohio State University has an active hardware program. Considerable efforts are apparently underway in the Soviet Union but virtually nothing is known of the details of this work, other than that they are proceeding vigorously. Successful development of a legged vehicle would apply to environmentally delicate regions such as tundra as well as space exploration.

The control of either wheeled, tracked, or legged rovers represent problems of substance which will require study. The wheeled or tracked vehicle control system will have to respond to constraints imposed by irregular terrains. In the case of locomotion on a flat plane, it is a straightforward matter to

specify a vehicle speed and a steering angle to a computer-driven or hard-wired control system to drive each wheel at the proper speed to achieve the desired motion without scuffing and without excessive stresses either on the propulsion system or the vehicle structure. However, if the vehicle is on irregular terrain so that the axle velocity vectors are no longer coplanar, then each wheel must be driven at a specific rate to achieve the desired result. Wheel speed and torque as well as the vehicle strut position locations, possibly force or stress sensors, and the pitch/roll of the rover will have to be combined with trajectory parameters to achieve an acceptable system.

The legged-vehicle control problem is of a different character. Certainly all the dynamic control problems discussed above in connection with manipulation reappear. Moreover, additional problems come up. The gaits (sequences in which the legs are moved)-which are selected are a function of the terrain to be traversed and the desired speed. The specific motion of an individual leg may also be a function of the terrain. In the case of irregular terrain, a significant lift of the leg to avoid hazards will be required before the foot can be lowered to the desired position. Sensors and control systems controlling the motion will have to be developed.

In order for the rover to autonomously execute highly sophisticated operations in an unpredictable environment, it must be capable of real-time interaction with sensory feedback. It must be capable of selecting and modifying its behavior sequences in response to many different types of sensory information over a wide range of response times. For example, the mobility system should respond almost instantaneously to pitch and roll accelerations, but may tolerate longer time delays as it picks its way around small rocks and ruts on a meter by meter basis. It should anticipate larger obstacles two to five meters ahead and impassable barriers should be detected 5 to 100 meters in advance. Minimum energy pathways along contour lines, through valleys, and between hills should be selected 0.1 to 1 km ahead, and long range navigational goals should be projected many kilometers ahead.

Similarly with manipulation, position servo corrections must be applied with very short time delays, whereas feedback from proximity sensors can be sampled only a few times per second in order to modify approach path motions which move slowly over a distance of a few centimeters. Processing of feedback in order to select alternative trajectory segments during the execution of elemental movements is more complex, and can be done at still coarser time intervals. The modification of plans for simple tasks to accommodate irregularities in the environment, the modification of complex task plans, or changes in scenarios for site exploration require increasingly complex sensor analysis processes which can be safely carried out over longer time intervals. The most natural way to deal with this hierarchy of ascending complexity and increasing time intervals is to map it onto a computing mechanism with the same hierarchical structure.

The important issue in the mobility control hierarchy is the wide range of time and distance scales to which the sensory data must interact with the mobility system. Some types of feedback must be incorporated into the control system with millisecond and centimeter resolution, while other feedback can be incorporated at intervals of days or kilometers. Only if the control system is hierarchically structured can such a wide range of resolution requirements be easily accommodated.

Automatic Assembly and Force Feedback. The most naive concept of automation is to make a robot that will repeat pre-programmed motions over and over. This will not work in many situations; using position control alone, a robot cannot insert a fastener in a tight hole or even turn a crank – because the inevitable small errors would cause binding or breakage. Consequently, it is necessary for robot manipulators to use force-sensing feedback or the equivalent.

In the 1960s, experimental systems demonstrated such methods for automatic assembly. Centers in Japan, the U.S., and the U.K. succeeded nearly simultaneously. In one such demonstration, Inoue, working at MIT, used an arm equipped with a force-sensing wrist designed by Minsky to assemble a radial bearing. Shortly thereafter researchers at the Draper Laboratory exhibited a device to do some kind of work with carefully arranged passive, compliant members replacing active force-sensing feedback loops. Using such techniques, we think that much of the automatic assembly of space structures already comes near to the state of this art.

Automatic Assembly and Problem Solving Systems. Problem solving and languages for problem solving has been a central focus in artificial intelligence since the science began. In the earliest stages of AI, it was seen that a computer could be programmed to try a variety of alternatives, when it encountered a situation not specifically anticipated by the programmer. Soon these "heuristic search" programs were succeeded by "goal-directed" problem-solvers, notably the GPS system of Newell and Simon at Carnegie-RAND. The symbolic integration program by Slagle is perhaps the best known example from that era.

Since that time, there has been a steady stream of new ideas, both for more general theories and for the design of problem solvers for particular problem domains. This work led to a variety of new computational organizations and languages. LISP, PLANNER, CONNIVER, STRIPS, QA4, and Production Systems are representative of this conceptual evolution. The

MYCIN program for bacteriological diagnosis and treatment, and the PARSIVAL program for analyzing English syntax are representative of what can be done to attack small, well-defined domains.

In the last few years, some steps have been taken to apply the resulting technology to the problem of assembly automation. It would be impractical and unreliable to program assembly machines at a low level, giving step-by-step instructions about exactly where to move the hand and exactly when to close it around an object. It is better – and easier in principle – to design languages with embedded problem-solving apparatus, so that the "programmer" can give instructions in much the same way as one communicates with people. First, one states the general goal, names the parts, and suggests an order in which the parts should be put together. Later, one makes suggestions as they come to mind, or if and when the assembly machine gets stuck.

Several research centers are now working on such problems, among them Stanford, SRI, IBM, NBS, and MIT. A full solution is some distance off, but the work has the fortunate character that each step in basic progress yields a corresponding step in application. In early stages, the amount of suggestion and detail supplied by the programmer is large, but the amount decreases as the problem solver gets smarter and knows more.

Automatic Assembly and Vision. We are still far away from being able to make a computer "see" – to describe and recognize objects and scenes as well as a person can. In spite of much brillant work done in this field, "general-purpose computer vision" is still far away. Still, the special and controllable environments involved in manufacturing have enabled exciting demonstrations with near-term promise. One of these, done by Rosen and his colleagues at SRI, uses binary image processing techniques to identify parts and their orientation after they have been placed randomly on a light table. In other work, done under the direction of Horn at MIT, inspection programs have successfully examined watches to make sure the hands are moving, castings to make sure the grain structure is correct, and IC lead frames to make sure the pins are straight. We believe that this sort of work has great promise of enabling work in space that might otherwise never be done. Still, we emphasize that of all problems described here, computer vision is likely to prove the most difficult and most deserving of attention and funding. The successful examples cited are included only to suggest that there is a technology to be explored for potential uses within NASA, not that there is a technology that can be merely bought. There is, for example, no general systems for selecting parts from a bin, even though it is well-known to be a serious problem and even though everyone in the field has thought about the problem from time to time.

1.3 Recommendations

We must re-emphasize two major obstacles to addressing the needs just outlined. The first is that of a fail-safe attitude. NASA pioneered in achieving extraordinary reliability in its fail-safe, redundant designs for missions. We have the impression that the use of these techniques is persisting in new problems to the point of some dogmatism, overlooking new possibilities enabled by progress in computer technology. In particular, we believe a great increase in flexibility and reliability might be obtained through centralizing many operations within one computer. But we see an opposite tendency; to design multiple, "distributed" computer systems that limit the flexibility of the system. On the surface, this seems sensible; but we believe that it leads to overlooking other, more centralized ways to do things that may be cheaper, more versatile, and at least equally reliable. For example, one might imagine missions that depend utterly on one central computer and one manipulator to replace many special systems. Of course, one of these two components might fail and lose the mission. On the other hand, eventually such a system might be (1) an order of magnitude cheaper and (2) possibly more reliable – because of extensive concentration on the two components and because of their ability to salvage or repair other failing components.

NASA's second major problem is that its current strength is low in artificial intelligence and even in general computer science. There are few people within NASA who understand the state-of-the-art. There is no place where those who do artificial intelligence work can reach critical mass with respect to the number of high-quality researchers or with respect to computational and other supporting resources. This has led to three regrettable consequences. First, present NASA workers are unable to be maximally productive. Second, it is extremely difficult to attract talented people to NASA. And third, those people in NASA that most need advice on artificial intelligence do not find it. Instead, they incorrectly suppose that they must be in good hands because NASA spends a great deal of money on computation.

This has led to a great gap. Much of what NASA does with computers is years out-of-date. Worse, with only a few exceptions, influential people in NASA do not realize how out-of-date most of their thinking has become. In such areas as computer languages, the situation is nearly scandalous. Part of the problem has to do with mission-oriented horizons, and part with distrust of outside researchers. Because typical "Earth-bound" workers do not have such concern with reliability and simplicity, we conjecture that NASA mission

workers feel that the techniques of non-NASA people are inapplicable. Instead of working with AI and robotics projects outside, NASA has tended to try to build its own. But these projects have never reached critical mass and have not attracted enough first-rate workers. The problem is connected, again, with the lack of modern computing power; modern vision and robotic control concepts require large computer programs and memories. We believe that there is no reason such systems cannot be space-qualified, and that they need not be very heavy or power-hungry. But without them, it is hard to use modern ideas about control and operations.

How to Correct the Central Problem of Insufficient Expertise. One idea is to contract with computer companies to provide advice and needed research. This idea, however, will not work. The large companies NASA is comfortable working with have not yet developed strength in artificial intelligence. NASA can only be led into a false sense of security by relying on them. Alternatively, NASA could increment its small existing budget for artificial intelligence and related topics, increasing the funds available at existing places. This also will not achieve the desired results. Indeed, such a plan could be counterproductive. The nature of the work demands a community of highly-motivated people working together. Efforts below critical mass in human or other resources are not likely to do well and such efforts could therefore lead to pessimism rather than excitement.

Still another possibility is that NASA could fund university research. This is a reasonable alternative as long as it is again understood that small, subcritical efforts are not cost-effective. Only a half-dozen university centers have sufficient existing size and strength to do really well. And finally, NASA could establish its own center. This is a good choice, especially if done in close proximity to and collaboration with an existing university center. It is our opinion that the need for artificial intelligence in space argues for such a center in the strongest terms. We believe that artificial intelligence will eventually prove as important to space exploitation and exploration as any of the other technologies for which there are large, focused, and dedicated NASA centers today.

Future NASA Role. At a certain level of abstraction, NASA's needs are not unique. Certainly such things as automated assembly and mining would be useful on Earth as well as in space. But it would be folly for NASA to expect someone else to produce the needed technology. NASA should plan to be the donator of artificial intelligence robotic developments rather than the benefactor for several reasons.

First, not enough is happening for reasons ranging from the shape of our antitrust laws to the lack of congressional concern for our declining position in productivity. Second, the extreme cost of placing people in space ensures that using robots and/or teleoperators will be the method of choice in space assembly and mining long before robots see much action on Earth. Consequently, cost/benefit ratios will be more of a driving force to NASA than to others. And third, doing things in space is sufficiently special that NASA must be in the act in a major way to ensure that the technology progresses with NASA's interests in mind. Otherwise, all NASA will have is a technology that is solving someone else's problems but skirting NASA's.

The Virtual Mission Concept – A Special Recommendation. The establishment of research efforts, well endowed with human and financial resources, should be accompanied by a new kind of attitude about mission planning and development, particularly with respect to space qualification of hardware. As it stands today, work seems to be done in two primary contexts, that of the paper study and that of the approved, assumed-to-fly mission. This automatically ensures two crippling results.

First, since the execution of a mission is very expensive, only a small number of the promising ideas will go forward to the point of full and fair evaluation and to the point of generating spinoff technology. Second, since space qualification is an assumed starting point for all thinking, the technology employed in mission development is guaranteed to be years behind the state of the art. The chances for pushing the state of the art via spin-offs is smaller than it should be. Paper studies, on the other hand, tend to produce mostly paper.

Consequently we see the need for a new kind of research context, that of the *virtual mission*. Such missions would have the same sort of shape as real missions, with two key exceptions: first, space hardened and qualified hardware would not be used; second, the objective would not be to fly, but rather to win an eligibility contest. As we see it, there would be many virtual missions competing for to-be-flown status. Taken together, they would produce a pool of alternatives, any of which could be selected and flown, with space qualification taking place after, rather than before selection. Since none would be fettered by the limits of space qualification for their entire life, all would be more imaginative, technically exciting, and technically productive by way of spinoff technology. We believe that the costs involved in doing things this way are likely to be reduced. Conceivably, several virtual missions could be done, using commercial equipment where possible, for the price of one, whereas real missions are restricted as now to old fashioned, one-of-an-obsolescent-kind antiques.

There could be, for example, several groups competing by applying different locomotion schemes to the same exploration job. Similarly, several groups could explore a variety of shuttle-based, large-structure assembly ideas.

We should begin thinking along these lines because spacecraft computer hardware is becoming more and more out-of-date and something simply must be done about it. Today it takes too long to "qualify" space computers. There seems to be no mission-independent way to do this. Individual missions have to use computers qualified by previous, almost accidental, qualification incidents. Memory sizes, in particular, are much too small. This leads to weak programs with minimal versatility and to doing things in hardware that might be lighter and more reliable in software. Therefore, at the very least, NASA should have a continuing program to space-qualify larger memories and more capable computers, as they evolve. We know enough about computation, today, to be able to assert that there is little reason to suppose that the computers will have to be adapted to particular missions much, except in regard to overall capacity parameters.

Recommendations for Rover Research. Given the need to take advantage of imminent opportunities on Mars, we believe the design, construction, and systematic evaluation of a *functional* reconfigurable rover should be undertaken to:

1. Determine optimal configuration alternatives from the standpoint of stability, maneuverability, and clearance with weight as a major, if not the major, consideration.

2. Evaluate alternative wheel/tracking/leg concepts as a function of terrain classes with respect to speed, steering, obstacle climbing ability, weight, and reliability both in the laboratory and out in the field.

3. Serve as a test bed for the development and evaluation of alternative vision/sensor/calculational/guidance control systems applicable separately to long-range, mid-range, and short-range path-planning levels and to integrated systems ultimately. There is a corollary need to develop additional sensors to provide real-time sensory feedback with a much broader range of spatial and temporal resolution.

2. Smart Sensor Technology

This section comments on NASA programs which use vision science to make an impact in its applications and mission programs. It summarizes the state of the art in the required technology areas, summarizes research recommendations, and suggests a structure which will encourage required research.

NASA conducts large imaging programs which produce enormous volumes of images. NASA programs are studying means of making image data more available and more useful for users (NASA End-to-End Data Management System program). Those activities are largely for presentation of images to humans for human perception. Those NASA projects with large potential benefit to society which involve machine visual perception include:

1. Construction of large space structures, particularly communications systems and antennas.

2. Remote sensing and agricultural resource evaluation.

3. Cartography.

4. Meteorology.

Advances in computer vision would enable increased effectiveness of the proposed Mars rover mission. These applications require a compact area of vision science and technology. NASA's vision applications are sophisticated. Suggestions are made to advance NASA objectives by:

1. Collaboration with other organizations which have an investment in applications requiring similar technology and which support research in this area of image science.

2. Involving the most advanced research groups in research program formulation and implementation.

3. Evaluation of current NASA imaging programs.

2.1 Introduction

Automated imaging and mapping systems are planned to meet objectives of NASA application programs and missions. Earth resources surveys include crop production, water resources, land use, forest resources, ocean resources, and oil spill monitoring. Meteorological prediction, monitoring, and climatic studies already make an impact in daily life. Geological studies include crustal dynamics and a world geological atlas. Large space structure construction is likely to be important for communications. Automated sensing has a role in these applications.

These activities overlap responsibilities of other organizations such as USGS, Forest Service, Defense Mapping Agency, etc. Capabilities necessary for NASA functions enable NASA to contribute significantly to development of automated imaging for civilian purposes. A major part of these NASA programs requires innovative and high level research to develop

required technology. NASA can lead in development of this technology. Organizations with similar responsibilities have little resources to sponsor and direct research. A major emphasis of this section is that it is important for NASA to get "leverage" in research and applications, that is, to work with existing research programs and to work with potential users of the technology.

NASA performs two functions in this area, data distribution and information extraction. Most effort has gone into data distribution. Much work remains. Current and planned imaging missions provide volumes of data beyond existing abilities to catalog, distribute, and assess the images. Smart sensors for data compression, automated image handling facilities, and high performance computers for imaging are needed. These needs are recognized by the NEEDS (NASA End-to-End Data Systems) program which addresses smart sensors, special purpose imaging computers, and image handling facilities.

A greater need exists in information extraction. Here, NASA's objectives require sophisticated vision science which has not yet been achieved. That need is recognized within NASA. The Space and Terrestrial Applications program is soliciting proposals for new technology in remote sensing and terrain topography. The content of the recommendations of this report is that efforts to develop new technology should be intensified, and that they should be strengthened by strong participation of major research groups outside NASA and by cooperation with other organizations with similar needs. The balance between research and production systems should be evaluated; a heavy research component is essential. Careful examination should be made of current and proposed production systems to evaluate whether they are founded on an adequate technology base.

2.2 The State of the Art

Remote Sensing and Crop Survey. An examination of crop census systems reveals that their performance is very limited. In summary, those systems do not seem to have met expectations of lowered cost and increased repeatability from automated classification. In these systems, humans make decisions, aided by computer clustering. The overall system accuracy is about 90%. Their computerized classification is not that good. What humans currently contribute to classification is use of spatial context. Both structural pattern recognition and scene analysis offer techniques to use spatial context in identification. Structural pattern recognition experiments indicate significant improvements in performance. Our evaluation of the mix between development and research indicates that a higher proportion of research and more innovative research should be supported, and that research results be incorporated into development systems continuously, with little lag. It appears that a production system was built with obsolete and inadequate technology.

A likely requirement for the application of structural pattern recognition and scene analysis techniques is imagery having much higher resolution than LANDSAT. Eighty meters resolution is probably too crude to use structural relations. High resolution imaging may make use of aerial photography, which is part of NASA's domain. A crop survey using structural analysis at high resolution is perhaps feasible now, and will be feasible in a few years. A scenario is outlined below which would require about 3.4 years to do a world-wide crop survey at 10^8 ops/second. The ultimate resolution is about 2 cm per pixel. Estimates are based on a two-stage analysis. For the first stage, a coarse sampling at 2 m/pixel is probably adequate. Alternatively, a coarse grid of linear scans would require about the same computation cost. The first stage is intended to separate major field boundaries. The second stage would use structural analysis at 2-cm resolution on limited parts of the fields. The use of smart sensors (for example, edge operators under development in the DARPA Image Understanding program) would be useful in this program. Smart sensors would cut computation cost significantly.

The Earth's area is 2×10^{19} cm^2. About 1/4 is land and of that, half is arable. If we sample 10% at 2 m/pixel, there are about 5×10^{12} pixels. Assume about 1000 ops/pixel for reasonably sophisticated processing, and 10^8 ops/second. Then the required computation time is 5×10^7 seconds. There are 3×10^7 seconds per year. A single computer would require 1.7 years now. An equal amount of computation would be required for the second stage, for a total of 3.4 years. If we assume that semiconductors will increase density at the rate of a factor of 2 every two years (a factor of 2 per year is the current rate) and if we assume a factor of 4 speed increase in 5 years (the historical rate), then in about 10 years, a single computer will be able to make a world-wide sampling in four days. The vision science and software technology should be developed now to make use of that computing power.

Cartography. The production of maps by traditional means is labor intensive. Partial automation of elevation contour mapping has been in use for years by DMA, with analog stereo correlation systems. It is often thought that automated stereo mapping is a solved problem because there are production systems; however, these systems require a great deal of human intervention. Typically, they are interactive systems in which the operator redirects the system whenever it gets lost and patches up errors. There are problems when tracking over

water and over uniform surfaces such as concrete. They do badly at surface discontinuities such as edges of buildings and cliffs. In trees, picking out the ground surface is beyond the capability of the system. The extent of human intervention required is enough to decrease mapping speed and limit mapping output.

The DMA has made a major study in automating cartography in a largely digital system. DMA studies revealed extensive requirements for advanced techniques in computer science with an emphasis on machine intelligence. There is a strong relationship of many DMA concerns with related issues in NASA particularly in the area of scene analysis and understanding, large database management, and information retrieval.

Research in stereo vision, some of it supported by NASA, has produced stereo systems which work in a research environment and has produced advances in our scientific understanding of stereo vision. A model is emerging of the stereo vision process from which newer high performance systems are being designed and developed. Preliminary research in linear feature tracing has been carried out and the results indicate that interactive systems using tracing aids are feasible for features such as roads and rivers. There is a growing body of research on edge finding systems which will support development of such aids to linear feature tracing. Building large data bases for cartographic applications requires the integration of research in vision, machine intelligence, and general systems issues in computer science.

Teleoperators. This issue is shared between the Study Group's vision and robotics subcommittees. This section will address only the vision part of teleoperator work in space. The building of large space structures for communications systems and possible experimental stations appears likely. The cost of maintaining a human worker in orbit, including life support systems and shielding from radiation, is estimated at $1.5M per year. It is hard to assess the difficulty of maintaining a crew of highly trained workers in this hazardous environment. Possible space power stations and space industrialization projects would involve large construction efforts. Development of teleoperator manipulation offers the possibility of increasing the productivity of human workers, while lowering their risk. Operation with large objects, such as the Shuttle-Attached Manipulator, imposes another requirement for advanced teleoperator systems.

This technology would contribute to electric power generation, to undersea oil drilling and mineral exploitation, and to rehabilitation of disabled people. Recent incidents with power shutdowns in nuclear electric power stations have highlighted the technical problems of servicing reactors. Work is currently done in a radioactive environment by humans. Advanced capabilities for remote operation with man in the loop offer opportunities to reduce hazards to workers, lower the cost, and increase the level of maintenance. On-line monitoring and maintenance are other possibilities. A high payoff is expected for a partially automated system. In this type of system, the teleoperator system takes over a set of limited operations, using sensing and knowledge of parts. Once the operator has positioned the manipulator to approximately the right orientation, the system completes the action itself. The payoff is in speed and ease for the operator.

Technical requirements for this application require the development of manipulator hardware, control systems, software, and sensor systems, in addition to a vision system. The vision system required for the simplest of teleoperator systems needs the ability to present multiple views, and could benefit from stereo if satisfactory stereo systems can be developed. For partially automated systems, stereo vision and the use of multiple views are highly important. Even when the views are separate (i.e., wide angle views which cannot be fused), the sort of modeling which is involved in stereo vision is important for autonomous vision in these contexts. Considerable use can be made of knowledge of the design of parts and joints, for model-based vision systems.

Mars Rover. A proposed Mars rover mission requires considerable onboard autonomy if one expects to achieve the objectives of a few hundred meters navigation per day, with communication for a short time once per day and round trip signal times of twenty-five minutes. The minimal navigation device is a laser ranging device. Its two limitations are limited range and limited number of samples. These limitations put restrictions on its reliability and utility since such a sensor can do little in looking for interesting samples. Navigation using only this device can be only local, with little look-ahead and low resolution. Under these conditions, the rover is likely at some time to reach a dead end that it can't back out of, or waste excessive time in getting out of, because of limited search strategy options.

NASA does sponsor some research in stereo vision. This is on a small scale and should be expanded. Functionally, stereo vision with motion parallax offers capabilities to maintain orientation by navigating with respect to landmarks, and to allow depth ranging and maping of distant objects by making use of large baselines accumulated in motion. It is thus possible for the rover to avoid problems and to return to base locations.

2.3 Recommendations

We recommend evaluation of NASA participation in the development of advanced automation in cartography for civilian purposes. Cartography and land use studies appear to be important applications areas. Progress in computer stereo vision makes possible major advances in cartography. The civilian organization with responsibility in this area, USGS, has limited facilities and limited research. Because of the strong relationship of the Defense Mapping Agency Pilot Digital Operations Project with NASA interests, it is recommended that NASA maintain strong liaison with the DMA and investigate possible collaboration with their efforts. NASA should evaluate the DMA planning process to aid in costing the development of detailed plans for implementing some of the related suggestions of this Study Group. A collaborative research program with DMA and USGS would have high potential benefit, and would be strengthened by research underway in DOD, particularly for cruise missile guidance.

We recommend the support of research in computer stereo vision for teleoperators intended for remote construction and maintenance of large space structures for communication facilities in space. Antennas and communication systems in space appear to have economic benefits in a reasonable time scale. We recommend that a small investment be made which would increase productivity of remote operations as the cost per man-hour in space will be high. Advanced teleoperator technology would lower exposure of human workers in the radiation belts and increase their effectiveness. The technology would be equally useful for large space structures for space power stations or space industrialization should NASA undertake them.

We recommend that NASA increase support of computer stereo vision for a proposed Mars rover mission. Current progress in stereo vision promises improved capabilities and increased scientific payoff.

We recommend that agricultural remote survey applications be reevaluated. It is urged that performance limitations of the current technology be evaluated. NASA should study the feasibility of using more powerful structural pattern recognition and scene analysis approaches, and that systems be built which incorporate new technology. Crude estimates indicate that high resolution structural analyses may be feasible soon for crop census.

We recommend that NASA support basic research in structural pattern recognition and scene analysis approaches.

We recommend that NASA diversify its research base in imaging research, that it evaluate the proportion of research to development investment. It is suggested that NASA support research at centers of excellence in computer vision. This approach is cost-effective since it is not necessary to support whole programs; these centers have broad support and well-established programs. This approach provides a means of collaboration with related research programs. It is suggested that the emphasis be on innovative focused research, not on applied research. It is recommended that a vigorous program of evaluation by members of the research community be used for program formulation and proposal review, and that they be involved in a strong periodic program monitoring effort. NASA is involved in the forefront of computer vision since its intended applications probably are not feasible by old technology. Yet, NASA does not have a broad enough base of imaging science within its organization. A significant part of NASA vision effort should be outside of NASA-related centers. It is recommended that hardware development work on smart sensors and image processing computers be carried out in collaboration with DOD and with broad contact with the research community. The NEEDS program represents a step toward a systems approach to providing data to users. There is a need for a program which integrates this data system with the information processing that users actually perform on the data.

3. Missions Operations Technology

This section discusses NASA's current mission operations and attempts to identify several areas in which machine intelligence can be brought to bear to increase the automation of control operations and to replace humans in time-critical, repetitive, and routine decision-making roles. A proposal to automate the mission-independent aspects of data collection and to provide a uniform facility for embedding mission-specific systems in the basic support system is reviewed.

3.1 Introduction

NASA currently builds and rebuilds mission-specific software for each mission's control. Although this state of affairs reflects the natural evolution of NASA as a large complex organization, there are indications that, without immediate and global reorganization of the mission control procedures, both NASA's science and economy will begin to suffer. Specifically, the Study Group sees a pervasive need to centralize and standardize mission operations procedures. In this regard, the Study Group sees a clear need for the development of a modular, "reusable" nucleus of mission operations software.

The scope of the standardization and centralization should include all aspects of mission control, from the lowest levels of

sequencing and monitoring to the highest levels of planning and problem solving. Current problems at the lower levels relate not so much to lack of mechanization as they do to lack of organization of the existing mechanization. Hence, cleaning up the lower levels calls for improved software development and integration techniques. On the other hand, establishing procedures and capabilities to organize and extend the effectiveness of the higher levels of mission control seems to call for the infusion of AI techniques; the goals at the higher levels would be to increase the automaticity of mission control, replacing humans in time-critical, repetitive, and routine decision-making roles.

All indications are that NASA is in immediate need of a more centralized, modular, and automated mission control center concept. This need for a reusable, centralized mission control center has already been recognized by certain groups within NASA. Des Jardains' POCCNET concept, reviewed below, provides an excellent overview of how mission operations could be cleaned up and standardized at the lower levels, providing a modular software foundation into which the specific scientific and technological needs of each mission could be grafted. At the higher levels, there are some AI methods that the Study Group feels are ready for immediate technology transfer, and others that NASA should invest in for longer term payoffs. We suggest several of the immediate and eventual payoffs from AI in mission operations below, and have included a brief survey of the state of the art in AI problem solving and programming languages.

3.2 State of the Art: Mission Operations

The view of mission operations developed by the Study Group is that there are three categories of human activity in mission control during a mission's lifetime:

1. Intimate control activities, where human intelligence and expertise seem to be demanded.

2. Mid-level intelligence problem solving tasks (real-time flight sequencing, resource scheduling, automatic conflict resolution) where humans are extensively employed because of their problem solving and modeling knowledge, but where no judgmental decisions *per se* must be made.

3. Repetitive monitoring and control activities, where enough intelligence and human intervention is required that humans are presently essential, yet where the tasks are unchallenging and wasteful of human resources.

In this subsection, we highlight what seem to be the most important aspects of mission operations from categories 2 and 3 that might be made more reliable, rapid, or economical if partially or fully automated via existing AI techniques.

Current Missions Operations. Mission operations is the control executive for a mission. As such it comprises the following specific activities:

1. Telemetry and Command – gathering the data transmitted from the payload, deframing and demultiplexing it, reconstructing the original raw telemetry frames, storing it, and transmitting commands and/or data to the payload.

2. Payload Navigation – determining actual payload orbital parameters, comparing them with nominal values, and making minor orbital corrections.

3. Monitoring – interpreting received data to ensure integrity of the craft and performing preventative and diagnostic tests and maneuvers.

4. Sequencing – devising, coding, and transmitting sequencing instructions to the craft for executing science experiments and remedying problems.

5. Data Interpretation and Display – capturing, formatting, and displaying scientific and technological data from the craft for convenient use by humans.

6. Manpower Coordination – sequencing ground-based human activities relating to the successful monitoring and science gathering of the mission; this includes such things as gathering together appropriate subsets of the scientific community for judgmental decisions, coordinating routine staffing of the monitoring facilities, locating technical experts to deal with specific problems, and so forth.

Although the Study Group saw a wide spectrum of detail across the various projects and missions within NASA, every project and mission seems to demand these core activities. Indeed, it appears that only a small fraction of a mission's cost in manpower and planning derives from the unique scientific aspects of the mission; without a doubt, the bulk of missions operations is common to all projects within NASA.

Nearly everyone in NASA seems to realize this. Yet there seems to be such great inertia from NASA's early days of rapid growth that no one seems able to initiate cross-mission technologies that would coalesce missions operations. We saw one notable exception, however; des Jardains' proposal for an automated, reusable Missions Control Center (POCCNET-RTOP #310-40-40). Des Jardains' proposal is well-conceived; but, as he points out, even the most ambitious automation

projects can profit from the use of AI technologies. Since we feel des Jardains' proposal represents a large step in the right direction, and since we have a relatively clear picture of where the incorporation of AI techniques might significantly enhance des Jardains' proposal, we first summarize his ideas; then suggest how the concept can be extended by incorporating existing AI problem solving and representation technologies.

Des Jardains' Proposal. Des Jardains' proposal concentrates primarily on the concept of a reusable missions control system which automates major portions of several of the categories of mission operations above. He thinks in terms of a "payload operations cycle" in which requests for data or science from users are queued up and scheduled by missions planning, taking into account their priorities, sequencing demands, and the current state of the craft and its sensors. The output of the missions planner is an "as-planned payload activity timeline," which, when combined with parametric data indicating the craft's current state, yields a specific sequence of commands to the craft. Results of commands yield an "as-performed time line," which reports back to the data management phase of the cycle. This phase organizes raw data collected during specific intervals, correlates them with the as-performed time line, and with original user requests, then delivers the data to the user. An intelligent data management facility would also notice missing data and unfilled user requests, and act as a sort of ombudsman for all users, following up with its own requests to mission planning to fill unmet original user requests.

Des Jardains' proposal is essentially (1) to automate the mission-independent aspects of this data collection and delivery cycle (and its implicit sequencing) and (2) to provide a uniform facility for embedding mission-specific systems in the basic support system. Since the sources of information with which the system must deal are both technically and geographically diverse, the proposal takes the form of a computer network which des Jardains calls the payload operations computing cycle (POCC) net.

As des Jardains correctly points out, such a POCC net would solve a number of NASA's current problems relating to mission cost, turnaround time, and efficiency. Currently, in the absence of a uniform, reusable facility, each mission must develop its own special purpose systems which are not reliable and which often just barely work. Users often must suffer lengthy delays, and must pay individually for computer time that relates more to NASA mission management than to the science the user derives. Original mission support teams break up and leave, taking with them much of the esoteric mission specific knowledge, making it difficult to train new staff to support the mission for the duration of its lifetime. In short, time, money, and effort are wasted by not having a central facility that serves as a large, automated backdrop of uniform computing resources useful to any specific mission.

AI Techniques: Mission Monitoring. The volumes of parametric data sent back from a craft are sampled, abstracted, and formatted for meaningful interface with human controllers. The role of a controller is to apply a knowledge of the mission's goals, the craft's capabilities and physics, and the current-phase phase of the mission in interpreting the data he sees. Our impression has been that this aspect of mission operations remains essentially unautomated, except possibly for continually improving sampling techniques, display technologies, and human interfaces. Our message to NASA is that this is an ideal area for increased automation from AI.

The key to automating this aspect of missions operations lies in the areas of symbolic modeling and representation, two of the pivotal areas of AI. Presently, the human's presence in the monitoring loop is required simply to make connections between what the human's symbolic model of the mission and craft say *should* be happening at any moment, and what is *actually* happening. In this role, the human monitor draws primarily upon his knowledge of cause-effect relationship, ones which are specific to the craft and others which are of a more generic nature. Because of what he knows about the current phase of the mission, he is able to compare the incoming parametric data with the expected conditions, supplied by his model. When anomalies arise, he can not only recognize them, but also use them in combination with his symbolic model to hypothesize the nature of the fault. He could then issue further diagnostic commands to the craft, commands that would remedy the fault, or commands to reconfigure and bypass it.

Such symbolic modeling, including representation of the craft, representation of cause-effect principles, symbolic simulation, and fault modeling and diagnosis, are favorite AI topics. Much of the best AI research has been carried out in these areas, and the Study Group feels that parts of this science are ready for transfer into NASA.

AI Techniques: Sequencing and Control. The Study Group heard reports of the agonizingly slow methods of controlling Viking. The process of conceiving, coding, verifying, and transmitting commands to the arm and science packages aboard Viking apparently took times measured in weeks, even for relatively modest operations. The Study Group appreciated the uniqueness of the first missions, and concurred that the procedures used were essential, given the importance and novelty of the Viking missions. However, as NASA proceeds with increasingly complex scientific missions, the increasing

autonomy of craft will demand far more automated sequence control regimes, both on the ground and on the craft. This appears to be another topic closely fitting current AI work.

The sequencing task appears to progress as follows. A committee of scientists convenes and decides on some immediate science goals. These are then roughly mapped onto craft capabilities, with some preliminary consideration that the goals are feasible, consistent with one another, and so forth. A team of experts is given the general goals; then produces a general sequencing plan. The general plan is progressively mapped down to the individual command level, resulting in a sequence of primitive steps to be sent to the craft. Before it is sent, however, the sequence must be verified both manually and by computer simulation to (a) meet the science goals and (b) to preserve craft integrity in all respects (electrical, mechanical, thermal, logical). After the sequence has been scrutinized, it is sent a step at a time, with very careful attention to feedback from the craft to ensure successful completion of each step before proceeding to the next. In a mission with the relatively simple arm and TV facilities of Viking, the bottlenecks seem to be the code sequence verification step and the feedback loop in which the sequence is administered to the craft. The conception of plans, and their mapping onto craft capabilities do not appear to be the bottlenecks. However, in a more complex mission *all* phases of sequencing will be bottlenecks, if attempted using the same level of control techniques found in Viking.

One of the larger areas of AI, *problem solving*, is directly relevant to all phases of mission sequencing. This is the study of the logical structure of plans, and their automatic generation for complex sequencing tasks. The Study Group is again unanimous in its opinion that AI problem solving theory is largely ready for use by NASA in complex sequencing environments, both ground-based and on semi-autonomous craft. Putting more sequencing intelligence on the craft becomes increasingly attractive as ground-craft distances increase and effective communication bandwidth decreases..

The scenario of a semi-autonomous craft with onboard problem solving intelligence and a symbolic model of its own capabilities might go as follows. Scientists decide that a sample of reddish material spotted about 15 meters away should be analyzed by science package 21. Using graphics techniques, they draw an outline around the sample on the TV image. Using this outline to identify the object of interest, the onboard vision system converts the image data to coordinate data in its local coordinate frame. The vision system issues the goal of causing a piece of the sample located at the coordinate to be transported to the input hopper of science package 21, located at another known position. The navigation problem solver then generates a course, moves the craft to within arm's distance of the sample, reaches, grasps, then verifies visually and by tactile feedback that a red mass exists in its grasper. It then plans an arm trajectory to package 21's input hopper, noting that the flap of package 13 is up, and must be avoided. After moving the sample to the hopper and ungrasping, it visually verifies that a red mass exists in the hopper, and no longer exists in the grasper. It turns on package 21, and reports back to ground.

Everything in this scenario is within the means of current or forseeable AI problem solving, manipulator, vision, and navigation technology. Its primary feature is that, because of a self-model and knowledge of problem solving strategies, the craft can do more science with less ground-based support in a given period of time. Furthermore, the advantages of such technology on any particular mission are miniscule when compared to the advantages NASA will derive from the underlying technology. Again, just as des Jardains has pointed out for the lower level aspects of mission operations, what NASA sorely needs is a mission independent repertoire of basic problem solving packages which can be molded around the automatic sequencing needs of each mission in a uniform way.

3.3 Recommendations

Up to this point, NASA has concentrated on those activities that, in a primary sense, result in successful missions. That is, NASA designs and builds the equipment required for space-related science. This includes ground-based control equipment and procedures, as well as the spacecraft and its support systems. The Study Group strongly feels it is essential that NASA begin to look at some metaissues of how to codify the knowledge it uses in primary development. AI research has shown that codification of the knowledge underlying the primary advances in a field can lead to a better understanding of the basic issues of the field. In NASA's case, the immediate and long-term payoffs from codification of existing knowledge about mission operations would be in increased automaticity, if the primary technologies underlying mission operations can then be handed over to the computer. As the computer assumes progressively more of the intelligent control functions, more ambitious missions become possible, each mission becomes cheaper, and the scientific community can be put in closer touch with the onboard science.

The Study Group's message to NASA is, therefore, that NASA is becomming more and more an information utility and less and less a space hardware enterprise. Because of this, NASA needs to begin new mission independent programs for managing information during a mission. The first step toward creating a metalevel (information-based, rather than hardware-

based) technology within NASA is the development of a unified Mission Control Center, with the goal of increasing the mechanization and standardization of sequencing, data handling and delivery, and related protocols at the low levels of the system, and increasing the automaticity of the center at the higher levels by introduction of existing AI problem solving and symbolic modeling techniques.

To begin the development of such a reusable, modular, intelligent Mission Control Center, the Study Group makes the following recommendations.

1. That NASA look seriously at des Jardains' proposal and establish a mission-independent fund for supporting the development of a system such as DesJardains proposes.

2. That NASA create a special internal, cross-mission division whose primary charge is to interact with the AI community on issues of increased automaticity, using AI techniques, throughout NASA mission operations. The division would serve as a membrane through which theoretical AI and advanced computer science could flow into NASA to meet practical mission operations needs. The division would eventually become a mission-independent resource from which the mission planners for individual missions could draw advanced control techniques for their specific goals.

3. That NASA charge the new division with constructing symbolic models of mission operation, and applying those models in the organization of an intelligent software library for use in specific missions. This library would provide basic AI technological support for automating various aspects of specific missions. It would serve much the same function as a machine shop now serves; but rather than new experimental hardware, it would draw upon advanced AI and computer science to provide mission-specific software tools, ranging from symbolic models of a spacecraft to models of the scientific uses of information derived from the craft.

4. That NASA adopt and support one of the advanced AI programming languages (and related research machinery) for use by the AI division in its role as a NASA-wide advanced technique resource and information facility.

4. Spacecraft Computer Technology

The intent of this section is to discuss computer requirements for onboard spacecraft operations in future NASA missions. Space missions have special computer needs that do not pertain in ground use of computers. The special needs and requirements that will be imposed on computers to meet scientific missions of exploratory space flights in the areas of fault tolerance, large scale integrated circuits, space qualification of computers, computer architectures, and research needed for space computers are discussed. Recommendations of actions to be taken by NASA are specified for each of these areas.

4.1 Technological Need

Computers in outer space face severe architectural constraints that do not exist with respect to ground-based computer operation. Because of this, special considerations must be taken with space computers that do not necessarily generalize from ground experience. The aspects that require special attention are discussed below.

1. **Power and weight** constraints are important for space missions. Fortunately, work in large scale integrated (LSI) technology has played a major role in decreasing power and weight requirements for computers.

2. **Hostile space environmental conditions** require that the computer be shielded from radiation, extreme temperatures, mechanical stress, and other space conditions.

Operational Requirements

1. Component reliability is essential since manual repair or maintenance in the conventional sense is not possible.

2. Autonomous operation of the computer is essential as there will be limited communications with Earth-based systems.

3. Computers must be both electronically and logically fault tolerant to:

   - Provide long operational life.
   - Provide self-contained recovery from transient and permanent faults.
   - Control automatic maintenance of the entire spacecraft. Error conditions must be readily detectable and isolated to permit recovery operations. Errors may be of two major varieties.
   - Physical faults due to component failures, temporary malfunctions, and external interference.
   - Man-made faults due to imperfections in the specifications and bugs in the program.

Scientific Needs

The scientific needs for space mission computers may vary greatly. Once a mission is approved and the science objectives are specified, it is necessary to analyze each scientific experiment to determine its needs for computation. Because of the development of microcomputer technology it is not unreasonable to place a small microcomputer into a scientific device to provide it with more intelligence. Hence, there will be a need for microprocessors.

To support devices which will be used to explore a celestial body, and which will exhibit "intelligent" behavior, large-scale computers will be necessary; that is, large, fast primary memory storage and backup storage devices will be required. Processing pictures, and developing detailed plans to permit robotic devices to operate in space so as to accomplish mission objectives given general guidance from ground, will be necessary. Large amounts of space and time are required to process real-time programs for robotics and machine intelligence.

4.2 State of the Art: Architectural Alternatives for Space Operations

The use of computers for space missions has been evolving since the start of the space age. First-generation space missions essentially had no computers. Second-generation missions had centralized computers that performed all computations required by the mission. Third-generation computers are now being considered. Three different computer architectures can be considered for space operations: distributed microcomputers, centralized processor, and distributed networks of computers. Some of the advantages and disadvantages of each approach will be explored below.

Distributed Microcomputers. If one is to have many small devices with their own built-in intelligence via a microprocessor, then a distributed microcomputer configuration is highly desirable. Such a concept has many advantages both from a technological view and a management view. A distributed network should permit any microprocessor qualified for space to be interconnected to the system. The interface between modules should be simple as the devices should be relatively independent of one another. Hence, errors can be isolated to devices, and it should simplify design problems. A simple executive routine could be developed to control the devices.

There are some virtues to a distributed microcomputer approach:

1. Changes in software sent from the ground to enhance a device need not pass through extensive reviews as the change affects only one experiment. Hence, coordination between experimenters and the various software will not, in general, be necessary.

2. Software needs to be developed primarily for small problems. The code will be short, and in most instances, will be written by one programmer. Hence, software can be verified and tested more readily than can large, complex software.

Some disadvantages of a distributed approach are:

1. Space, weight, and computer memory requirements may be larger than that for a centralized approach since memory and logic is not being shared.

2. "Intelligent devices" that have their own microprocessors cannot obtain more memory than initially planned for the space mission. There may be instances whereby information learned on the ground could cause new software to be developed for the device. However, unless the new software fits into the preplanned memory size, it will not be possible to make the change.

Centralized Processor. In a centralized processor system, all functions relative to "intelligent" devices are placed in one computing system. Devices may time-share the central processor so as to have the same effect of "intelligence" as with a distributed processor system in which the "intelligence" is built into the device with a small microprocessor. A centralized processor would have a dynamic storage allocation routine built into it to account for space required by separate programs.

Some of the virtues of a centralized processor configuration are:

1. Large, fast memories become available for complex "machine intelligence" tasks such as picture processing, high resolution, and plan formation needed to permit robotic devices to explore terrestrial bodies in space.

2. "Intelligent devices" that time-share the central processor can have their "intelligence" augmented by new software since more core memory should be readily acquired from the dynamic storage allocation routine if needed.

3. Space and weight is saved since only one control logic is required for the single computer, and memory is shared.

Some disadvantages are:

1. The executive routine for the central processor will be complicated and verification of the executive routine

will be more complex than for the distributed processor approach.

2. Changes in software made on the ground to enhance a device may require extensive coordination and testing on the ground before it can be approved and transmitted to the spacecraft.

Distributed Networks of Computers. In a distributed network of computers, tasks to be performed can be assigned to any of the computers in the network. Peripheral devices and memory in each of the processors can be shared. Many central processors permit parallel computing to take place. A virtue of such an approach is that if one central processor fails, computation can still continue since other processors can be used to perform the work, albeit at a reduced processing speed.

Some disadvantages of the approach are:

1. Complex executive routines are required to control and to transfer data between processors.

2. A considerable amount of time may be expended to simply manage the configuration than in performing work in support of the scientific mission of the flight.

Fault Tolerance. Computers sent into space must be robust. They must be able to operate in space even when malfunctions occur. Fault tolerance is an attribute of information processing systems that enables the continuation of expected system behavior after faults occur. Fault tolerance is essential to space missions as it is impossible to adequately test components of transistor-like devices on a *single* chip. A single computer would have hundreds of such chips.

Faults fall primarily into two fundamentally distinct classes:

- **Physical faults** caused by adverse natural phenomena, component failures, and external interference originating in the environment.

- **Man-made faults** caused by human errors including imperfections in specifications, design errors, implementation errors, and erroneous man/machine interactions.

Fault tolerance and fault-avoidance are complementary approaches to the fault problem. Fault avoidance attempts to attain reliable systems by:

- Acquisition of the most reliable components and their testing under various conditions.

- Use of thoroughly refined techniques for the interconnections of components and assembly of subsystems.

- Packaging and shielding of the hardware to screen out expected forms of external interference.

- Carrying out of extensive testing of the complete system prior to its use.

Fault tolerance of physical faults attempts to employ *protective redundancy*, which becomes effective when faults occur. Several redundancy techniques are:

- **Fault masking** to assure that the effect of a fault is isolated to a single module.

- **Fault detection** to detect that an error has occurred so that a recovery algorithm may be initiated.

- **Fault recovery** to correct a detected fault. Automatic recovery algorithms are essential for space flights since human intervention will not be possible.

Fault masking appears to be a good approach primarily for short missions that consist of several days duration. Both hardware and software controlled recovery systems are required for successful space operations.

Two techniques for realizing fault tolerance of man-made faults are:

- **Design faults:** prove correctness of programs and mathematical models for software reliability and prediction (both are in the research stage); "software engineering" techniques include procedures for the collection and analysis of fault data; management procedures for software development; structures programming approach to program design; and software verification and validation techniques.

- **Interaction faults** due to man/machine interaction errors. Control of such faults has been implemented primarily by operator training and maintenance manuals. Techniques used in AI could suggest approaches that would eliminate this kind of fault by screening all inputs.

LSI Technology. Large scale integrated circuit technology has yielded relatively large processors on small chips. These devices are highly important for space technology. Today's high performance MOS microprocessor has the following features:

- Architecture – 16 bit minicomputer on one chip.

- Cycle Time – 125 nanosecond operation speed.
- Power – 1.0 watt.
- Die Size – 5.25 millimeters on a side.

It is not clear, however, that such a fast device could be space certified in the near future.

Future high performance MOS microprocessors are likely to have the following features:

- Architecture – Full scale information processing system.
- Cycle Time – <100 nanoseconds. (Such a speed may not be achieved in the near future and may require an even longer time to be space qualified.)
- Power – 2-4 watts.
- Die Size – 6.5 millimeters in a side.
- Device Count – 60,000.

In addition, it would have a large logical address space, multiprocessing capability, a language orientation, and a firmware operating system.

Space-Qualified Computers. Space qualified computers appear to be lagging significantly behind ground-based computers both in speed and memory capacity. Specifications for a fault-tolerant space computer (FTSC) under development at the Raytheon Corporation are as follows:

- Operations/second – 250,000.
- Word and Memory Size – 32 bit words up to 60 K memory.
- Operations – floating point and vector operations.
- Weight – 23 kg for a 60 K memory and 36 K spares 14 g for 16 K memory and 12 K spares.
- Power – 25 watts.

The system is expected to be triply redundant, where all modules are on single chips.

4.3 Recommendations

Digital computers onboard spacecraft have been playing an ever increasing role in NASA space missions. They are destined to play a dominant role in future space missions. The miniaturization of computers that has revolutionized computers on Earth provides even greater opportunities for space missions. They will permit NASA to develop "intelligent" sensors and devices which permit information, rather than raw data to be acquired in space and be sent to Earth. Significant size computers can be developed which will permit robotic devices to be built and controlled using general plans developed on Earth. Such devices will permit the terrestrial exploration of remote bodies that cannot be explored by man.

Fault Tolerance and Hardware. Whereas the development of smaller, more powerful computers on chips will progress without support from NASA, these developments will not meet NASA needs for spacecraft. Ground computers do not require absolute fault tolerance. Because they are relatively inexpensive, chips can be replaced on the ground. This, however, is not possible onboard spacecraft, where fault-tolerance is crucial to the success of a mission. Fault-tolerant hardware systems need to be supported both by NASA and the Department of Defense who are also concerned with computers onboard spacecraft. If funding were coordinated, it could benefit both organizations. Fault tolerance must proceed at two levels – considering both hardware and software. At the current time, a major problem exists with respect to large scale integrated circuit technology. Because of their complexity, chips cannot be tested adequately now. Random logic chips (e.g., INTEL 8080) may have failure rates that are unacceptable for space use. The random logic makes it extremely difficult to test chips adequately.

A hierarchic, or top-down, approach to designing chips, rather than random design methods could increase chip reliability and permit easier testing. NASA should support efforts in hierarchic design, or other design techniques which will improve chip reliability and ease of testing. Until major developments are made by manufacturers in improving the reliability and testing of chips, NASA should plan to test its own wafers thoroughly before qualifying them for space. Testing performed by manufacturers on wafers has been, at best, poor. Planning for fault tolerant hardware must start at the inception of a space mission and must be a part of the mission management plan.

Fault Tolerance and Software. Fault tolerance is needed not only for hardware, but also for software. Because of a trivial software error, an entire space mission costing billions of dollars can be lost. By having intelligent devices with their own hardware and software, small programs, relatively easy to code, verify, and test can be developed. However, one cannot always guarantee small programs. Hence, a fault tolerant and software effort must be initiated at the inception of a mission and must be an integral part of the management plan. Software recovery procedures and algorithms to handle single

and multiple failures are required, and need considerable research. A systematic effort is needed for error detection and recovery algorithms for space computers. Fault-tolerant hardware and software for space computers is still in its infancy and needs considerable support from NASA.

Computer Architecture. There is no one computer architecture uniquely suited to all NASA's needs. The particular architecture for a specific mission will depend upon the mission objectives. The three architectures discussed in Subsection 4.2, all have advantages and disadvantages. The distributed processor concept and large central processors are useful architectures and should be considered for near and future term space missions. However, the distributed network of computers requires considerably more research to determine its applicability to space operations. Because much is still not known about the control of distributed networks on ground-based systems, this type of architecture is not realistic for a Mars 1986 flight which would include a robotic device. A distributed processor concept is attractive from a management view of space computing. It provides for separation of functions. It is particularly useful for missions on which "intelligent" devices and sensors have special timing requirements that cannot be fulfilled by a central processor.

Missions that require robotic devices will require large central processors. Because of weight and space limitations, "intelligent" devices should be reviewed carefully on such missions to determine if their needs could be met by the central processor. If this is possible, then the central processor should be shared to service the device. Trade-off studies will be needed to determine the role of the central processor and the number of "intelligent" devices that will meet the space-weight restrictions. Computers are destined to play essential roles in space missions. They will have a major impact on "intelligent" devices and sensors. Exploration of terrestrial bodies by robots can be possible only with adequate computer hardware and software. NASA must place greater stress and funds into the support of space-board computers, fault tolerant techniques and systems, and software support for future space missions.

5. Computer Systems Technology

This section addresses the use of computer technology throughout NASA. We will review the rapidly evolving state of hardware technology and describe its implications upon the practicality of machine intelligence.

5.1 Introduction

With computer technology so central to the organizations's mission, and consuming such a large percentage of its resources, one would expect to find, a massive research and development program to advance this technology and thereby further its mission objectives. Yet we have found scant evidence of NASA innovation within this field, and strong indications that it is not even adequately adopting technology developed elsewhere. As an indication of this lack of innovation, though it is certainly not conclusive evidence, at the most recent AIAA Computers in Aerospace Conference (1977) sponsored in part by NASA, only four of the eighty-six papers presented (less than 5%) were by NASA Headquarters or NASA centers people.

5.2 State of the Art: Computer Systems

In the workshop deliberations of this Study Group several trends within NASA have become quite apparent which may seriously proscribe the potential benefits available from spacecraft based machine intelligence. It is therefore important to identify these trends, uncover their basic cause, and suggest alternative cures which preserve and enhance the opportunities to utilize machine intelligence. These same trends also exist, though to a lesser extent, for ground-based systems, and hence have broad applicability throughout the agency.

NASA Missions Are Engineered and Preplanned to Minimize Dependence on Autonomous Operations. Because of NASA's no-fail philosophy for missions, an extremely conservative force is applied to mission plans and objectives. All aspects of the mission are carefully thought out in minute detail and all interactions between components meticulously accounted for. Besides increasing mission planning costs and lead time, the resulting plans are extremely inflexible and are incapable of having experimental components. As an example of this approach, the Mars rover mission reduced the need for autonomous control to local obstacle avoidance within a 30-meter path. The rest of the control was provided via ground supplied sequencing produced on an overnight basis. As a result half of the available picture bandwidth was devoted to pictures for the ground based path planning function rather than for science content, and no autonomous capability was provided to photograph and/or analyze targets of opportunity not selected in the ground-based plan. Similarly, in ground-based systems, we found evidence that investigators working in data reduction were not able to use the most advanced technology available because the NASA monitors were not convinced that it was 100% reliable. Instead, a proven, but obsolete, method requiring much more user intervention was chosen because of NASA excessive conservatism and because the concept of experimental upgrade of baseline capabilities has not been embraced. Clearly what is needed is a new mission planning model which establishes minimum baseline capabilities for mission components, enables use of enhanced versions, and provides protection from component malfunc-

tion with automatic reversion to baseline capabilities. While such redundancy is commonplace in hardware, similar software redundancy, especially utilizing enhanced "experimental" versions is quite novel, but technically feasible within a properly constituted operating system. Developing such a capability is part of our recommendations below.

Increased Use of Distributed Computations. There appears to be a strong push for correlating software function with hardware modules, so that software separation is paralleled by hardware separation. This tendency seems to be predicated on the current inability to separate and protect software modules from one another except by placing them in separate hardware units.

The cost of this practice is to preallocate computing resources on a fixed basis rather than dynamically allocate them from a common pool. This results in underutilization of the computing resource, reduced capability, and/or decreased system flexibility. Current machine intelligence systems require large allocations of resources, but they only utilize them intermittently. Since such machine intelligence systems will initially be experimental they are less likely to justify fixed allocation of the resources only occasionally required.

The benefits of increased utilization of dynamically allocated resources could be realized if protection mechanisms enforcing separation of software modules required above for "experimental" versions and a resource allocator (a standard part of operating systems) existed.

Use of Standardized Computer Hardware. As part of NASA's standardization program, standards are being set for onboard spacecraft. This is intended to reduce costs by avoiding new hardware development and space qualification efforts, decrease development time by ensuring the early availability of the hardware, and increase reutilizations of existing software. However, since hardware technology is changing faster than the mission launch rate, standardization results in the use of obsolete hardware. This limits the resources available to any machine intelligence system. Development of software portability, or equivalently hardware compatability in a family of machines, would mitigate all of these problems except for the time and cost of space qualification.

Long Lead Times Required by System Integration. Currently all software, like all hardware, must be created, debugged, and integrated many months before mission launch to ensure proper spacecraft functioning. But unlike hardware, software can be modified after launch via telemetry. This is especially important in long-life missions lasting many years. During that period, as the result of increased scientific knowledge, better software development techniques, and/or changed mission objectives, there may well be a need to modify and/or update the onboard software.

The benefits would be reduced lead time for software and an increased flexibility in mission objectives. This capability could be quite critical to early utilization of maturing machine intelligence technology. This notion is equally applicable for ground-based systems which may be utilized long after the mission launch date. They too must be capable of being upgraded, modified, and/or supplanted by experimental capabilities during mission operations. The basis for such capability is a centralized pool of computing resources with dynamic allocation and a protection mechanism for system integrity. It should be noted that this notion has already been incorporated into the Galileo mission plan (though for cost rather than flexibility reasons) in which the spacecraft software will be delivered after launch.

Desire to Minimize Onboard Software Complexity. This is part of NASA's larger effort to minimize spacecraft complexity to increase reliability. As above, special recognition of software's unique characteristics must be made. Otherwise onboard capability will be unnecessarily restricted. The minimized complexity criteria should be applied to only the baseline software and the protection mechanism, for this is the only portion of the spacecraft software relating to reliability, rather than the entire package of "enhanced" modules. With such an approach, capabilities, including experimental machine intelligence systems, could be incorporated in spacecraft without compromising reliability.

Central to each of these spacecraft trends is the notion that software is an ill understood and difficult to control phenomenon. It must therefore be managed, restricted, and carefully isolated into separate pieces. This notion has, in the past, been all too true, and its recognition has been manifest in the trends described above. However, current experience with time-sharing systems have developed techniques, combining hardware facilities and their software control, for providing separate virtual machines to several processess. Each virtual machine while protected from the others shares a dynamically allocated pool of resources (such as memory, time, and bandwidth) which may include guaranteed minimums. With such a capability, simulating separate machines via a hardware/software mechanism, all of the reliability advantages of separate machines are retained while the flexibility of dynamic resource allocation are also achieved. With proper software design these capabilities could be built into a general facility for incremental replacement of baseline modules by enhanced, and possibly experimental, versions with automatic reversion to the baseline module upon failure of the enhanced module.

5.3 Computer Systems Development Recommendations

We recommend a virtual machine and software-first approach to system development:

1. For both ground and spacecraft software, that NASA develop a virtual machine approach to software development in which protection between processes is maintained by the operating system which also allocates resources as required with certain guaranteed minimums.

2. That within such a facility provisions be made for supplanting modules with upgraded or "experimental" versions. The operation of such modules will be monitored automatically and/or manually and upon failure will be automatically replaced by the reliable baseline module.

3. That NASA adopt a "software first" approach so that hardware can be supplied as late as possible (to take advantage of the latest capabilities). To support such an approach, either software portability (for newly developed code) or compatible machine families must be provided.

6. Software Technology

This section makes recommendations concerning the use of machine intelligence to further the production and maintenance of software throughout NASA. In addition, we will strongly recommend increased utilization of (non-machine intelligence) computer science to improve NASA's current capabilities in software.

6.1 Introduction

NASA is basically an information organization. Its mission is to collect, organize, and reduce data from near- and deep-space sensors into usable scientific information. Computers are obviously essential to this mission as well as to the launch and control of the spacecraft involved. Annual computer expenses, for both hardware and software, represent about 25% (?) of NASA's total budget. Compared with other users of computer technology, such as military and commercial organizations, NASA appears to be merely a state of the art user. But compared with the programming environments found in universities and research institutes from which this Study Group personnel panel was drawn, there is a world of difference. The technology lag represented by this gap is not NASA's responsibility alone, but is indicative that an effective technology transfer mechanism does not yet exist within the computer field. NASA would do well for itself, and set a fine example, to remedy this.

There are two main issues we wish to cover in this section. The first concerns characterizing the state of software development within NASA, comparing it to the advanced software development facilities available in selected universities and research institutes, and outlining a short-term plan to effectively transfer this technology into the NASA environment. Secondly, there exists some preliminary, but far from practical machine intelligence work on automating various parts of the software development process. We will briefly examine this work and its potential for NASA; then suggest an appropriate role for NASA in this field.

6.2 State of the Art

Software Development within NASA. With rare exception, NASA software is developed in a batch environment. Often the medium is punched cards. Programs are keypunched and submitted. Results are obtained from line-printer listings hours or even days later. The only debugging information provided is what the programmer explicitly created via extra statements within the program. Deducing the cause of a failure from the debug evidence produced is a purely manual operation. Changes are made by keypunching new cards and manually merging the corrections with the program. Then the cycle is repeated. In some NASA environments, cards have been replaced by card images stored on a file and corrections are made with an online editor, but the process is essentially the same. Only the keypunching and manual manipulation of cards has been supplanted. The programs are still developed and debugged in a batch mode.

Software Development in Machine Intelligence Laboratories. In striking contrast to the NASA program development environment is that existing at several laboratories (such as CMU, MIT, Stanford, BBN, ISI, SRI, and Xerox) working on machine intelligence. This environment is characterized by being totally online and interactive. The heart of this environment is a fully compatible interpreter and compiler and an editor specifically designed for the language and this interactive environment. The remarkable thing is that this environment is based neither on machine intelligence mechanisms nor concepts, but rather on a machine intelligence philosophical commitment to flexibility and a few key computer science ideas (that programs can be manipulated as normal data structures and that all the mechanisms of the language and system must be accessible so that they too can be manipulated).

These key ideas and a long development by many talented people, have created an unrivaled software development

environment. In it, changes to programs are automatically marked on reformatted listings, the author and date of the changes are recorded, the correspondence between source and object modules is maintained, automatic instrumentation is available, non-existent code is easily simulated for system mock-up, extensive debugging and tracing facilities exist including interactively changing the program's data and restarting it from any active module. In addition, arbitrary code can be added to the interface between any two modules to monitor their actions, check for exceptional conditions, or quickly alter system behavior. Also an analysis capability exists to determine, via natural language, which modules call a given one, use a variable, or set its value.

There are many other capabilities far too numerous to mention but the key issues are that they are fully integrated into an interactive environment and that a commitment has been made to substitute computer processing for many of the programmers' manual activities. As computer power becomes less and less expensive (By 1985, according to a CCIP-85 study, hardware will represent only 15% of the cost of a computer system. The rest will be cost of people producing the software.) while people get more expensive, such a policy must clearly predominate. Furthermore, several studies have shown that software quality and cost improve as the number of people involved decreases. Thus, environments which improve programmer productively by automating certain functions also improve quality while reducing costs.

6.3 Software Development Recommendations

For these reasons, we recommend:

1. That NASA immediately undertake a program to recreate within NASA the interactive programming environment found in various machine intelligence laboratories for some NASA language.

2. That NASA consider creating a modern data-encapsulation language (of the DODI variety) as the basis for this interactive facility.

3. That NASA only undertake this project with close cooperation of an advisory group drawn from these laboratories and with NASA personnel familiarized with these interactive environments via extended onsite training visits (approximately 6 months duration)

4. That NASA acquire the necessary computer hardware to support such an environment.

Automatic Programming. Having dealt with the current state of NASA's software production and its improvement through utilization of existing computer science technology, the central issue of utilizing machine intelligence for software production can now be addressed.

Software is essential to NASA's mission. It is used to launch and control spacecraft, to collect, reduce, analyze, and disseminate data, and to simulate, plan, and direct mission operations. The other sections of this report address extension of these capabilities through incorporation of machine intelligence in these software systems. Here, holding the functionality of the software constant, the use of machine intelligence to produce, or help produce, the software is considered.

Even with the capabilities suggested above for improving the production and utilization of software, the development of software is still largely a manual process. Various tools have been created to analyze, test, and debug existing programs, but almost no tools exist which aid the design and implementation processes. The only available capabilities are computer languages which attempt to simplify the statement of a finished design or implementation. The formulation of these finished products is addressed only by a set of management guidelines. As one can imagine, these manual processes with only minimal guidelines, unevenly followed, are largely responsible for the variability currently found in the quality, efficiency, cost, and development time of software.

It is quite clear that significant improvements will not occur as the result of yet "better" design and implementation languages or "better" guidelines, but only by introducing computer tools which break these processes down into smaller steps, each of which is worked on separately and whose consistency with each other is ensured by the computer tool.

This approach defines the field of automatic programming. It is based on machine intelligence technology and, like other machine intelligence systems, it is domain specific. Here the domain is the knowledge of programming: how programs fit together, what constraints they must satisfy, how they are optimized, how they are described, etc. Programming knowledge is embedded within a computer tool which utilizes the knowledge to automate some of the steps which would otherwise have to be manually performed. There is considerable diversity of opinion over the division between manual and automated tasks.

The critical issues, however, are that the unmanaged manual processes of design and implementation which currently exist only in people's heads and, hence are unavailable and unexaminable, have been replaced by a series of smaller explicit steps, each of which is recorded, and that some

portion of them have been automated. Over time, more and more of these steps will be automated and the programmer's role will become more supervisory. For the first time, the programming process will have been rationalized and recorded, open for examination and analysis. This will enable programs to be produced which are guaranteed to be consistent with their specification. It will eliminate the need for program testing and the cost and unreliability associated with undiscovered bugs. In addition, as automation increases, costs and effort will plummet. Besides the obvious advantages these reductions offer, a very important side benefit will occur. We know from instrumentation studies that large systems are not efficient when first implemented. Unanticipated bottlenecks always occur. The drastically lower costs of implementation will afford the opportunity for people to experiment with alternative implementations. These experiments wil broaden their experience base and enable them to develop better intuitions about how such implementation should be constructed. Furthermore, once people have gained this knowledge, it can be incorporated as a further automation of the programming process.

All of this paints a very rosy picture about automatic programming. The catch, of course, is that these capabilities don't yet exist. The field is in a quite formulative stage. Impressive work is being done in a number of research labs, but none of these systems is close to practical use by an external user community. A period of research support followed by specialization to particular applications is needed if NASA is to reap any of these potential benefits. Since each of NASA's missions require similar, but different, software, a number of such specialized automatic programming systems could be constructed to cover a large percentage of NASA's total software effort. Recommendations:

1. That NASA develop a research and development plan, in conjunction with experts in automatic programming, for the creation of automated tools for the design and implementation stages of the software development process.

2. That NASA identify its major areas of software concentration and that specialized AP systems be developed for these as the field matures.

7. Data Management Systems Technology

This section briefly outlines a proposal for a coherent data management system which would control data acquisition, reduction, analysis, and dissemination. We discuss NASA's *NEEDS* effort highlighting those areas where machine intelligence techniques may be brought to bear. We propose a greater emphasis on intelligent sensors to perform data reduction and selective data transmission, and the development of knowledge data bases to aid in experimentation and planning.

7.1 Introduction

Current and future planned missions within NASA are oriented heavily towards the acquisition, dissemination, and analysis of data transmitted from space. The amount of such data is currently voluminous and will become larger by an order of magnitude in the 1980s. An estimate of the problem in the 1980s indicates that some 10^{10} bits of data/day will be generated for non-imaging data, while some 10^{12} bits/day will be generated for imaging data. The magnitude of the data acquisition and dissemination problem is staggering. When one adds the increased sophistication in data processing needed to convert raw data to information and to make it accessible to the users one has a major problem in managing such data.

The present NASA data management system has evolved in an *ad hoc* manner. Continuation of an *ad hoc* approach will neither be resource-effective nor meet the needs of the scientific user community for the post-1980 time frame. Greater reliance must be placed upon computers playing a greater role in space. The heavy density of data, instrument sophistication, and miniaturized microprocessors in space mandate that resource effectiveness be achieved on and between missions by end-to-end management of data. This will involve policy, management, software, and hardware. It is extremely important to have careful planning or central management planning for data. To achieve resource-effectiveness, the management of data must become a controlling force in the development and plans for any mission. In the following sections, we shall briefly describe the flow of data as it exists now, and the end-to-end data management concept that will be necessary to meet the demands of the 1980 era and beyond. We shall also discuss the steps required by NASA to meet the major challenge of the data management problem.

7.2 State of the Art: Flow of Data Within Missions

The flow of data from an instrument to a principal investigator in today's technology goes from the instrument onboard to data processing on the ground and then is transmitted to a principal investigator or to facility instrument team members.

Future missions will require that, instead of a one-instrument to one- or many-instrument users, it will be necessary to have the outputs from many instruments onboard

the spacecraft undergo data processing and provide outputs for many users. For example, weather and climate information, spacecraft thematic data, and hydrological data obtained from many instruments are combined with data obtained through non-space observations to prepare food and fiber production forecasts.

Current Data Control. The management of data as it is obtained from the spacecraft is currently provided by onboard control management. They specify the data to be sensed, conditioned, handled, and transmitted by the instruments on the spacecraft. They have available to them flight direction data, and can make adjustments during the flight. Investigators who desire changes, must negotiate with the management team. Data obtained from a mission must undergo processing, sorting, and distribution. Further reduction, extraction, and analysis of the data takes place to transform the data into useful information. The transformed data and the raw data are stored in central repositories and distributed to principal investigators for further processing, analysis, and use.

This flow of data is illustrated by the LANDSAT project. LANDSAT data is currently transmitted to line-of-sight ground stations located at Beltsville, Sioux Falls, and Goldstone in the United States and in three foreign countries. The data is now in the planning stages to be transmitted to several other foreign ground stations. It is then retransmitted over the Space Tracking Data Network (STDN) or mailed to the Goddard Space Flight Center. In either case a three or four day delay results in the transmission receipt at Goddard.

The raw data is assembled at Goddard where it must be spooled-up waiting for other data related to the flight, such as orbital information and fine attitude of the spacecraft. Some processing is performed on the data to account for such factors as the curvature of the Earth. Goddard then cuts a tape and transmits the processed data to the EROS Data Center run by the Department of the Interior in Sioux Falls. EROS catalogs the data, stores it in its data base and distributes data to users on a payment basis. Upon request, EROS produces high quality enhanced data. However, no LANDSAT data conforms to any particular map.

The LANDSAT data system for the U. S. should experience considerable improvement when a Master Data Processor (MDP) becomes available at Goddard. Such a MDP will provide space oblique mercator projections analogous to those obtained from ortho-photo aircraft images. Furthermore it is able to use selected ground control points for each frame to permit sequential frame overlays from several passes over a particular area. The master data processor can solve the problem of making the digital images look right, and can provide temporal registration. However, the MDP is limited in that the images are not keyed to a specific map projection.

Future Control: NASA End-to-End Data System (NEEDS). Projected mission data requirements exceed the present system capabilities to handle them. The increase in data volume can only partially be met through engineering technology improvements, as there promises to be a concomitant increase both in the number of users and complexity of sensor-related tasks. New demands continually arise for more complex instruments, better interfaces between instruments, and more sophisticated data processing. Many experiments and applications tasks in the near future will require a direct user/sensor coupling on a non-interference basis. This should require the development of dedicated, distributed microprocessors on board the spacecraft. Other applications in space will require large centralized processing on the ground to effectively integrate information provided by several satellites. For both instances, data management adminstration prior to launch of each mission is needed to assure coordinated information acquisition and integration.

An end-to-end data system will consist of the following elements:

1. **Instruments Aboard Spacecraft** – which sense data, provide attitude and position information, Greenwich mean time, and provide control information on the downlink to ground. They are provided uplink or control feedback to specify information to the sensors as to where to look, when to look, and how to look. Such control may emanate from mission operations or directly from users who can access the sensor remotely from terminals.

2. **Operations** – monitors system performance, develops housekeeping information, coordinates activities, maintains a directory of the system, provides user assurance, and accounting information of the downlink. On the uplink to the spacecraft operations allocates resources, modifies the real time configuration, specifies flight maneuvers, coordinates user needs to the spacecraft, and provides housekeeping and repair for the spacecraft. A data base is maintained and updated on space flight information.

3. **Data Staging** – receives data from operations on the downlink through commercial networks and packages the data by operating on the output of many instruments required by a user. The output of instruments may require calibration, and the packaged data must be distributed to multiple users. Such information must be amassed in short periods of time (near real-time) to permit the user the ability to control and change

instrument settings and programs. Data staging is a downlink operation. A data base is maintained at the data staging area.

4. **Distributed Users** – provides near real-time data and investigates and screens output from instruments on the downlink. The data may be received directly from operations via commercial networks or be transmitted via commercial lines from the data staging area. On the uplink the users provide planning, scheduling, and control information directly to sensors that they control and which are independent of other instruments onboard the spacecraft. The user maintains a specialized data base.

5. **Knowledge Centers** – maintains data and knowledge on specialized topics. The data from a mission is transmitted via commercial networks to one or more knowledge centers. The knowledge centers provide services to the distributed user community. They maintain not only mission supplied data, but data from other sources. A knowledge center concerning weather data would maintain temperature, barometric pressure, and other weather data obtained by ground observation and measurement. The knowledge centers maintain archival records as well as data records. Knowledge centers will be linked together through commercial networks so that they may access one another. Users may access data in the knowledge centers through their remote terminals, and may thus perform operations on the data either at their own facility, or through the knowledge center facilities.

Data Onboard the Spacecraft. Decisions must be made concerning the management of data within the spacecraft itself. These decisions will be a function of the particular mission, and whether or not there is ready access or interaction required with the user. For example, on a robotics application on Mars, because of the distance involved and the attendant time lag, it will not be possible to direct the robot from the ground, except to provide it with general goals to be achieved. This will require that the robot contain a large amount of data and information to permit it to maneuver on Mars. It will have to have the following, as a minimum:

1. Information as to the location, size, and composition of objects.

2. A model of the terrain.

3. General rules about the relationships between objects.

Item 1 can be supplied partially from ground analysis of images and by measurements taken by the robot itself. Item 2 can also be obtained partly on the ground and partly by the robot. General rules and axioms, on the other hand, needed to devise and effect plans, must be provided by the ground. Regardless of where the data and information arises, it is essential that capabilities exist within the robot to store, retrieve, and manipulate large amounts of data. Hence, a *data management system* will be necessary as a part of the robot itself. A non-conventional data management system will be required–one that can do deductive searching and plan formation. Hence, it will require an extensive knowledge base system, a semantic network, and an inference mechanism.

Even if the spacecraft is to be used to transmit data to Earth, where a data management system exists, there is considerable planning that can be accomplished so as to improve the efficiency of data acquisition. For example, the following topics should be addressed on every mission:

- Need for data compression to minimize the number of bits transferred and to conserve communication channels.

- Data needed by the investigator as obtained by his instrument and other instruments. For example, if an image is to be transmitted to Earth, related data needed by the user should be transmitted at the same time. If, for example, the spacecraft is orbiting Earth, the precise location of the space vehicle, the angle of the Sun, and the angle of the camera tilt would be required as a minimum. Incorporating such data will save considerable effort on the ground by a small amount of processing in space.

- Use of data by the scientist. Knowing the use to which the scientist will make of the data can determine the needs for archival data, and whether or not processing of the data either on the spacecraft, at mission control, or at a central repository would be of help to the scientist.

Data management planning for the spacecraft is, therefore, one important element of a data management plan for a mission.

Operations. Operations plays a central role on each mission. Hence, the data management system requires careful attention here. Design of the command system to the spacecraft must include consideration of integrity constraints. Such constraints assure that commands given to the spacecraft are legal ones, and that inevitable ground errors are minimized. Users who have sensor control on the spacecraft should have their commands passed via commercial digital networks to the operations group where the user command may be overridden if deemed necessary by operations, and if not overridden, the

command is scheduled for transmittal to the spacecraft, logged, and the user notified automatically as to when the command is to be transmitted. The delay between user transmittal and override should be in the order of a few minutes at most as user/sensor control should take place automatically only when the sensor is independent of other sensors onboard the spacecraft.

Operations will require a sophisticated message switching system to determine where space data is to be transmitted. It must also have a data base management system to be able to store and retrieve data retrieved from a mission. The amount of data for storage and retrieval at the operations facility will depend on the mission. Data for a few days receipt can be maintained while data of more than a few days can be retained at knowledge centers and retrieved through the network as needed.

Data Staging. Data staging provides many-instruments to many-user capabilities. Data from many instruments are transmitted via commercial lines to the data staging area. The data undergoes data processing to transform it into usable modules for many users remotely connected to the data staging area. Capabilities should be provided to allow user access to raw and processed data. The users should be able to specify to the data staging area the operations to be performed on the data. The results can be placed into operational data sets consisting of processed data. All users should have access to all operational data sets. It will not be unusual for the many users to want common data. Making available the operational data sets to all users could save duplication of processing.

The management of the data staging facility should assure that the same function is not applied many times and should recognize common requests by multi-users for the same processing functions. The data staging area should transmit processed data not only to users on an automatic basis, but to the knowledge centers for archival purposes. Data staging may be viewed as a switching and processing center. Its primary function is to operate upon spacecraft data and transmit the results to the user. It will have to maintain data base directory services, as required by the user.

Users. The purpose of a mission will be to supply instrument information to the user population. The data staging area and operations can supply processed data and raw data to the user. Neither operations nor data staging can be expected to perform all the processing required by the user. Users will require their own processors and a means for storing and retrieving large quantities of data. One would not anticipate that users would require their own archival system as such a function can be provided by the knowledge centers.

The range of needs for the user cannot be anticipated in advance with respect to data processing functions. However, some users will require conventional data base management systems. In the former, there should be a major effort to standardize the data base management systems so that each user does not build or buy his own system. User proposals should be scrutinized carefully by a data base management system organization. Knowledge base systems will become prevalent in the 1980s. One can incorporate a knowledge base capability with a conventional data base management system (primarily relational data base systems), or build a special purpose system using one of the artificial intelligence languages. Such systems will be needed to do image analysis and picture processing, and to extract new data relations from given relations. Knowledge base systems can be general, but will require specific details based on the particular application. For instance, knowledge base systems could contain many of the features required for robots to accomplish their jobs on remote planets.

Knowledge Base Centers. Knowledge base centers will require three different types of data base management systems:

1. Archival.
2. Generalized Data Base Management System.
3. Knowledge Base System.

Knowledge base centers should contain archived data relating to missions and related data. They must be able to retrieve requests and transmit responses to users who can access the knowledge centers through remote terminals. Requests can be specific – such as to retrieve the tape for the fifth orbit of data transmitted on a certain mission. They can be general, such as to send all tapes where the infrared reading was between certain limits on a mission. Thus, knowledge centers will have to maintain indexes of the data, and must perform some classification of data as it is stored in the system.

Knowledge centers should also contain conventional data base management systems used to store and retrieve data conveniently contained in records. Whereas user data base management systems should fit on minicomputers, large scale computers and sophisticated data base management systems will be required. A distributed network of data base management systems should be investigated to iterate the various knowledge centers.

Sophisticated knowledge base systems which contain specific facts, general rules, and world models (e.g., a map containing roads that will be used as a template to match

against images and be used to detect roads in images) will be required. By a general rule is meant a statement of the type, "If object 1 is to the left of object 2 and object 2 is to the left of object 3, then object 1 is to the left of object 3." Complex knowledge base systems may also have to interface with sophisticated mathematical models. The area of knowledge base data systems is important and will require additional research support.

7.3 Opportunities and Recommendations

The NEEDS effort provides the potential for improving NASA use of data. Part of the improvement can come about by developing intelligent sensors and digital computer systems onboard spacecraft. Another part of the improvement can come about by developing an efficient ground communication/data processing system.

Intelligent sensors and digital computer systems onboard spacecraft can:

- Send back processed, rather than raw data.
- Obviate the need for documenting data on the ground as attitude, Greenwich mean time, and other information can be sent back to Earth with the sensor data as it is collected.
- Decrease the data flow as only relevant data need be returned to Earth (for example, if image data of Earth is to be sent and the scene is obstructed by cloud cover, the space computer should detect this occurrence, and not send the useless cloudy image).
- Respond to changes as to what should be collected as received in commands from the users.
- Compress data so that needless or redundant information does not overload the communications channel.
- Allow direct user-remote control.

An efficient ground communication/data processing system can:

- Permit near real time processing of data.
- Provide enhanced user services.
- Transmit processed multisensor data to users.
- Retrieve archival data more readily.

It is estimated that substantially less than 10% of all data received from space is ever used. By decreasing the amount of useless data by introducing intelligent sensors, and by providing better data management facilities to store, retrieve, and manage real-time and archival data, substantially greater use of data may be anticipated.

Although the NEEDS effort could yield considerable benefits for NASA, the efforts being conducted do not appear to be promising. NASA is taking a bottom-up approach to NEEDS. That is, rather than developing a comprehensive systems engineering approach to achieving such a system, a piecemeal approach is being taken. Various technologies are being investigated in an attempt to develop NEEDS. Although new technologies are clearly necessary, there is scant investigation into how they are to be brought to bear in a final system. To achieve an end-to-end data system that will provide users greater control over sensors, and will enhance the acquisition and dissemination of information from space, requires a systems approach in addition to a technology approach. The work will require significant planning at the management level, sophisticated software developments, and matching hardware capabilities. At the present time there appears to be no appreciation of the complexity of the NEEDS effort and the importance of engineering an entire system. Not only are intelligent sensors and reliable microprocessors needed in space but the management and flow of data from the spacecraft to the users and archival stores is essential. The following are some specific recommendations.

Management Recommendations

Data Management Plan Coordination Group. A centralized group within NASA consisting of computer scientists will be necessary to provide overall plans for managing mission data. The group is needed to assure coordination between missions, to minimize duplication, and to determine the general tools that should be provided to user scientists by NASA. They should be concerned with assuring that there is a cost-effective, appropriate data management plan on all missions. They should further be concerned with the acquisition and development of equipment and software.

Mission Data Management Plan and Coordination Group. The mission-oriented group should provide plans as to how end-to-end data management will be achieved on a mission. They should be concerned with how to integrate the mission objectives with current and planned data management systems within NASA. The group should also consist of computer scientists and should review all data management plans with the centralized group.

Technical Recommendations

NASA End-to-End Data Management System. The NASA end-to-end management system outlined in this report must undergo considerable planning and detail before it can become a reality. A systems engineering group consisting of computer scientists and hardware experts is needed now to achieve an effective system concept and implementation.

Knowledge Centers. The concept of knowledge centers must be explored carefully from a technical level to determine how they are to be achieved. Careful consideration will be required to develop or obtain appropriate archival and data base management systems. Insufficient attention is being placed on this aspect of NEEDS.

Knowledge Base Systems. Research support is required for work in this area. Emphasis should be placed on enhancing relational data base systems so that they can be used in conjunction with problem solving systems needed to achieve knowledge base system capabilities. A knowledge base system can be achieved, and is necessary for NEEDS. Again, insufficient attention is being placed on such systems.

8. Man-Machine Systems Technology

This section deals with the three major components of any advanced man-machine system: modeling human control processes, the design of interfaces between human and intelligent computer systems, and the design of the manipulators themselves.

8.1 Introduction

The major deficiency in the application of machine intelligence and robotics to the special problems of NASA is a lack of knowledge about how to design effective man-machine systems. This deficiency is fundamental and is based on a lack of knowledge of human processes of machine control and of the interface. For example, teleoperators are understood neither at the level of the human control processes nor at the level of determining an appropriate design of manipulation and the proper design of the information interface between human and teleoperators.

There does exist considerable knowledge about each of the fields that contribute to their problems. Cognitive psychology has developed considerable expertise and knowledge about the structures of human information processing, most especially those of perception, language, and memory. Workers in control theory have developed sophisticated procedures. Human-computer interaction has been widely studied. The weaknesses lie in the lack of useful predictive models of the interaction of these components.

Despite the fact that human-machine interaction is critical to the success of almost all of NASA's missions, NASA's present organizational structure does not appear to accommodate research on man-machine systems or man-computer interaction required. NASA Life Science programs (under Office of Space Science) have concentrated on medical, physiological, botanical, bacteriological, and biochemical disciplines. OAST has sponsored some man-machine research, but primarily as related to aeronautics. The NASA centers have done more man-machine research on an ad hoc basis, as required by the project offices. Thus, human information processing, man-computer cooperation, and man-machine control basic research has tended to "fall between the cracks."

The result is that in current and planned missions there is confusion about what mechanisms are most appropriate for communication in control and feedback of information to human operators, and what constitute appropriate tasks for humans and for machines. So far, the ambiguous status of the human-machine systems research has not led to any grave difficulties, primarily because the conservative engineering philosophy of NASA helps avoid major difficulties. The lack does severely limit applications, however.

Fundamental improvements in human-machine interaction will come about only if NASA leads the way, supporting basic research directed at the fundamental problems, developing applied laboratories, developing new conceptualizations and new techniques. This work must be mission independent, developed from broadly based fundamental research and development programs that are not subject to the complex limitations posed by mission-oriented studies. There must be better means for life science and technology organizational components to interact in bringing a more rigorous focus on these crucial long-range research problems.

8.2 Human Information Processing

Human information processing is the study of the psychological mechanisms underlying mental functioning. Memory, problem solving, language, perception, thinking – these are some of the major areas studied. In the past decade there have been sufficient systematic advances in our knowledge that these areas now constitute perhaps the best understood problems in contemporary psychology. Studies of attention are of special importance to problems faced by NASA. Humans have limited mental resources, and the deployment of these resources constitutes an important part of behavior. The limitation appears to apply primarily to conscious control.

Tasks that require conscious decision-making or control can suffer in times of stress or when other tasks must be performed or thought about simultaneously. When several tasks simultaneously demand a share of conscious resources, deterioration of performance results. Tasks that are learned well enough that they appear "automated" seem to suffer little as a result of other activity.

Despite the relative amount of knowledge about human processing mechanisms and control structures, we know surprisingly little about aspects that are relevant to the problems faced by NASA. We do not know enough about the nature of conscious and subconscious-control mechanisms. We do not know enough about the various modes of operation of the human. We know very little about the human's ability to interact with and control the environment. Almost all our knowledge deals with the processing of arriving information, or the operation of the human as an element of a control structure. This leaves unanswered much of importance. The human has two cortical hemispheres, each one appearing to be specialized for different types of processing. One hemisphere appears to be serial, the other more parallel or distributed. We have just begun to explore the implications of these differences; exactly how they apply to control issues is not understood, although there are obvious implications.

This area of knowledge about the human has many potential applications for NASA. On the spacecraft, at mission control, onsite during a mission, all these situations require different aspects of human capability. We find no evidence that NASA is engaged in systematic study of these issues. Yet they are critical to the success of NASA's missions, the more so as missions become longer, more complex, with space repair, manufacture, and mining as possible tasks.

8.3 Human-Machine Control Processes

Whether humans are physically present in space or not, their intelligent control and supervision are fundamental requirements for any space mission within this century. Their sensory, cognitive, and motor-control systems can play useful roles in combination with machine intelligence. The combination of human and computer has potential for more capability and reliability than either by itself. But realization of this synthesis requires much progress at the level of basic research and application.

A primitive discipline and art of human-machine systems does now exist, although it is unevenly developed. There are a small number of texts and several scientific journals. There are regular workshops and annual meetings. In such areas as the performance of the pilot and air traffic controller in aviation, the work has been sufficiently successful that aircraft manufacturers and government regulators determine their hardware and procedures to a significant degree from research findings. But the most sophisticated of this research and the most successful empirical applications have been devoted to situation, where the human is in continuous control of the system. As the computer becomes more capable of intelligent operation, the human operator becomes more like a "manager" or a "supervisor" than an active in-the-loop controller. This new "supervisory control" mode of operation is not well understood. Ames and Langley both have in-house and university research programs to study these new roles in the context of aviation. This is a start, but more needs to be done, especially directed towards the particular problems faced by space programs.

8.4 Human Interactions For Ground-Based Missions

A large fraction of NASA's budget is spent on an extensive system for monitoring, controlling, and processing data from a large number of Earth-orbital and deep-space vehicles. The Study Group severely questions much of this work. There has not been sufficient research done on the proper balance between human and machine judgment and analysis, not proper studies of the appropriate balance between decisions made in space and on the ground at mission control. While the new color-graphic computer based displays offer much possibility, their appropriate display formats are not well understood. The role of intelligent programs is significantly underestimated.

8.5 Teleoperators

Ever since the development of remote manipulators for nuclear applications in the early 1950s, it has been clear that teleoperators can be used to extend the human's vision, mobility, and manipulation capability into space, undersea, and difficult environments. Nonetheless, compared to the magnitude of the potential saving over sending man into space to sense and manipulate, there has been little developmental work on the control factors of teleoperators. (In similar fashion, there has been surprisingly little development of the manipulators themselves, but this is covered in a different part of this report.) When continuous remote control is not possible because of signal transmission time delays, restrictive "move and wait" strategies are required for remote operation. This can lend to awkwardness and instability, especially when force or touch feedback is used. It is clear that to perform large space construction tasks or planetary mining, etc., in other than near-Earth orbit, this type of control is intolerable.

The Study Group finds that there has been surprisingly little advancement in manipulator development since the 1960s, though recently some significant theoretical contributions to kinematics and control of manipulators are evident in both U.S. and Soviet literature. Current manipulators do not have the reach, precision, or sense ability required for space assembly. End effectors are awkward and tool changing is slow. Even use of the human operator in a supervisory mode (with a computer doing much of the control), requires close contact with the task via television and force-reflecting sensing mechanisms. Manipulation dexterity must be understood better at a fundamental level.

Thus, assuming semiautomatic assembly, where the human plays a supervisory or decision-making role, two things are needed: development of intelligent computer control systems, and the understanding of the role of the human in this mode of operation. There has been insufficient research to understand the proper interface that should exist when the human plays this higher-order role in the feedback system.

8.6 Spin-Offs

NASA sponsored research dealing with capabilities of the human-machine interaction are bound to lead to important spin-offs. Increased understanding of sensory and control mechanisms will be important for the development of sensory prostheses, for development of new systems to aid in the control and management of complex tasks, and perhaps systems capable of expanding our cognitive abilities by cooperative interaction with machines.

Research on man-machine questions should have direct application to various non–NASA needs such as unmanned mining, deep ocean exploration and construction, disposal of nuclear waste, and intracorporeal surgery performed remotely using optical fiber bundle devices. New techniques of industrial automation are a possible outgrowth of such research.

8.7 Recommendations

NASA must reassess the role of the human in the control of sophisticated systems. Low level, detailed control will probably best be done by intelligent computers, giving the humans higher level, more complex decision making and administrative responsibilities.

NASA should develop a strong research program on human information processing, on man-machine control processes, and on the interface issue. There is special need for study of situations with high data rates, with a need for rapid decisions, and with stress. There should be direct interrelationships between NASA's mission needs and the research programs. Formal ties with the university and industrial research communities would be useful, with collaborative research being potentially of great value. A scientific visitors' program could educate NASA scientists to existing research; young university personnel would become educated about NASA's particular problems.

Note that many of the problems faced by NASA occur in rich, complex environments. University research laboratories are unlikely to have the facilities to simulate these conditions. Accordingly, if NASA laboratories were made available to researchers from university settings, with sufficient time and resources, it might be possible simultaneously to increase the level of basic understanding of problems dealing with man-machine interface, and also to get direct results relevant to NASA. Thus, experiments on simultaneous attention, or on performance under stress, or on the control of teleoperators, using NASA simulators can be expected to make it possible for new kinds of phenomena to be studied. NASA has the facilities, but has not used them for general development. Universities have the technical expertise, but lack the facilities to do research relevant to the real needs of NASA.

In addition, the Study Group recommends:

1. That research in the areas of man-computer cooperation and man-machine communication and control be accelerated, in view of the long-range critical need for basic understanding of these problems. This is in lieu of supporting such research primarily on an ad hoc and mission-oriented basis.

2. That NASA organizational entities representing life sciences and the technological disciplines of computers and control develop better cooperative mechanisms and more coherent research programs in man-machine interaction to avoid the "falling between the cracks" problem.

3. That future NASA missions realize the full potential of teleoperators by developing improved means for "supervisory control" of robotic teleoperators. Thus the human operator on Earth can benefit from what the teleoperator senses, and can intermittently reprogram its computer as necessary. In this way the advantages of human intelligence in space may be had without the costs and risks of bodily presence there.

9. Digital Communication Technology

There is an aspect of NASA activity that did not receive much attention at any of the workshop meetings. This involves the transfer of information among a complex, geographically and institutionally disparate set of groups that need to exchange messages, ideas, requirements, documents, to keep informed, plan activities, and arrive at decisions quickly. The clerical infrastructure to support this network of activity, not counting the information users and generators, managers, engineers, and scientists at the nodes, must account for approximately 15% of NASA's budget for both inhouse and contractor personnel. If total personnel costs are 2/3 of the total budget, then the total costs for the mechanics of this information exchange is several hundred million dollars per year. A computer-based communication system can make significant improvements in the quality of information transfer, and probably increase the productivity of the information exchange infrastructure.

The implementation of such a system would not be predicated on new developments in artificial intelligence. It would use tools that are common practice at AI nodes of the ARPA network and are part of the developing technology of digital information and word processing. Once such a development were carried out, it would provide the data base that could take advantage of sophisticated techniques of information retrieval, semantic search, and decision making as they became available. Costs can be estimated accurately from systems at those AI sites on the ARPA network.

The functions to be carried out are the Directory and File management facilities described in the TENEX Executive manual and programs like SENDMESSAGE, MESSAGE, TV EDIT, and BULLETIN BOARD. These would operate interactively. Programs like SPELL and PUB would be offered. Teleconferencing facilities and input of documents with OCR devices could be implemented. These services offer mail to distribution lists, creation and editing of documents, spelling checking, and publication formatting, with book quality print, arbitrary fonts and graphics, with quick turnaround. Documents would be instantly available for online access and continuous updating.

In addition to the cost savings, which would probably be large, there would be the following:

- On-line documentation and record-keeping system, with computer readable documentation, instant availability to updated versions, quick copies of documents, using hardcopy devices.

- Document preparation services including editors, spelling checking, publishing and formatting programs (e.g., PUB); with arbitrary fonts and book quality text, with short turnaround time.

- Benefiting from on-line access to catalogs of images, data bases, and images; communication and sharing of algorithms and evaluation of algorithms would be enhanced.

Section VI
Conclusions and Recommendations

We believe that NASA should institute a vigorous and long-range program to incorporate and keep pace with state-of-the-art developments in computer technology, both in its spaceborne and its ground-based computer systems; and to ensure that advances, tailored to NASA's mission, continue to be made in machine intelligence and robotics. Such advances will not occur of their own accord. Many NASA requirements in computer architecture and subsystem design will in turn have a stimulating effect on the American computer and microprocessor industry, which now faces an extremely strong challenge by foreign competition. We believe that an agency such as NASA, which is devoted to the sophisticated acquisition and analysis of data, must play a much more vigorous role in the design and acquisition of data processing systems than has been its practice in the past.

These findings are supported by the recommendations independently arrived at by the Space Science Board of the National Academy of Sciences[7]:

> From experience with *mission operations* on previous space missions, we anticipate that there will be *even greater demands on data acquisition, processing, and storage;* on mission coordination; and on interaction with the spacecraft and scientific experiments. The complex nature of mission operations and the long time scale required to prepare, certify, and transmit routine commands in previous missions indicates that substantial changes will be necessary. We believe that significant technical and managerial advances must be made in anticipation of future planetary missions, in order to provide reliable, more efficient, and lower cost systems for operation of the spacecraft and scientific instruments.
>
> The testing of these systems on the ground as operational units including the participation of science teams should be carried out well before the mission. These tests should include the operation with possible failure modes. *These approaches will be more important in the future when extensive coordination must be obtained by use of more intelligent or autonomous control systems.* The choice of onboard preprocessing versus earth-based processing and the utility of block telemetry formatting and distributive data handling and control subsystems will require assessment. In the past, computing facilities and command and data-processing software were not always efficient, and early attention was not given to overall system design in laying out missions. Further, experience with past and current spaceflight missions has shown that complicated systems with higher levels of intelligence are difficult to handle without substantial experience.
>
> We are apprehensive about recommending that radical new approaches be utilized without further study; nonetheless, it appears that some significant changes must be considered. Recognizing that mission operations is the key to *the success of any complicated undertaking, we therefore recommend that an assessment of mission operations, including spacecraft control and scientific instrument and data management and the design and management of software control systems, be studied by the Agency at the earliest possible time and the evaluation be presented to the Committee.*

The Federal Data Processing Reorganization Project has indicated serious failings in virtually all government agencies in the utilization of modern computer technology. While the National Science Foundation and the Advanced Research Project Agency (ARPA) of the Department of Defense continue to support some work in machine intelligence and robotics, this work, especially that supported by ARPA, is becoming more and more mission-oriented. The amount of fundamental research supported by these agencies in machine intelligence and robotics is quite small. Because of its mission, NASA is uniquely suitable as the lead civilian agency in the federal government for the development of frontier technology in computer science, machine intelligence, and robotics. NASA's general engineering competence and ability to carry out complex missions is widely noted and admired. These are just the capabilities needed by any federal agency designated to develop these fields. Although we are hardly experts on federal budgetary deliberations, it seems to us possible that incremental funds might be made available to NASA, over and above the usual NASA budget, if NASA were to make a compelling case for becoming the lead agency in the development of frontier technology in computer science and applications.

[7] *Strategy for Exploration of the Inner Planets: 1977-1987,* Committee on Planetary and Lunar Exploration, Space Science Board, Assembly of Mathematical and Physical Sciences, National Research Council, National Academy of Sciences, Washington, D.C., 1978.

The beneficial impact of such a step for the industrial economy, for other branches of government, for the public well-being, and for NASA's own future effectiveness in an era of tight budgets is likely to be substantial.

We, the NASA Study Group, here state our overall conclusions and recommendations. Our report is complete with supporting documentation leading to these conclusions and recommendations.

A. Conclusions

Conclusion 1. NASA is 5 to 15 years behind the leading edge in computer science and technology.

There are some examples of excellence, but in general we find NASA's use of computer technology disappointing. NASA installations still employ punched-card-based batch processing and obsolete machine languages. There is no NASA nationwide computer network and no widespread time-sharing use of computers. Although Viking was a brilliant technological success, given its design limitations, Viking's use of robotics technology and *in situ* programming was rudimentary. These techniques must be greatly advanced for the complex missions of the future, both planetary and Earth orbital. Most Earth-satellite and much planetary exploration imaging data remains unanalyzed because of the absence of automated systems capable of performing content analyses. Even missions being planned for the 1980s are being designed almost exclusively for traditional data collection with little apparent provision being made for automated extraction of content information.

Conclusion 2. Technology decisions are, to much too great a degree, dictated by specific mission goals, powerfully impeding NASA utilization of modern computer science and technology. Unlike its pioneering work in other areas of science and technology, NASA's use of computer science and machine intelligence has been conservative and unimaginative.

Strict funding limitations and an understandable aversion to mission failure cause mission directors to settle for proven but obsolete and, ironically, often very expensive technologies and systems. As machine intelligence and robotics continue to advance outside of NASA, the consequences of these traditions for higher cost and less efficient data return and analysis become more glaring. The inertial fixation on 15-year-old technologies, including slow processors and very limited memories, strongly inhibit NASA contact with and validation of advanced machine intelligence techniques. Flight minicomputer memories are typically at 16,000 or 21,000 words, enormously restricting options. (For example, a very large number of scientific targets on Jupiter and the Galilean satellites, which otherwise could be acquired, had to be abandoned because of the memory limitations of the Voyager onboard computer.) But million byte memories are now routinely employed and, once space-qualified, could provide enormous flexibility.

Because of the long lead times in the planning cycle, many decisions relating to computers are made five to seven years before launch. Often, the computer technology involved is badly obsolete at the time hardware is frozen. Further, no deliberate effort is made to provide flexibility for software developments in the long time interval before mission operations. (Uplinking mission programs after launch is a small but significant step in the right direction.)

Conclusion 3. The overall importance of machine intelligence and robotics for NASA has not been widely appreciated within the agency, and NASA has made no serious effort to attract bright, young scientists in these fields.

In 1978/1979, the Space Systems and Technology Advisory Committee of the NASA Advisory Council had 40 members. Not one was a computer scientist, although two had peripherally related interests. Few, if any, of the best computer science PhDs from the leading academic institutions in the field work for NASA. There is a looped causality with NASA's general backwardness in computer science (Conclusion 1): An improvement of the quality of computer science at NASA cannot be accomplished without high quality professionals; but such professionals cannot be attracted without up-to-date facilities and the mandate to work at the leading edge of the field.

The problems summarized in Conclusions 1 and 3 cannot be solved separately.

Conclusion 4. The advances and developments in machine intelligence and robotics needed to make future space missions economical and feasible will not happen without a major long-term commitment and centralized, coordinated support.

A table of various planned future space missions and an estimate of technology development efforts needed to automate their system functions was given in Section IV (see Table 4-1). Without these automatic system functions, many of the missions will not be economically and/or technologically feasible.

B. Recommendations

Recommendation 1. NASA should adopt a policy of vigorous and imaginative research in computer science, machine intelligence, and robotics in support of broad NASA objectives.

The problems summarized in the preceding list of conclusions have solutions. They require, most of all, an awareness that the problems exist and a commitment of resources to solve them. Table 6-1 gives the published R&D budgets of the seven largest computer corporations in the United States. In all cases, the total R&D spending is greater than 42% of total profits. The advanced R&D budget would be only a fraction of this amount. Leading corporations in computer science and technology characteristically spend 5.4 percent of gross earnings on relevant research and development. The same percentage of NASA's annual expenditure in computer-related activities would suggest an annual NASA budget for research in computer science, machine intelligence, and robotics approaching one hundred million dollars. An expenditure of half that would equal the combined annual budget for this field for ARPA and the National Science Foundation. If NASA were selected as lead agency (or lead civilian agency) for federal research and development in computer science and technology, such amounts might not be at all impractical. Any significant expenditures should have detectable benefits in three to five years, and very dramatic improvements in NASA programs in 10 years. If NASA were to play such a lead agency role, one of its responsibilities would be to study the long-term implications for individuals and for society of major advances in machine intelligence and robotics.

Recommendation 2. NASA should introduce advanced computer science technology to its Earth orbital and planetary missions, and should emphasize research programs with a multimission focus.

A balance is needed onboard NASA spacecraft between distributed microprocessors and a centralized computer. Although function-directed distribution of processors might be useful, such architectures should not preclude the use of these computing resources for unanticipated needs. Distributed computer concepts emphasizing "fail-safe" performance should receive increased attention. For example, in the case of failure of a computer chip or a unit, a long-term goal is to effect migration of the program and data to other working parts of the systems. Such fail-safe systems require innovative architectures yet to be developed. Dynamically reconfigurable processors with large redundancy are badly needed in NASA.

NASA relies on 256-bit computer memory chips; 16,000 bit and 64,000 bit chips are currently available. A million-bit chip is expected to be available within a few years. The cost of space-qualification of computer hardware may be very high, but the possibility exists that high information-density chips may already work acceptably in the space environment. We recommend that NASA perform space qualification tests on the Shuttle of multiple batches of existing microprocessors and memory chips.

These two examples of developments in computer science and technology will have applications to many NASA missions. We also recommend a transitional period in spacecraft computer system design in which existing miniprocessors and new microprocessors are both utilized, the former as a conservative guarantor of reliability, the latter as an aperture to the future.

Table 6-1. R&D of the Big Seven Computer Companies

Company	1977 Sales in millions of dollars	1977 Profits in millions of dollars	R&D EXPENSE			
			Actual in millions of dollars	As a percent of Sales	As a percent of Profits	Cost of Employees in millions of dollars
IBM	18,133	2,719	1,142	6.3	42	3682
Sperry Rand	3,270	157	168	5.1	107	1965
Honeywell	2,911	134	152	5.2	113	2009
NCR	2,522	144	118	4.7	82	1845
Burroughs	2,901	215	122	4.2	57	2386
Control Data	1,493	62	73	4.9	118	1592
Digital Equipment	1,059	109	80	7.6	73	2218
Composite	32,289	3,540	1,855	5.4	85	2242

In planetary exploration, "... it is clear ... that more advanced mission techniques and instrumentation are required to fulfill the science strategy and achieve the objectives..." of intensive study of a planet.[8] Surface rovers and return-sample missions will be required to meet the science goals for Mars, the Galilean satellites of Jupiter, Titan, and perhaps Venus, as well as for investigation of such specific locations on the lunar surface as putative volatile-rich deposits at permanently shaded regions of the poles. With the exception of the Lunakhod and other Luna-class missions of the Soviet Union, there is little experience with such systems. Because of the long lead times and the complex nature of rover missions, they provide an ideal testing ground for the implementation of the multimission focus of some of our recommendations.

Recommendation 3. Mission objectives should be designed flexibly to take advantage of existing and likely future technological opportunities.

Hardware should be designed to exploit state-of-the-art software and likely near-future software developments. Adoption of this recommendation implies a careful re-examination of missions currently in the planning stages. This recommendation applies not only to spacecraft systems but to ground-based computer systems as well. The man/machine interface, both in Shuttle systems and in mission operations ground equipment, has not, in our opinion, been optimized. In routine mission operations, particularly in mission crisis management, there is a severe short-term competition for human attention and intellectual resources. The problem is a combinatorial one, requiring systematic and exhaustive failure-mode analysis, which can be optimally provided by computer systems, via a probability analysis, analogous to existing computer programs in medical diagnosis. In addition to their value in crisis management, such computer systems will lead to the optimization of subsequent missions.

Recommendation 4. NASA should adopt the following plan of action:

(a) *Establish a focus for computer science and technology at NASA Headquarters for coordinating R&D activities.*

The pace of advance in computer science and technology is so great that even experts in the field have difficulty keeping up with advances and fully utilizing them. The problem is, of course, much more severe for those who are not experts in the field. By establishing a program in computer sciences, NASA can ensure that there is a rapid transfer of new technology to NASA programs. Space exploration offers a unique environment in which to develop and test advanced concepts in this discipline.

This leads to the following specific recommendation: NASA should consider Computer Science and Technology sufficiently vital to its goals to treat the subject as an independent area of study. The specific concerns of this field, enumerated below, should become research and technology issues within NASA on the same basis as propulsion technology, materials science, planetary science, atmospheric physics, etc. This means the creation of a discipline office for computer science with interests in the major subdisciplines of the field and with appropriate contacts within NASA. A suitable budget and program of research and technology grants and contracts would provide the focus in this field the Study Group has found lacking in NASA. On the one hand, it would help make the outstanding workers in the field aware of and interested in serving NASA's needs. Graduate students participating in such a research program would become a source of future employees for NASA centers and contractors. On the other hand, it would provide NASA Headquarters with a better awareness of the potential contributions of computer science to its programs. To be effective, the initial operating budget of such a program should not be below 10 million dollars a year, with a long-term commitment for at least a constant level of funding in real dollars.

Most of the fundamental research under such a program would be carried out at universities and at appropriate NASA centers. Collaboration with industry should be encouraged to expedite technology transfer. To meet the emerging mission requirements, parallel advanced development programs within all of NASA's mission offices are required.

Following is a list of problem areas that should set some goals for both the basic science research program and the advanced development effort:

- Smart sensing; automated content analysis; stereo mapping for eventual Earth and planetary applications.
- Manipulator design, particularly for autonomous use, including structures and effectors, force and touch detectors.

[8] *Ibid*, p. 39.

- Control and feedback systems, particularly those relevant to manipulation and teleoperator development.
- Spacecraft crisis analysis systems.
- Locomotion systems, particularly legged locomotion for difficult terrain.
- Attempts at space qualification of multiple batches of existing microprocessors and memory chips.
- Preliminary studies of automatic and teleoperator assembly of large structures for Earth orbital, lunar, and asteroidal environments.
- Vision systems, particularly for use in locomotion and automated assembly.
- Control and reasoning systems, particularly in support of lunar and planetary rovers.
- Computer architectures for space systems.
- Software tools for space system development.
- Algorithm analysis for critical space-related problems.
- Computer networks and computer-aided teleconferencing. (See paragraph (d) below.)

The current university-based support from NSF and ARPA in computer science and machine intelligence is about 15 million dollars each annually. The level of university funding recommended here would be larger by about 30 percent, allowing NASA to compete effectively for the best talent and ideas. Parallel programs conducted by NASA program offices, which would be based strongly at NASA centers and industry, would approximately double the support requirement. The total support might eventually approach the 100 million dollar level, if NASA were seriously to pursue a broad program of research in computer science.

(b) *Augment the advisory structure of NASA by adding computer scientists to implement the foregoing recommendations.*

NASA is far enough behind the leading edge of the computer science field that major improvements in its operations can be made immediately using existing computer science systems and techniques such as modern data abstraction languages, time-sharing, integrated program development environments, and larger virtual memory computers (especially for onboard processing). Such general improvements in sophistication are almost a prerequisite for a later utilization of machine intelligence and robotics in NASA activities. The advisory organizations should help plan and coordinate NASA's effort in the field and establish contacts with the centers of computer science research.

(c) *Because of the connection of the Defense Mapping Agency's (DMA) Pilot Digital Operations Project with NASA interests, NASA should maintain appropriate liaison.*

DMA has studied the advanced techniques in computer science with an emphasis on machine intelligence. There may be a strong relationship between many DMA concerns and related issues in NASA, particularly in scene analysis and understanding, large database management, and information retrieval. An evaluation by NASA of the DMA planning process associated with the DMA Pilot Digital Operations Project should aid in estimating the costs of NASA's development in this field.

(d) *NASA should form a task group to examine the desirability, feasibility, and general specification of an all-digital, text-handling, intelligent communication system.*

A significant amount of NASA's budget is spent in the transfer of information among a very complex, geographically, and institutionally disparate set of groups that need to exchange messages, ideas, requirements, and documents quickly to keep informed, to plan activities, and to arrive at decisions.

Based on a rough estimate, we predict that such an all-digital network would lead to significant improvements over the present method of carrying out these functions. In addition to the cost savings, there would be improvements in performance. Although it would not eliminate the use of paper and meetings as a means of communication, it would save tons of paper and millions of man-miles of energy-consuming travel. This system would facilitate and improve the participation of scientists in all phases of missions as well as enhance their ability to extract the most value from postmission data analysis.

The implementation of such a system would not be predicated on new developments in artificial intelligence, but on the tools that are in common use at artificial intelligence nodes of the ARPA network and are part of the developing technology of digital information and word processing. If such a development were carried out, it would provide the data base for sophisticated techniques, as they become available, for information retrieval, semantic search, and decision making; and a model for other public and private organizations, scientific, technological, and industrial.

The task group to investigate this development should include elements of NASA management, mission planning and operations, scientific investigators, and information scientists, as well as specialists in artificial intelligence.

Appendix A
The Study Group and Its Workshop Activities

Appendix A
The Study Group and Its Workshop Activities

Biographical Sketches

Dr. Carl Sagan is Director of the Laboratory for Planetary Studies, and David Duncan Professor of Astronomy and Space Sciences at Cornell University. His principal research activities are in the physics and chemistry of planetary atmospheres and surfaces, space-vehicle exploration of the planets, the origin of life on Earth and the search for life elsewhere. Dr. Sagan played a leading role in the Mariner, Viking and Voyager missions to the planets, for which he received the NASA Medals for Exceptional Scientific Achievement and for Distinguished Public Service, and the international astronautics prize, the Prix Galabert. He has served as Chairman of the Division for Planetary Sciences of the American Astronomical Society, as Chairman of the Astronomy Section of the American Association for the Advancement of Science, and as President of the Planetology Section of the American Geophysical Union. For twelve years he was Editor-in-Chief of *ICARUS: International Journal of Solar Systems Studies,* the leading professional magazine devoted to planetary research. He is a Fellow of the American Academy of Arts and Sciences and a member of the Presidential Commission on a National Agenda for the 1980's. In addition to 400 published scientific and popular articles, Dr. Sagan is author, co-author or editor of more than a dozen books including, *Intelligent Life in the Universe* (1966); *The Cosmic Connection* (1973); *The Dragons of Eden* (1977), for which he was awarded the Pulitzer Prize; *Murmurs of Earth* (1978); and, *Broca's Brain* (1979). In 1975, he received the Joseph Priestley Award "for distinguished contributions to the welfare of mankind."

Dr. Raj Reddy is professor of computer science at Carnegie-Mellon University, Pittsburgh, Pennsylvania. Prior to joining the Carnegie-Mellon faculty in 1969, he was an assistant professor of computer science at Stanford University, and also was an applied science representative with the IBM World Trade Corporation. He received a Ph.D. degree in computer science from Stanford in 1966, having previously attended the University of Madras, India, and the University of New South Wales, Australia. His research interests in computer science are in the areas of artificial intelligence and man-machine communication. In particular, Dr. Reddy is working on speech input to computers, visual input to computers, graphics, and task-oriented computer architectures. He is on the editorial boards of *Artificial Intelligence, Image Processing and Computer Graphics, Cognitive Science,* and *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Dr. Ewald Heer is a technical manager in the Office of Technology and Space Program Development at the Jet Propulsion Laboratory (JPL) leading the technology development programs for autonomous systems and space mechanics. After receiving a Dr. Engr. Sc. degree, specializing in system engineering, he conducted and managed several research and advanced development projects at McDonnell Douglas Corporation, General Electric Space Science Laboratory, and JPL. On assignment to NASA Headquarters in 1970, he developed NASA plans for future teleoperator and robot technology. He organized international conferences on Remotely Manned Systems at Caltech in 1972 and at USC in 1975. In addition to publishing technical articles on systems theory, teleoperatory, and robotics, he is robotics editor of the *Journal of Mechanisms and Machine Theory* and has edited two books. Since 1973, he is also associated with the University of Southern California teaching operations research and planning as an adjunct professor of industrial and systems engineering.

Dr. James S. Albus is project manager for sensors and computer control in the automation technology program of the National Engineering Laboratory of the National Bureau of Standards. He has received the Department of Commerce Silver Award for his work in control theory and manipulator design and the Industrial Research IR-100 Award for his work in brain modeling and computer design. Before joining the Bureau of Standards he designed attitude measurement systems for NASA spacecraft and for a short period was program manager of the NASA artificial intelligence program.

Dr. Robert M. Balzer attended Carnegie Institute of Technology under a George Westinghouse scholarship and a National Science Foundation fellowship, where he received his B.S., M.S., and Ph.D. degrees in electrical engineering in 1964, 1965, and 1966 respectively. He joined the RAND Corporation in June 1966 where he was concerned with reducing the effort required to utilize computers for problem solving, especially in on-line environment. In April 1972, he left RAND to help form the USC/Information Sciences Institute. He is currently an associate professor of computer science and project leader of the Specification Acquisition From Experts (SAFE) project. This project is attempting to aid users to compose precise and correct program specifications from informal natural language descriptions by resolving the ambiguity present through context provided by the system's knowledge of programs, programming, and the application domain.

Dr. Thomas O. Binford, a research associate in the Artificial Intelligence Laboratory of the Department of Computer Science at Stanford University, is presently working in the area of computer visions and robotics. Dr. Binford was at the AI lab at MIT previously. The Ph.D. received by Dr. Binford is from the University of Wisconsin.

Dr. R. C. Gonzalez is professor of electrical engineering and computer science at the University of Tennessee, Knoxville, where he is also director of the Image and Patter Analysis laboratory. He is an associate editor of the *International Journal of Computer and Information Sciences* and is a consultant to government and industry, such as the Oak Ridge National Laboratory, NASA, the U.S. Army, and the Martin Marietta Corporation. Dr. Gonzalez is co-author of three books on pattern recognition and image processing. In 1978

he received a UTK Chancellor's Research Scholar Award for his work in these fields.

Dr. Peter E. Hart is the director of the Artificial Intelligence Center at SRI International, which is doing research on experimental automation and is developing expert consultation systems. Other professional experience of Dr. Hart has been as a lecturer, computer science department, Stanford University and as a staff engineer at the Philco Western Development Laboratory. His academic background, BEE (1962) Rensselaer Polytechnic Institute, MS (1963), and PhD (1966) in electrical engineering, Stanford University. Dr. Hart is a coauthor of *Pattern Classification and Scene Analysis*, John Wiley & Sons, Inc. (1973) and has published 16 articles in, for example, *Proc. Int. Joint Conf. on Artificial Intelligence, Commun. ACM, Artificial Intelligence, IEEE Trans. Sys. Sci & Cybernetics.* Professional associations and honors of Dr. Hart are American Association for the Advancement of Science; Association for Computing Machinery; past chairman, Bay Area Chapter of IEEE Group on Systems Science and Cybernetics; Eta Kappa Nu; Sigma Xi; Tau Beta Pi; editorial board, *Current Contents.*

Dr. John Hill is a staff member of the Artificial Intelligence Center at SRI International, Menlo Park, California. His background includes work as a research engineer for Stanford Research Institute in teleoperator research and as charge' de researchne for the Biomechanics Research Laboratory in France in prosthetics control. He received a BSEE (1961) and a MSEE (1963) degree from the University of Illinois, Urbana, and a PhD in electrical engineering (1967) from Stanford University. His research interests are design and control of robot devices for automatic assembly and advances in industrial automation. Dr. Hill is a member of the Robot Institute of America and the Society of Manufacturing Engineers.

Mr. B. Gentry Lee is the manager of mission operations and engineering for the Galileo project at the Jet Propulsion Laboratory, Pasadena, California. Mr. Lee's education is as follows: BA, *summa cum laude*, University of Texas, January 1963, Phi Beta Kappa at age 19, undergraduate studies in languages, literature, mathematics; MS, mathematics, physics and aerospace, MIT, June 1964; Woodrow Wilson fellow at MIT, and Marshall fellow at University of Glasgow, Scotland, 1964-65. Professionally Mr. Lee has been as follows: Aerospace engineer, Martin Marietta Corporation, 1965-1975. Director of science analysis and mission planning for the Viking flight team in Pasadena, California. This executive position involved the operational management of all 200 scientists and mission planners associated with the first landing on the planet Mars. Earlier Viking management positions included mission operations manager and navigation manager. Aerospace manager, Jet Propulsion Laboratory, 1975-1978. His first JPL position was manager of Mission Design Section. Mr. Lee was responsible for top-level design of all U.S. lunar and interplanetary missions. Mr. Lee's current position is manager of Mission Operations and Engineering for Project Galileo, which will be an in-depth investigation of Jupiter and its moons during the middle of the 80s decade.

Dr. Elliott C. Levinthal is adjunct professor and director of the Instrumentation Research Laboratory Department of Genetics, Stanford University School of Medicine. Previously he was associate dean for research affairs, Stanford University School of Medicine. He received a PhD degree from Stanford and holds degrees from Columbia and MIT. He is a principal investigator on the Viking 1975 Lander Imaging Science team and deputy team leader. In 1977 he received the NASA Public Service Medal for "Exceptional Contribution to the Success of the Viking Project." He has served for several years as a consultant to NASA and was co-investigator on the Mariner Mars 1971 photo interpretation team. Dr. Levinthal's research interests include medical electronics, exobiology, application of computers to image processing, and medical information systems. He is a member of the American Association for the Advancement of Science, American Physical Society, IEEE, Sigma Xi, Optical Society of America, and the Biomedical Engineering Society.

Dr. Jack Minker is a professor and chairman of computer science at the University of Maryland, College Park, Maryland. Prior to joining the faculty at the University of Maryland, he served as acting office manager and technical director of the Auerbach Corporation's Washington office, and as manager of information systems technology at RCA. He received a PhD degree in mathematics from the University of Pennsylvania in 1959 and previously received an MS in mathematics from Brooklyn College. His research interests in computer science are in artificial intelligence, automatic theorem proving, and database systems. He is a member of the Association for Computing Machinery, SIAM, and IEEE. He is on the editorial boards of *Information Systems* and *Encyclopedia of Library and Information Science.*

Dr. Marvin Minsky occupies the chair of Donner professor of science at MIT. He was founder and director of the MIT Artificial Intelligence Laboratory until appointing Dr. Winston to the position. Dr. Minsky has played a central role in the scientific design of many of today's research programs in robotics and artificial intelligence. He is author of books and papers on the subjects of artificial intelligence, theory of computational complexity, cognitive psychology, and physical optics. For outstanding contributions to computer science, he received the ACM's Turing Award. He is a member of the National Academy of Sciences.

Dr. Donald A. Norman is a professor of psychology at the University of California at San Diego where he is also director of the program in cognitive science in the Center for Human Information Processing. From 1974 to 1978, he was chair of the Department of Psychology. His research interests concentrate on understanding the psychological mechanisms of human cognition, with emphasis on problems in attention and memory. Dr. Norman received a BS degree from MIT and a MS degree from the University of Pennsylvania, both in electrical engineering. His doctorate, from the University of Pennsylvania, is in psychology. He has published in journals and books, and is the author or editor of four books. He is on the editorial boards of *Cognitive Psychology, Journal of Cognitive Science*, and the *Journal of Experimental Psychology*. He is a fellow of the American Psychological Association and of the American Association for the Advancement of Science.

Dr. Charles J. Rieger received the BS degree in mathematics and computer science from Purdue University, Lafayette, Indiana, in 1970 and the PhD from Stanford University, Palo Alto, California in 1974. He is currently an associate professor in computer science at the University of Maryland, College Park, Maryland. His research interests are in the area of artificial intelligence and cognitive modeling with particular emphasis on models of human inference in language understanding.

Dr. Thomas B. Sheridan is professor of engineering and applied psychology at MIT where his research is on man-machine control of aircraft, nuclear plants, and undersea teleoperators. He has served as visiting faculty member at the University of California at Berkeley, Stanford University, and the Delft University of Technology, the Netherlands. He is co-author of *Man-Machine Systems* (MIT Press, 1974) and *Monitoring Behavior and Supervisory Control* (Plenum, 1976). Formerly president of the IEEE Systems, Man and Cybernetics Society and editor of *IEEE Transactions on Man-Machine Systems*, he presently serves on the NASA Life Sciences Advisory Committee and the Congressional Office of Technology Assessment Task Force on Appropriate Technology. He is a fellow of the Human Factors Society and a recipient of the Society's Paul M. Fitts award.

Dr. William M. Whitney is the section manager for information systems research at the Jet Propulsion Laboratory in Pasadena, California. He also has a part-time assignment in the Office of Technology and Space Program Development to plan activities that will strengthen JPL in the technologies underlying its information-system applications. He received a BS degree from Caltech in 1951 in physics, and a PhD from MIT in 1956 in experimental low-temperature physics. He served as instructor and as assistant professor in the MIT physics department from 1956 until August 1963, when he joined JPL to conduct research in the guidance and control section. In 1967, he was appointed manager of that section, which became the forerunner of the present information systems research section. He led the planning that culminated in the creation of JPL's robotics research program in 1972, and served as its technical leader until 1978. He was editor and a principal author of the section on information management in "A Forecast of Space Technology, 1980 – 2000," prepared by JPL as a part of a broad "Outlook for Space" study conducted by NASA in 1974-75. Dr. Whitney has taken part in studies to set new directions for JPL and NASA research and development efforts.

Dr. Patrick H. Winston is an associate professor of computer science at the Massachusetts Institute of Technology, where he is also director of the Artificial Intelligence Laboratory. He is the editor of *The Psychology of Computer Vision* and the co-editor of *Artificial Intelligence: An MIT Perspective*, as well as author of the textbook *Introduction to Artificial Intelligence*, Addison-Wesley; New York. His research focuses on the subject of making computers learn.

Dr. Stephen Yerazunis is associate dean of engineering and professor of chemical engineering at Rensselaer Polytechnic Institute, Troy, New York. He received a doctorate in chemical engineering from Rensselaer Polytechnic Institute. His earlier research interests included vapor-liquid equilibria and heat and mass transfer phenomena at high mass transfer rates in turbulent flows. His current research involves appraisal of electrical energy alternatives for the State of New York and guidance of an autonomous rover for unmanned planetary exploration. His interest in the latter is directed towards the development of short-range hazard detection and avoidance systems, as well as the configuration of the rover. He has been a consultant to the Knolls Atomic Power Laboratory and the General Electric Company. He is a fellow of the American Institute of Chemical Engineers.

Dr. William B. Gevarter is in charge of NASA's Space Guidance and Control, and artificial intelligence and robotics research programs. Previously he was with NASA Headquarters Office of Policy Analysis where he carried out research and analysis on the interaction of technology and society. He received a PhD degree in engineering from Stanford University, specializing in modern control theory. He has served as vice chairman of the American Society for Cybernetics and as chapter chairman of the Systems, Man and Cybernetics Society, Washington, D.C. He is a member of the American Institute of Aeronautics and Astronautics, the World Future Society, the Society for General Systems Research, and the Association for Humanistic Psychology.

Stanley R. Sadin is program manager of Space Systems Studies and Planning at NASA Headquarters.

NASA Study Group on Machine Intelligence and Robotics – Workshop I

University of Maryland

June 27–29, 1977

Speaker	Subject
Stanley R. Sadin	Program Overview
Daniel H. Herman	Planetary Exploration
King S. Fu	Pattern Recognition
Samuel W. McCandless	Global Resources and Earth Observation
Tom O. Binford	Scene Analysis
David Blanchard	Flight Operations and Mission Control
Richard C. Henry	Search for Extraterrestrial Intelligence
Harold B. Alsberg	End-to-End Data Management
J. H. von Puttkamer	Space Industrialization
Douglas A. Gilstad	Large Area Space Structures
William L. Smith	Remotely Operated Systems Development
Dan C. Popma	Teleoperator Supporting Research and Technology
Leonard Friedman	Robotic Tool Systems
Lester K. Fero	Space Transportation Systems and Associated Ground Operations
Simon V. Manson	Space Power
William B. Gevarter	NASA's Robotics Program
Stephen Yerazunis	Locomotion and Navigation of Roving Vehicles
Charles J. Rieger	Communicating with Machines
Berthold Horn	Machine Intelligence and Robotics – Prospects for Practical Applications

NASA Study Group on Machine Intelligence and Robotics – Workshop IIA

Jet Propulsion Laboratory

September 28–29, 1977

Speaker	Subject
Henry W. Norris	Results of Mars 1984 Mission Study
Arden L. Albee	Science Goals Achieved by a Rover
Victor C. Clarke	Mission Design – Character of the 1984 Opportunity
James R. French	Rover System Description
Henry W. Norris	Project Overview – Alternate System Description, Costs, Schedule
Marvin Minsky	Manipulation for Planetary Surface Rovers
Boris M. Dobrotin	Manipulation and Sensing Requirements as a Function of Science Objectives
Marvin Minsky	Summation and Discussions
William Whitney	Locomotion for Planetary Surface Rovers
James D. Burke	Locomotion for Planetary Surface Rovers
Garrett Paine	Locomotion for Planetary Surface Rovers
Berthold Horn	Locomotion for Planetary Surface Rovers
Robert B. McGhee	Locomotion for Planetary Surface Rovers
William Whitney	Summation and Discussions
Charles J. Rieger	Operations for Planetary Surface Rovers
George P. Textor	Operations for Planetary Surface Rovers
James S. Albus	Operations for Planetary Surface Rovers
Charles J. Rieger	Summation and Discussions

NASA Study Group on Machine Intelligence and Robotics – Workshop IIB

Jet Propulsion Laboratory

September 30, 1977

Subject	Speaker
Algirdas A. Avizienis	Architectures for S/C Computers – Introduction and Overview
David A. Rennels	Architectures for S/C Computers – S/C Dist. Computer Architectures
Herbert Hecht	Architectures for S/C Computers – Centralized Satellite Computer
Danny Cohen	Architectures for S/C Computers – Discussions
William M. Whitney	Architectures for S/C Computers – Discussions
Samuel Fuller	Trends in Computer Architectures – Multiprocessor Architectures
Richard Greenblatt	Trends in Computer Architectures – LISP Processor
Marvin Minsky	Trends in Computer Architectures – Discussions
Justin Ratner	Trends in LSI Technology – New Directions in MOS Technology
Carver A. Mead	Trends in LSI Technology – Designing in LSI
Michael Ebersole	Panel Discussion
Alan Perlis	Panel Discussion
Ivan Sutherland	Panel Discussion
Carver A. Mead	Panel Discussion
Algirdas A. Avizienis	Panel Discussion

NASA Study Group on Machine Intelligence and Robotics – Workshop III

Goddard Space Flight Center

November 30, 1977

Speaker	Subject
David Blanchard	NASA Organizations and Project Development Programs
John B. Zegalia	A Typical NASA End-to-End Data System
Richard des Jardins	Mission Independent Ground Operation Systems
Stephen R. McReynolds	Survey of NASA Applications of Advanced Automation
John Y. Sos	Trends in Space Telemetry Data Processing
John J. Quann	Large Data Base Application Requirements
Robert D. Chapman	Need of Space Lab Facility Class Instruments of the Future
Press Rose	Payload Software Technology
Robert Balzer	Report on MSFC Data Management Symposium
Leonard Friedman	Report on AIAA Computers in Aerospace Conference
B. Gentry Lee	Mission Operations for Planetary Missions
James Porter	Viking Mission Operation Strategy
B. A. Claussen	Viking Lander Software
Harlan Mills	System Development Methodology
Azriel Rosenfeld	Spacial Data Bases: Problems and Prospects
Nico Habermann	System Development Control
Robert Balzer	Program Specification and Verification
John V. Guttag	Aspects of Program Specifications
Brian Smith	KRL: Knowledge Representation Language
Mary Shaw	ALPHARD: A Language for the Development of Structured Programs
Warren Teitelman	Interactive Development of Large Systems
Allen Newell	ZOG: An Iterative System for Exploring Large Knowledge Bases
Donald A. Norman	Powers and Limitations of the Human Brain, Mind, Storage
Thomas B. Sheridan	Discussion
Donald A. Norman	Discussion

NASA Study Group on Machine Intelligence and Robotics – Workshop IV

Johnson Space Center
February 1–2, 1978

Speaker	Subject
Brian O'Leary	The Mining, Delivery and Processing of Non-Terrestrial Materials in Space
Earle M. Crum	Lunar Resources Utilization for Space Construction
George F. von Tiesenhausen	Space Processing and Manufacturing
W. H. Steurer	Recovery of Lunar Metals for Terrestrial Consumption
C. C. Kraft	Discussions
Allen J. Louviere	Attached Manipulators and Fabrication in Space
Robert V. Powell	Large Antenna Reflectors Deployment and Erection
Ted Carey	Geostationary Platform Studies and Teleoperators
Hugh J. Dudley	Fabrication in Space and Simulation of Fabrication Operations
George W. Smith	Teleoperator Control for Space Construction
William R. Ferrell	Human Performance and Man-Machine Allocation for Space Tasks
Oliver Selfridge	Multilevel Exploration: Human and Computer Roles
Donald A. Norman	Human Information Processing
Thomas B. Sheridan	Summary Remarks

NASA Study Group on Machine Intelligence and Robotics – Workshop V

NASA Headquarters
March 8–9, 1978

Speaker	Subject
Donald Williams/ Robert Cunningham	Automated Scene Analysis for Space Systems
Jay M. Tenenbaum	Application of AI to Remote Sensing
Phillip H. Swain	At the Frontiers of Earth Resources Image Processing
Henry Cook	DMA Applications of Automatic Cartography & Possible Requirements for Machine Intelligence
Alex F. H. Goetz	Geological Resource Exploration
Thomas Young	Future Mission Requirements
Charles Elachi	Radar Imaging, Venus Orbiting Radar, SEASAT
Edward J. Groth	Large Space Telescopes
James Cutts	Planetary Geology
Raj Reddy	SMU Signal Processor
David Schaeffer	The Massive Parallel Processor (MPP)
Graham Nudd	CCD Image Processing
Q. R. Mitchell	Coding and Data Compression Techniques
B. R. Hunt	Enhancement, Restoration, and Geometric Transformations
V. Casler/Ivan Sutherland	Graphics and Simulation Systems
Berthold Horn	Motion and Texture
David Milgram/ Azriel Rosenfeld	Relaxation Algorithms
Jay M. Tenenbaum	Segmentation

Appendix B

Marvin Minsky. "Steps Toward Artificial Intelligence," pp 406-450, in *Computers and Thoughts*, edited by Edward A. Feigenbaum and Julian Feldman, copyrighted 1963 by McGraw-Hill, Inc. Reproduced by permission of the Institute of Electrical and Electronics Engineers, Inc., successor to the Institute of Radio Engineers, original copyright holder of the *Proceedings of the Institute of Radio Engineers*, January 1961, Vol 49, pp 8-30.

Appendix B

STEPS TOWARD ARTIFICIAL INTELLIGENCE

by Marvin Minsky

Introduction

A visitor to our planet might be puzzled about the role of computers in our technology. On the one hand, he would read and hear all about wonderful "mechanical brains" baffling their creators with prodigious intellectual performance. And he (or it) would be warned that these machines must be restrained, lest they overwhelm us by might, persuasion, or even by the revelation of truths too terrible to be borne. On the other hand, our visitor would find the machines being denounced, on all sides, for their slavish obedience, unimaginative literal interpretations, and incapacity for innovation or initiative; in short, for their inhuman dullness.

Our visitor might remain puzzled if he set out to find, and judge for himself, these monsters. For he would find only a few machines (mostly "general-purpose" computers, programmed for the moment to behave according to some specification) doing things that might claim any real intellectual status. Some would be proving mathematical theorems of rather undistinguished character. A few machines might be playing certain games, occasionally defeating their designers. Some might be distinguishing between hand-printed letters. Is this enough to justify so much interest, let alone deep concern? I believe that it is; that we are on the threshold of an era that will be strongly influenced, and quite possibly dominated, by intelligent problem-solving machines. But our purpose is not to guess about what the future may bring; it is only to try to describe and explain what seem now to be our first steps toward the construction of "artificial intelligence."

Along with the development of general-purpose computers, the past few years have seen an increase in effort toward the discovery and mechanization of problem-solving processes. Quite a number of papers have appeared describing theories or actual computer programs concerned with game playing, theorem proving, pattern recognition, and other domains which would seem to require some intelligence. The literature does not include any general discussion of the outstanding problems of this field.

In this article, an attempt will be made to separate out, analyze, and find the relations between some of these problems. Analysis will be supported with enough examples from the literature to serve the introductory function of a review article, but there remains much relevant work not described here. This report is highly compressed, and therefore, cannot begin to discuss all these matters in the available space.

There is, of course, no generally accepted theory of "intelligence"; the analysis is our own and may be controversial. We regret that we cannot give full personal acknowledgments here—suffice it to say that we have discussed these matters with almost every one of the cited authors.

It is convenient to divide the problems into five main areas: Search, Pattern Recognition, Learning, Planning, and Induction; these comprise the main divisions of the report. Let us summarize, the entire argument very briefly:

A computer can do, in a sense, only what it is told to do. But even when we do not know exactly how to solve a certain problem, we may program a machine to *Search* through some large space of solution attempts. Unfortunately, when we write a straightforward program for such a search, we usually find the resulting process to be enormously inefficient. With *Pattern Recognition* techniques, efficiency can be greatly improved by restricting the machine to use its methods only on the kind of attempts for which they are appropriate. And with *Learning*, efficiency is further improved by directing Search in accord with earlier experiences. By actually analyzing the situation, using what we call *Planning* methods, the machine may obtain a really fundamental improvement by replacing the originally given Search by a much smaller, more appropriate exploration. Finally, in the section on *Induction*, we consider some rather more global concepts of how one might obtain intelligent machine behavior.

I. *The Problem of Search*[1]

If, for a given problem, we have a means for checking a proposed solution, then we can solve the problem by testing all possible answers. But this

[1] The adjective "heuristic," as used here and widely in the literature, means *related to improving problem-solving performance;* as a noun it is also used in regard to any method or trick used to improve the efficiency of a problem-solving system. A

always takes much too long to be of practical interest. Any device that can reduce this search may be of value. If we can detect relative improvement, then "hill-climbing" (Sec. I-B) may be feasible, but its use requires some structural knowledge of the search space. And unless this structure meets certain conditions, hill-climbing may do more harm than good.

When we talk of problem-solving in what follows we will usually suppose that all the problems to be solved are initially *well defined* (McCarthy, 1956). By this we mean that with each problem we are given some systematic way to decide when a proposed solution is acceptable. Most of the experimental work discussed here is concerned with such well-defined problems as are met in theorem-proving, or in games with precise rules for play and scoring.

In one sense all such problems are trivial. For if there exists a solution to such a problem, that solution can be found eventually by any blind exhaustive process which searches through all possibilities. And it is usually not difficult to mechanize or program such a search.

But for any problem worthy of the name, the search through all possibilities will be too inefficient for practical use. And on the other hand, systems like chess, or nontrivial parts of mathematics, are too complicated for complete analysis. Without complete analysis, there must always remain some core of search, or "trial and error." So we need to find techniques through which the results of *incomplete analysis* can be used to make the search more efficient. The necessity for this is simply overwhelming: a search of all the paths through the game of checkers involves some 10^{40} move choices (Samuel, 1959*a*), in chess, some 10^{120} (Shannon, in Newman, 1956). If we organized all the particles in our galaxy into some kind of parallel computer operating at the frequency of hard cosmic rays, the latter computation would still take impossibly long; we cannot expect improvements in "hardware" alone to solve all our problems! Certainly we must use whatever we know in advance to guide the trial generator. And we must also be able to make use of results obtained along the way.[2,3]

"heuristic program," to be considered successful, must work well on a variety of problems, and may often be excused if it fails on some. We often find it worthwhile to introduce a heuristic method which happens to cause occasional failures, if there is an over-all improvement in performance. But imperfect methods are not necessarily heuristic, nor vice versa. Hence "heuristic" should not be regarded as opposite to "foolproof"; this has caused some confusion in the literature.

[3] McCarthy (1956) has discussed the enumeration problem from a recursive-function-theory point of view. This incomplete but suggestive paper proposes, among other things, that "the enumeration of partial recursive functions should give an early place to compositions of functions that have already appeared."

I regard this as an important notion, especially in the light of Shannon's results (1949) on two-terminal switching circuits—that the "average" n-variable switching

A. Relative Improvement, Hill-climbing, and Heuristic Connections

A problem can hardly come to interest us if we have no background of information about it. We usually have some basis, however flimsy, for detecting *improvement;* some trials will be judged more successful than others. Suppose, for example, that we have a *comparator* which selects as the better, one from any pair of trial outcomes. Now the comparator cannot, alone, serve to make a problem well defined. No goal is defined. But if the comparator-defined relation between trials is "transitive" (*i.e,* if *A dominates B* and *B dominates C* implies that *A dominates C*), then we can at least define "progress," and ask our machine, given a time limit, to do the best it can.

But it is essential to observe that a comparator by itself, however shrewd, cannot alone give any improvement over exhaustive search. The comparator gives us information about partial success, to be sure. But we need also some way of using this information to direct the pattern of search in promising directions; to select new trial points which are in some sense "like," or "similar to," or "in the same direction as" those which have given the best previous results. To do this we need some additional structure on the search space. This structure need not bear much resemblance to the ordinary spatial notion of direction, or that of distance, but it must somehow tie together points which are heuristically related.

We will call such a structure a *heuristic connection.* We introduce this term for informal use only—that is why our definition is itself so informal. But we need it. Many publications have been marred by the misuse, for this purpose, of precise mathematical terms, *e.g., metric* and *topological.* The term "connection," with its variety of dictionary meanings, seems just the word to designate a relation without commitment as to the exact nature of the relation.

An important and simple kind of heuristic connection is that defined when a space has coordinates (or parameters) and there is also defined a numerical "success function" E which is a reasonably smooth function of the coordinates. Here we can use local optimization or *hill-climbing* methods.

---

function requires about $2^n/n$ contacts. This disaster does not usually strike when we construct "interesting" large machines, presumably because they are based on composition of functions already found useful. One should not overlook the pioneering paper of Newell (1955), and Samuel's discussion of the minimaxing process in (1959*a*).

[8] In 1952 and especially in 1956, Ashby has an excellent discussion of the search problem. (However, I am not convinced of the usefulness of his notion of "ultra-stability," which seems to be little more than the property of a machine to search until something stops it.)

B. Hill-climbing

Suppose that we are given a black-box machine with inputs $\lambda_1, \ldots, \lambda_n$ and an output $E(\lambda_1, \ldots, \lambda_n)$. We wish to maximize E by adjusting the input values. But we are not given any mathematical description of the function E; hence we cannot use differentiation or related methods. The obvious approach is to explore locally about a point, finding the direction of steepest ascent. One moves a certain distance in that direction and repeats the process until improvement ceases. If the hill is smooth this may be done, approximately, by estimating the gradient component $\partial E/\partial \lambda_i$ separately for each coordinate λ_i. There are more sophisticated approaches (one may use noise added to each variable, and correlate the output with each input, see Fig. 1), but this is the general idea. It is a fundamental technique, and we see it always in the background of far more complex systems. Heuristically, its great virtue is this: the sampling effort (for determining the direction of the gradient) grows, in a sense, only linearly with the number of parameters. So if we can solve, by such a method, a certain kind of problem involving many parameters, then the addition of more parameters of the same kind ought not cause an inordinate increase in difficulty. We are particularly interested in problem-solving methods which can be so extended to more difficult problems. Alas, most interesting systems which involve combinational operations usually grow exponentially more difficult as we add variables.

A great variety of hill-climbing systems have been studied under the names of "adaptive" or "self-optimizing" servomechanisms.

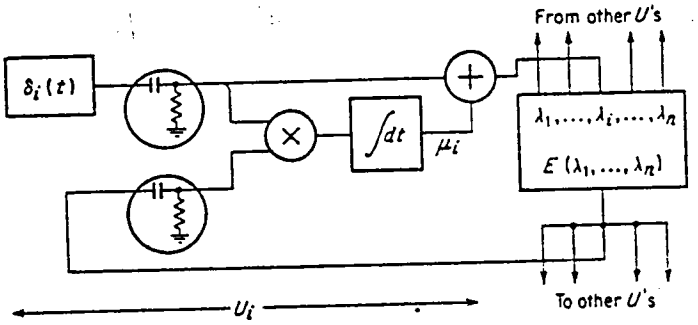


Figure 1. "Multiple simultaneous optimizers" search for a (local) maximum value of some function $E(\lambda_1, \ldots, \lambda_n)$ of several parameters. Each unit U_i independently "jitters" its parameter λ_i, perhaps randomly, by adding a variation $\delta_i(t)$ to a current mean value μ_i. The changes in the quantities δ_i and E are correlated, and the result is used to (slowly) change μ_i. The filters are to move d-c components. This simultaneous technique, really a form of coherent detection, usually has an advantage over methods dealing separately and sequentially with each parameter. [Cf. the discussion of "informative feedback" in Wiener (1948, pp. 133ff.).]

C. Troubles with Hill-climbing

Obviously, the gradient-following hill-climber would be trapped if it should reach a *local peak* which is not a true or satisfactory optimum. It must then be forced to try larger steps or changes.

It is often supposed that this false-peak problem is the chief obstacle to machine learning by this method. This certainly can be troublesome. But for really difficult problems, it seems to us that usually the more fundamental problem lies in finding any significant peak at all. Unfortunately the known E functions for difficult problems often exhibit what we have called (Minsky and Selfridge, 1960) the *"Mesa Phenomenon"* in which a small change in a parameter usually leads to either no change in performance or to a large change in performance. The space is thus composed primarily of flat regions or "mesas." Any tendency of the trial generator to make small steps then results in much aimless wandering without compensating information gains. A profitable search in such a space requires steps so large that hill-climbing is essentially ruled out. The problem-solver must find other methods; hill-climbing might still be feasible with a different heuristic connection.

Certainly, in our own intellectual behavior we rarely solve a tricky problem by a steady climb toward success. I doubt that in any one simple mechanism, *e.g.*, hill-climbing, will we find the means to build an efficient and general problem-solving machine. Probably, an intelligent machine will require a variety of different mechanisms. These will be arranged in hierarchies, and in even more complex, perhaps recursive, structures. And perhaps what amounts to straightforward hill-climbing on one level may sometimes appear (on a lower level) as the sudden jumps of "insight."

II. The Problem of Pattern Recognition

In order not to try all possibilities, a resourceful machine must classify problem situations into categories associated with the domains of effectiveness of the machine's different methods. These pattern-recognition methods must extract the heuristically significant features of the objects in question. The simplest methods simply match the objects against standards or prototypes. More powerful "property-list" methods subject each object to a sequence of tests, each detecting some *property* of heuristic importance. These properties have to be invariant under commonly encountered forms of distortion. Two important problems arise here—inventing new useful properties, and combining many properties to form a recognition system. For complex problems, such methods will have to be augmented by facilities for subdividing complex objects and describing the complex relations between their parts.

Any powerful heuristic program is bound to contain a variety of different methods and techniques. At each step of the problem-solving process the machine will have to decide what aspect of the problem to work on, and then which method to use. A choice must be made, for we usually cannot afford to try all the possibilities. In order to deal with a goal or a problem, that is, to choose an appropriate method, we have to recognize what kind of thing it is. Thus the need to choose among actions compels us to provide the machine with classification techniques, or means of evolving them. It is of overwhelming importance that the machine have classification techniques which are realistic. But "realistic" can be defined only with respect to the environments to be encountered by the machine, and with respect to the methods available to it. Distinctions which cannot be exploited are not worth recognizing. And methods are usually worthless without classification schemes which can help decide when they are applicable.

A. Teleological Requirements of Classification

The useful classifications are those which match the goals and methods of the machine. The objects grouped together in the classifications should have something of heuristic value in common; they should be "similar" in a useful sense; they should depend on relevant or essential features. We should not be surprised, then, to find ourselves using inverse or teleological expressions to define the classes. We really do want to have a grip on "the class of objects which can be transformed into a result of form Y," that is, the class of objects which will satisfy some goal. One should be wary of the familiar injunction against using teleological language in science. While it is true that talking of goals in some contexts may dispose us towards certain kinds of animistic explanations, this need not be a bad thing in the field of problem-solving; it is hard to see how one can solve problems without thoughts of purposes. The real difficulty with teleological definitions is technical, not philosophical, and arises when they have to be used and not just mentioned. One obviously cannot afford to use for classification a method which actually requires waiting for some remote outcome, if one needs the classification precisely for deciding whether to try out that method. So, in practice, the ideal teleological definitions often have to be replaced by practical approximations, usually with some risk of error; that is, the definitions have to be made *heuristically effective*, or economically usable. This is of great importance. (We can think of "heuristic effectiveness" as contrasted to the ordinary mathematical notion of "effectiveness" which distinguishes those definitions which can be realized at all by machine, regardless of efficiency.)

B. Patterns and Descriptions

It is usually necessary to have ways of assigning *names*—symbolic expressions—to the defined classes. The structure of the names will have a

crucial influence on the mental world of the machine, for it determines what kinds of things can be conveniently thought about. There are a variety of ways to assign names. The simplest schemes use what we will call *conventional* (or *proper*) names; here, arbitrary symbols are assigned to classes. But we will also want to use complex *descriptions* or *computed names;* these are constructed for classes by processes which *depend on the class definitions.* To be useful, these should reflect some of the structure of the things they designate, abstracted in a manner relevant to the problem area. The notion of description merges smoothly into the more complex notion of *model;* as we think of it, a model is a sort of active description. It is a thing whose form reflects some of the structure of the thing represented, but which also has some of the character of a working machine.

In Sec. III we will consider "learning" systems. The behavior of those systems can be made to change in reasonable ways depending on what happened to them in the past. But by themselves, the simple learning systems are useful only in recurrent situations; they cannot cope with any significant novelty. Nontrivial performance is obtained only when learning systems are supplemented with classification or pattern-recognition methods of some inductive ability. For the variety of objects encountered in a nontrivial search is so enormous that we cannot depend on recurrence, and the mere accumulation of records of past experience can have only limited value. Pattern Recognition, by providing a heuristic connection which links the old to the new, can make learning broadly useful.

What is a "pattern"? We often use the term teleologically to mean a set of objects which can in some (useful) way be treated alike. For each problem area we must ask, "What patterns would be useful for a machine working on such problems?"

The problems of *visual* pattern recognition have received much attention in recent years and most of our examples are from this area.

C. Prototype-derived Patterns

The problem of reading *printed* characters is a clearcut instance of a situation in which the classification is based ultimately on a fixed set of "prototypes"—*e.g.*, the dies from which the type font was made. The individual marks on the printed page may show the results of many distortions. Some distortions are rather systematic: change in size, position, orientation. Some are of the nature of noise: blurring, grain, low contrast, etc.

If the noise is not too severe, we may be able to manage the identification by what we call a *normalization and template-matching* process. We first remove the differences related to size and position—that is, we *normalize* the input figure. One may do this, for example, by constructing a similar figure inscribed in a certain fixed triangle (see Fig. 2); or one may transform the figure to obtain a certain fixed center of gravity and a unit second central moment. [There is an additional problem with rotational

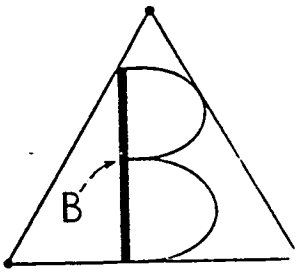


Figure 2. A simple normalization technique. If an object is expanded uniformly, without rotation, until it touches all three sides of a triangle, the resulting figure will be unique, and pattern recognition can proceed without concern about relative size and position.

equivalence where it is not easy to avoid all ambiguities. One does not want to equate "6" and "9." For that matter, one does not want to equate (0,*o*), or (*X*,x) or the *o*'s in x_o and x^o, so that there may be context dependency involved.] Once normalized, the unknown figure can be compared with *templates* for the prototypes and, by means of some measure of *matching*, choose the best fitting template. Each "matching criterion" will be sensitive to particular forms of noise and distortion, and so will each normalization procedure. The inscribing or boxing method may be sensitive to small specks, while the moment method will be especially sensitive to smearing, at least for thin-line figures, etc. The choice of a matching criterion must depend on the kinds of noise and transformations commonly encountered. Still, for many problems we may get acceptable results by using straightforward correlation methods.

When the class of equivalence transformations is very large, *e.g.*, when local stretching and distortion are present, there will be difficulty in finding a uniform normalization method. Instead, one may have to consider a process of adjusting locally for best fit to the template. (While measuring the matching, one could "jitter" the figure locally; if an improvement were found the process could be repeated using a slightly different change, etc.) There is usually no practical possibility of applying to the figure *all* of the admissible transformations. And to recognize the *topological* equivalence of pairs such as those in Fig. 3 is likely beyond any practical kind of iterative local-improvement or hill-climbing matching procedure. (Such recognitions can be mechanized, though, by methods which follow lines, detect vertices, and build up a *description* in the form, say, of a vertex-connection table.)

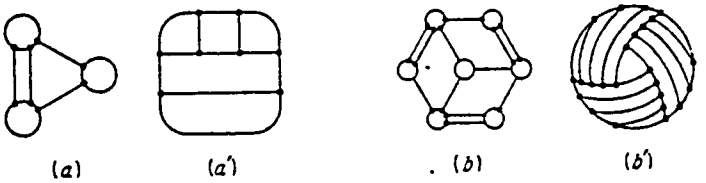


Figure 3. The figures *A*, *A'* and *B*, *B'* are topologically equivalent pairs. Lengths have been distorted in an arbitrary manner, but the connectivity relations between corresponding points have been preserved. In Sherman (1959) and Haller (1959) we find computer programs which can deal with such equivalences.

The template-matching scheme, with its normalization and direct comparison and matching criterion, is just too limited in conception to be of much use in more difficult problems. If the transformation set is large, normalization, or "fitting," may be impractical, especially if there is no adequate heuristic connection on the space of transformations. Furthermore, for each defined pattern, the system has to be presented with a prototype. But if one has in mind a fairly abstract class, one may simply be unable to represent its essential features with one or a very few concrete examples. How could one represent with a single prototype the class of figures which have an even number of disconnected parts? Clearly, the template system has negligible descriptive power. The property-list system frees us from some of these limitations.

D. Property Lists and "Characters"

We define a *property* to be a two-valued function which divides figures into two classes; a figure is said to have or not have the property according to whether the function's value is 1 or 0. Given a number N of distinction properties, we could define as many as 2^N subclasses by their set intersections and, hence, as many as 2^{2^N} *patterns* by combining the properties with AND's and OR's. Thus, if we have three properties, *rectilinear, connected,* and *cyclic,* there are eight subclasses (and 256 patterns) defined by their intersections (see Fig. 4).

If the given properties are placed in a fixed order then we can represent any of these elementary regions by a vector, or string of digits. The vector so assigned to each figure will be called the *Character* of that figure (with respect to the sequence of properties in question). [In "Some Aspects of Heuristic Programming and Artificial Intelligence" (1959*a*), we use the

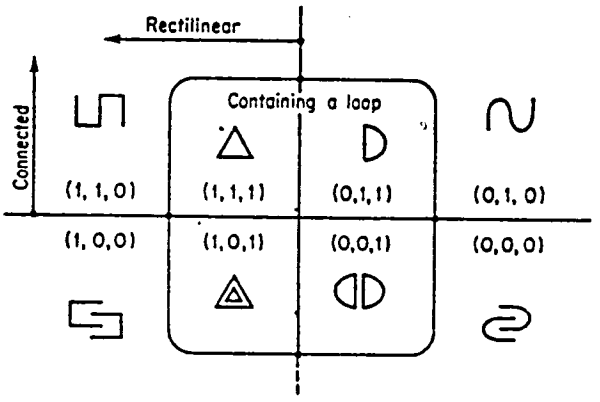


Figure 4. The eight regions represent all the possible configurations of values of the three properties "rectilinear," "connected," "containing a loop." Each region contains a representative figure, and its associated binary "Character" sequence.

term *characteristic* for a property without restriction to 2 values.] Thus a square has the Character (1,1,1) and a circle the Character (0,1,1) for the given sequence of properties.

For many problems one can use such Characters as names for categories and as primitive elements with which to define an adequate set of patterns. Characters are more than conventional names. They are instead very rudimentary forms of description (having the form of the simplest symbolic expression—the *list*) whose structure provides some information about the designated classes. This is a step, albeit a small one, beyond the template method; the Characters are not simple instances of the patterns, and the properties may themselves be very abstract. Finding a good set of properties is the major concern of many heuristic programs.

E. Invariant Properties

One of the prime requirements of a good property is that it be invariant under the commonly encountered equivalence transformations. Thus for visual Pattern Recognition we would usually want the object identification to be independent of uniform changes in size and position. In their pioneering paper Pitts and McCulloch (1947) describe a general technique for forming invariant properties from noninvariant ones, assuming that the transformation space has a certain (group) structure. The idea behind their mathematical argument is this: suppose that we have a function P of figures, and suppose that for a given figure F we define $[F] = \{F_1, F_2, \ldots\}$ to be the set of all figures equivalent to F under the given set of transformations; further, define $P[F]$ to be the set $\{P(F_1), P(F_2), \ldots\}$ of values of P on those figures. Finally, define $P^*[F]$ to be AVERAGE $(P[F])$. Then we have a new property P^* whose values are independent of the selection of F from an equivalence class defined by the transformations. We have to be sure that when different representatives are chosen from a class the collection $[F]$ will always be the same in each case. In the case of continuous transformation spaces, there will have to be a *measure* or the equivalent associated with the set $[F]$ with respect to which the operation AVERAGE is defined, say, as an integration.[4]

This method is proposed (Pitts and McCulloch, 1947) as a neurophysiological model for pitch-invariant hearing and size-invariant visual

[4] In the case studied in Pitts and McCulloch (1947) the transformation space is a *group* with a uniquely defined measure: the set $[F]$ can be computed without repetitions by scanning through the application of all the transforms T_α to the given figure so that the invariant property can be defined by

$$P^*(F) = \int_{\alpha \in G} P(T_\alpha(F))\, d\mu$$

where G is the group and μ the measure. By substituting $T_\beta(F)$ for F in this, one can see that the result is independent of choice of β since we obtain the same integral over $G\beta^{-1} = G$.

recognition (supplemented with visual centering mechanisms). This model is discussed also by Wiener.[5] Practical application is probably limited to one-dimensional groups and analog scanning devices.

In much recent work this problem is avoided by using properties already invariant under these transformations. Thus a property might count the number of connected components in a picture—this is invariant under size and position. Or a property may count the number of vertical lines in a picture—this is invariant under size and position (but not rotation).

F. Generating Properties

The problem of generating useful properties has been discussed by Selfridge (1955); we shall summarize his approach. The machine is given, at the start, a few basic transformations $A_1, \ldots, A_n$, each of which transforms, in some significant way, each figure into another figure. A_1 might, for example, remove all points *not on a boundary* of a solid region; A_2 might leave only *vertex* points; A_3 might *fill up hollow regions*, etc. (see Fig. 5). Each sequence $A_{i1}A_{i2} \ldots A_{ik}$ of these forms a new transformation, so that there is available an infinite variety. We provide the machine also with one or more "terminal" operations which convert a picture into a number, so that any sequence of the elementary transformations, followed by a terminal operation, defines a property. [Dineen (1955) describes how these processes were programmed in a digital computer.] We can start with a few short sequences, perhaps chosen randomly. Selfridge describes how the machine might learn new useful properties.

> *We now feed the machine A's and O's telling the machine each time which letter it is. Beside each sequence under the two letters, the machine builds up distribution functions from the results of applying the sequences to the image. Now, since the sequences were chosen completely randomly, it may well be that most of the sequences have very flat distribution functions; that is, they [provide] no information,*

[5] See pp. 160ff. of Wiener (1948).

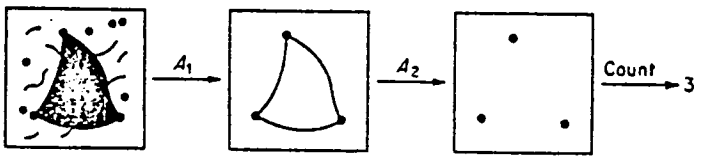


Figure 5. An arbitrary sequence of picture transformations, followed by a numerical-valued function, can be used as a *property* function for pictures. A_1 removes all points which are not at the edge of a solid region. A_2 leaves only vertex points—at which an arc suddenly changes direction. The function C simply counts the number of points remaining in the picture. All remarks in the text could be generalized to apply to properties like A_1A_2C, which can have more than two values.

and the sequences are therefore [by definition] not significant. Let it discard these and pick some others. Sooner or later, however, some sequences will prove significant; that is, their distribution functions will peak up somewhere. What the machine does now is to build up new sequences like the significant ones. This is the important point. If it merely chose sequences at random it might take a very long while indeed to find the best sequences. But with some successful sequences, or partly successful ones, to guide it, we hope that the process will be much quicker. The crucial question remains: how do we build up sequences "like" other sequences, but not identical? As of now we think we shall merely build sequences from the transition frequencies of the significant sequences. We shall build up a matrix of transition frequencies from the significant ones, and use those as transition probabilities with which to choose new sequences.

We do not claim that this method is necessarily a very good way of choosing sequences—only that it should do better than not using at all the knowledge of what kind of sequences has worked. It has seemed to us that this is the crucial point of learning.[6]

It would indeed be remarkable if this failed to yield properties more useful than would be obtained from completely random sequence selection. The generating problem is discussed further in Minsky (1956*a*). Newell, Shaw, and Simon (1960*b*) describe more deliberate, less statistical, techniques that might be used to discover sets of properties appropriate to a given problem area. One may think of the Selfridge proposal as a system which uses a finite-state language to describe its properties. Solomonoff (1957, 1960) proposes some techniques for discovering common features of a set of expressions, *e.g.*, of the descriptions of those properties of already established utility; the methods can then be applied to generate new properties with the same common features. I consider the lines of attack in Selfridge (1955), Newell, Shaw and Simon (1960*a*), and Solomonoff (1960, 1958), although still incomplete, to be of the greatest importance.

G. Combining Properties

One cannot expect easily to find a *small* set of properties which will be just right for a problem area. It is usually much easier to find a large set of properties each of which provides a little useful information. Then one is faced with the problem of finding a way to combine them to make the desired distinctions. The simplest method is to choose, for each class, a typical character (a particular sequence of property values) and then to use some matching procedure, *e.g.*, counting the numbers of agreements and disagreements, to compare an unknown with these chosen "Character

[6] See p. 93 of Selfridge (1955).

prototypes." The linear weighting scheme described just below is a slight generalization on this. Such methods treat the properties as more or less independent evidence for and against propositions; more general procedures (about which we have yet little practical information) must account also for nonlinear relations between properties, *i.e.*, must contain weighting terms for joint subsets of property values.

1. "BAYES NETS" FOR COMBINING INDEPENDENT PROPERTIES

We consider a single experiment in which an object is placed in front of a property-list machine. Each property E_i will have a value, 0 or 1. Suppose that there has been defined some set of "object classes" F_j, and that we want to use the outcome of this experiment to decide in which of these classes the object belongs.

Assume that the situation is basically probabilistic, and that we know the probability p_{ij} that, if the object is in class F_j then the ith property E_i will have value 1. Assume further that these properties are independent; that is, even given F_j, knowledge of the value of E_i tells us nothing more about the value of a different E_k in the same experiment. (This is a strong condition—see below.) Let ϕ_j be the absolute probability that an object is in class F_j. Finally, for this experiment define V to be the particular set of i's for which the E_i's are 1. Then this V represents the Character of the object. From the definition of conditional probability, we have

$$\Pr(F_j,V) = \Pr(V) \cdot \Pr(F_j|V) = \Pr(F_j) \cdot \Pr(V|F_j)$$

Given the Character V, we want to guess which F_j has occurred (with the least chance of being wrong—the so-called *maximum likelihood* estimate); that is, for which j is $\Pr(F_j|V)$ the largest? Since in the above $\Pr(V)$ does not depend on j, we have only to calcuate for which j is

$$\Pr(F_j) \cdot \Pr(V|F_j) = \phi_j \Pr(V|F_j)$$

the largest. Hence, by our independence hypothesis, we have to maximize

$$\phi_j \cdot \prod_{i \in V} p_{ij} \cdot \prod_{i \in \bar{V}} q_{ij} = \phi_j \prod_{i \in V} \frac{p_{ij}}{q_{ij}} \cdot \prod_{\text{all } i} q_{ij} \qquad (1)$$

These "maximum-likelihood" decisions can be made (Fig. 6) by a simple network device.[7]

[7] At the cost of an additional network layer, we may also account for the possible cost g_{jk} that would be incurred if we were to assign to F_k a figure really in class F_j; in this case the minimum cost decision is given by the k for which

$$\sum_j g_{jk}\phi_j \prod_{i \in V} p_{ij} \prod_{i \in \bar{V}} q_{ij}$$

is the least. $\bar{V}$ is the complement set to V. q_{ij} is $(1 - p_{ij})$.

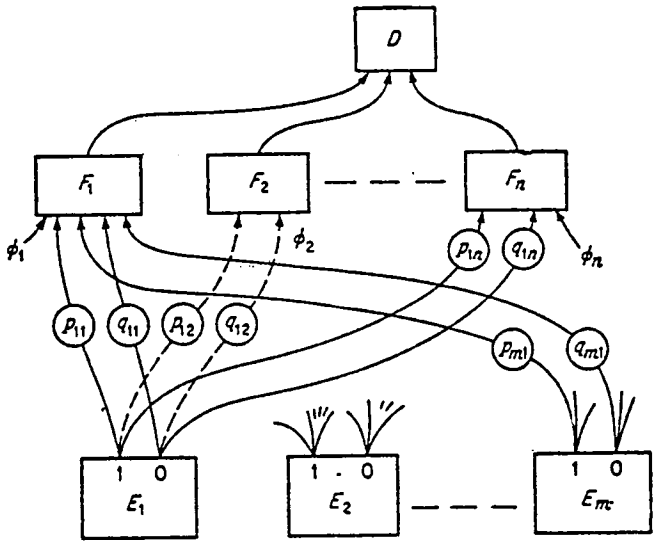


Figure 6. "Net" model for maximum-likelihood decisions based on linear weightings of property values. The input data are examined by each "property filter" E_i. Each E_i has "0" and "1" output channels, one of which is excited by each input. These outputs are weighted by the corresponding p_{ij}'s, as shown in the text. The resulting signals are multiplied in the F_j units, each of which "collects evidence" for a particular figure class. [We could have used here log (p_{ij}), and *added* at the F_j units.] The final decision is made by the topmost unit D, who merely chooses that F_j with the largest score. Note that the logarithm of the coefficient p_{ij}/q_{ij} in the second expression of (1) can be construed as the "weight of the evidence" of E_i in favor of F_j. [See also Papert (1961) and Rosenblatt (1958).]

These nets resemble the general schematic diagrams proposed in the "Pandemonium" model of Selfridge (1959) (see his fig. 3). It is proposed there that some intellectual processes might be carried out by a hierarchy of simultaneously functioning submachines suggestively called "demons." Each unit is set to detect certain patterns in the activity of others and the output of each unit announces the degree of confidence of that unit that it sees what it is looking for. Our E_i units are Selfridge's "data demons." Our units F_j are his "cognitive demons"; each collects from the abstracted data evidence for a specific proposition. The topmost "decision demon" D responds to that one in the multitude below it whose shriek is the loudest.[8]

It is quite easy to add to this "Bayes network model" a mechanism which will enable it to *learn* the optimal connection weightings. Imagine that, after each event, the machine is told which F_j has occurred; we could implement this by sending back a signal along the connections leading to that F_j unit. Suppose that the connection for p_{ij} (or q_{ij}) contains a two-terminal device (or "synapse") which stores a number w_{ij}. Whenever the joint event $(F_j, E_i = 1)$ occurs, we modify w_{ij} by replacing it by

[8] See also the report in Selfridge and Neisser (1960).

$(w_{ij} + 1)\theta$, where θ is a factor slightly less than unity. And when the joint event $(F_j, E_i = 0)$ occurs, we decrement w_{ij} by replacing it with $(w_{ij})\theta$. It is not difficult to show that the expected values of the w_{ij}'s will become proportional to the p_{ij}'s [and, in fact, approach $p_{ij}[\theta/(1 - \theta)]$. Hence, the machine tends to learn the optimal weighting on the basis of experience. (One must put in a similar mechanism for estimating the ϕ_j's.) The variance of the normalized weight $w_{ij}[(1 - \theta)/\theta]$ approaches $[(1 - \theta)/(1 + \theta]p_{ij}q_{ij}$. Thus a small value for θ means rapid learning but is associated with a large variance, hence, with low reliability. Choosing θ close to unity means slow, but reliable, learning. θ is really a sort of memory decay constant, and its choice must be determined by the noise and stability of the environment—much noise requires long averaging times, while a changing environment requires fast adaptation. The two requirements are, of course, incompatible and the decision has to be based on an economic compromise.[9]

2. POSSIBILITIES OF USING RANDOM NETS FOR BAYES DECISIONS

The nets of Fig. 6 are very orderly in structure. Is all this structure necessary? Certainly if there were a great many properties, *each of which provided very little marginal information,* some of them would not be missed. Then one might expect good results with a mere sampling of all the possible connection paths w_{ij}. And one might thus, *in this special situation,* use a random connection net.

The two-layer nets here resemble those of the "Perceptron" proposal of Rosenblatt (1958). In the latter, there is an additional level of connections coming directly from randomly selected points of a "retina." Here the properties, the devices which abstract the visual input data, are simple functions which add some inputs, subtract others, and detect whether the result exceeds a threshold. Equation (1), we think, illustrates what is of value in this scheme. It does seem clear that a maximum-likelihood type of analysis of the output of the property functions can be handled by such nets. But these nets, with their simple, randomly generated, connections can probably never achieve recognition of such patterns as "the class of figures having two separated parts," and they cannot even achieve the effect of template recognition without size and position normalization (unless sample figures have been presented previously in essentially all sizes and positions). For the chances are extremely small of finding, by random methods, enough properties usefully correlated with patterns appreciably more abstract than those of the prototype-derived kind. And these networks can really only separate out (by weighting) information in the individual input properties; they cannot extract further information present in nonadditive form. The "Perceptron" class of machines have facilities neither for obtaining better-than-chance properties nor for assembling

[9] See also Minsky and Selfridge (1960), and Papert (1961).

better-than-additive combinations of those it gets from random construction.[10]

For recognizing *normalized* printed or hand-printed characters, single-point properties do surprisingly well (Highleyman and Kamentsky, 1960); this amounts to just "averaging" many samples. Bledsoe and Browning (1959) claim good results with point-pair properties. Roberts (1960) describes a series of experiments in this general area. Doyle (1959) without normalization but with quite sophisticated properties obtains excellent results; his properties are already substantially size- and position-invariant. A general review of Doyle's work and other pattern-recognition experiments will be found in Selfridge and Neisser (1960).

For the complex discrimination, *e.g.*, between one and two connected objects, the property problem is very serious, especially for long wiggly objects such as are handled by Kirsch (1957). Here some kind of recursive processing is required and combinations of simple properties would almost certainly fail even with large nets and long training.

We should not leave the discussion of some decision net models without noting their important limitations. The hypothesis that, for given j, the p_{ij} represent independent events, is a very strong condition indeed. Without this hypothesis we could still construct maximum-likelihood nets, but we would need an additional layer of cells to represent all of the joint events V; that is, we would need to know all the $\Pr(F_j|V)$. This gives a general (but trivial) solution, but requires 2^n cells for n properties, which is completely impractical for large systems. What is required is a system which computes some sampling of all the joint conditional probabilities, and uses these to estimate others when needed. The work of Uttley (1956, 1959) bears on this problem, but his proposed and experimental devices do not yet clearly show how to avoid exponential growth.[11]

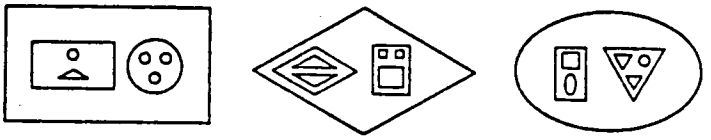
H. Articulation and Attention—Limitations of the Property-list Method

Because of its fixed size, the property-list scheme is limited (for any given set of properties) in the detail of the distinctions it can make. Its ability to deal with a compound scene containing several objects is critically weak, and its direct extensions are unwieldy and unnatural. If a machine can recognize a chair and a table, it surely should be able to tell us that "there is a chair and a table." To an extent, we can invent properties which allow some capacity for superposition of object Characters.[12] But there is no way to escape the information limit.

[10] See also Roberts (1960), Papert (1961), and Hawkins (1958). We can find nothing resembling an analysis [see (1) above] in Rosenblatt (1958) or his subsequent publications.

[11] See also Papert (1961).

[12] Cf. Mooers' technique of Zatocoding (1956*a*, 1956*b*).



(a) (b) (c)

Figure 7. The picture (*a*) is first described verbally in the text. Then, by introducing notation for the relations "inside of," "to the left of" and "above," we construct a symbolic description. Such descriptions can be formed and manipulated by machines. By abstracting out of the complex relation between the parts of the figure we can use the same formula to describe the related pictures (*b*) and (*c*), changing only the list of primitive parts. It is up to the programmer to decide at just what level of complexity a part of a picture should be considered "primitive"; this will depend on what the description is to be used for. We could further divide the drawings into vertices, lines, and arcs. Obviously, for some applications the relations would need more metrical information, *e.g.*, specification of lengths or angles.

What is required is clearly (1) a *list* (*of whatever length is necessary*) of the primitive objects in the scene and (2) a statement about the relations among them. Thus we say of Fig. 7*a*, "A rectangle (1) contains two subfigures disposed horizontally. The part on the left is a rectangle (2) which contains two subfigures disposed vertically; the upper a circle (3) and the lower a triangle (4). The part on the right . . . etc." Such a description entails an ability to separate or "articulate" the scene into parts. (Note that in this example the articulation is essentially *recursive;* the figure is first divided into two parts; then each part is described using the same machinery.) We can formalize this kind of description in an expression language whose fundamental grammatical form is a pair (R,L) whose first member R names a *relation* and whose second member L is an *ordered list* $(x_1,x_2, \ldots ,x_n)$ of the objects or subfigures which bear that relation to one another. We obtain the required flexibility by allowing the members of the list L to contain not only the names of "elementary" figures but also "subexpressions" of the form (R,L) designating complex subfigures. Then our scene above may be described by the expression

$$[\odot, (\square, (\rightarrow, \{(\odot, (\square, (\downarrow, (\bigcirc, \triangle)))), (\odot, (\bigcirc, (\triangledown, (\bigcirc, \ \bigcirc, \bigcirc))))\})]$$

where $(\odot, (x,y))$ means that y is contained in x; $(\rightarrow,(x,y))$ means that y is to the right of x; $(\downarrow,(x,y))$ means that y is below x, and $(\triangle,(x,y,z))$ means that y is to the right of x and z is underneath and between them. The symbols $\square$, $\bigcirc$, and $\triangle$ represent the indicated kinds of primitive geometric objects. This expression-pair description language may be regarded as a simple kind of "list-structure" language. Powerful computer techniques have been developed, originally by Newell, Shaw and Simon,

for manipulating symbolic expressions in such languages for purposes of heuristic programming. (See the remarks at the end of Sec. IV. If some of the members of a list are themselves lists, they must be surrounded by exterior parentheses, and this accounts for the accumulation of parentheses.)

It may be desirable to construct descriptions in which the complex relation is extracted, *e.g.*, so that we have an expression of the form FG where F is an expression which at once denotes the composite relation between all the primitive parts listed in G. A complication arises in connection with the "binding" of variables, *i.e.*, in specifying the manner in which the elements of G participate in the relation F. This can be handled in general by the "λ" notation (McCarthy, 1960) but here we can just use integers to order the variables.

For the given example, we could describe the relational part F by an expression

$$\odot(1,\rightarrow(\odot(2,\downarrow(3,4)),\odot(5,\nabla(6,7,8))))$$

in which we now use a "functional notation"; "$(\odot, (x,y))$" is replaced by "$\odot\,(x,y)$," etc., making for better readability. To obtain the desired description, this expression has to be applied to an ordered list of primitive objects, which in this case is $(\square,\square,\bigcirc,\triangle,\bigcirc,\bigcirc,\bigcirc,\bigcirc)$. This composite functional form allows us to abstract the composite relation. By changing only the object list we can obtain descriptions also of the objects in Fig. 7*b* and *c*.

The important thing about such "articular" descriptions is that they can be obtained by *repeated application of a fixed set of pattern-recognition techniques*. Thus we can obtain *arbitrarily complex* descriptions from a fixed complexity classification mechanism. The new element required in the mechanism (beside the capacity to manipulate the list structures) is the ability to articulate—to "attend fully" to a selected part of the picture and bring all one's resources to bear on that part. In efficient problem-solving programs, we will not usually complete such a description in a single operation. Instead, the depth or detail of description will be under the control of other processes. These will reach deeper, or look more carefully, only when they have to, *e.g.*, when the presently available description is inadequate for a current goal. The author, together with L. Hodes, is working on pattern-recognition schemes using articular descriptions. By manipulating the formal descriptions we can deal with overlapping and incomplete figures, and several other problems of the "Gestalt" type.

It seems likely that as machines are turned toward more difficult problem areas, *passive* classification systems will become less adequate, and we may have to turn toward schemes which are based more on internally

generated hypotheses, perhaps "error-controlled" along the lines proposed by MacKay (1956).

Space requires us to terminate this discussion of pattern-recognition and description. Among the important works not reviewed here should be mentioned those of Bomba (1959) and Grimsdale *et al.* (1959), which involve elements of description, Unger (1959) and Holland (1960) for parallel processing schemes, Hebb (1949) who is concerned with physiological description models, and the work of the Gestalt psychologists, notably Kohler (1947), who have certainly raised, if not solved, a number of important questions. Sherman (1959), Haller (1959) and others have completed programs using line-tracing operations for topological classification. The papers of Selfridge (1955, 1956) have been a major influence on work in this general area.

See also Kirsch *et al.* (1957) for discussion of a number of interesting computer image processing techniques, and see Minot (1959) and Stevens (1957) for reviews of the reading machine and related problems. One should also examine some biological work, *e.g.*, Tinbergen (1951) to see instances in which some discriminations which seem, at first glance very complicated are explained on the basis of a few apparently simple properties arranged in simple decision trees.

III. Learning Systems

In order to solve a new problem, one should first try using methods similar to those that have worked on similar problems. To implement this "basic learning heuristic" one must generalize on past experience, and one way to do this is to use success-reinforced decision models. These learning systems are shown to be averaging devices. Using devices which learn also which events are associated with reinforcement, *i.e.*, reward, we can build more autonomous "secondary reinforcement" systems. In applying such methods to complex problems, one encounters a serious difficulty—in distributing credit for success of a complex strategy among the many decisions that were involved. This problem can be managed by arranging for local reinforcement of partial goals within a hierarchy, and by grading the training sequence of problems to parallel a process of maturation of the machine's resources.

In order to solve a new problem one uses what might be called the basic learning heuristic—first try using methods similar to those which have worked, in the past, on similar problems. We want our machines, too, to benefit from their past experience. Since we cannot expect new situations to be precisely the same as old ones, any useful learning will have to involve generalization techniques. There are too many notions associated

with "learning" to justify defining the term precisely. But we may be sure that any useful learning system will have to use records of the past as *evidence for more general propositions;* it must thus entail some commitment or other about "inductive inference." (See Sec. V*B*.) Perhaps the simplest way of generalizing about a set of entities is through constructing a new one which is an "ideal," or rather, a typical member of that set; the usual way to do this is to smooth away variation by some sort of averaging technique. And indeed we find that most of the *simple* learning devices do incorporate some averaging technique—often that of averaging some sort of product, thus obtaining a sort of correlation. We shall discuss this family of devices here, and some more abstract schemes in Sec. V.

A. Reinforcement

A reinforcement process is one in which some aspects of the behavior of a system are caused to become more (or less) prominent in the future as a consequence of the application of a "reinforcement operator" Z. This operator is required to affect only those aspects of behavior for which instances have actually occurred recently.

The analogy is with "reward" or "extinction" (not punishment) in animal behavior. The important thing about this kind of process is that it is "operant" [a term of Skinner (1953)]; the reinforcement operator does not initiate behavior, but merely selects that which the Trainer likes from that which has occurred. Such a system must then contain a device M which generates a variety of behavior (say, in interacting with some environment) and a Trainer who makes critical judgments in applying the available reinforcement operators. (See Fig. 8.)

Let us consider a very simple reinforcement model. Suppose that on each presentation of a stimulus S an animal has to make a choice, *e.g.*, to

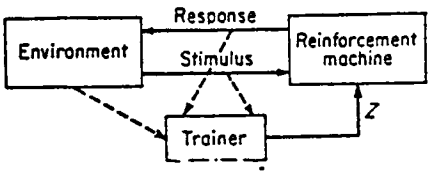


Figure 8. Parts of an "operant reinforcement" learning system. In response to a stimulus from the environment, the machine makes one of several possible responses. It remembers what decisions were made in choosing this response. Shortly thereafter, the Trainer sends to the machine positive or negative reinforcement (reward) signal; this increases or decreases the tendency to make the same decisions in the future. Note that the Trainer need not know how to solve problems, but only how to detect success or failure, or relative improvement; his function is selective. The Trainer might be connected to observe the actual stimulus-response activity, or, in a more interesting kind of system, just some function of the state of the environment.

turn left or right, and that its probability of turning right, at the nth trial, is p_n. Suppose that *we* want it to turn right. Whenever it does this we might "reward" it by applying the operator Z_+;

$$p_{n+1} = Z_+(p_n) = \theta p_n + (1 - \theta) \qquad 0 < \theta < 1$$

which moves p a fraction $(1 - \theta)$ of the way toward unity.[13] If we dislike what it does we apply negative reinforcement,

$$p_{n+1} = Z_-(p_n) = \theta p_n$$

moving p the same fraction of the way toward 0. Some theory of such "linear" learning operators, generalized to several stimuli and responses, will be found in Bush and Mosteller (1955). We can show that the learning result is an average weighted by an exponentially-decaying time factor: Let Z_n be ± 1 according to whether the nth event is rewarded or extinguished and replace p_n by $c_n = 2p_n - 1$ so that $-1 \leq c_n \leq 1$, as for a correlation coefficient. Then (with $c_0 = 0$) we obtain by induction

$$c_{n+1} = (1 - \theta) \sum_{i=0}^{n} \theta^{n-i} Z_i$$

and since

$$\frac{1}{1 - \theta} \approx \sum_{0}^{n} \theta^{n-i}$$

we can write this as

$$c_{n+1} \approx \frac{\Sigma \theta^{n-i} Z_i}{\Sigma \theta^{n-i}} \tag{1}$$

If the term Z_i is regarded as a product of (i) how the creature responded and (ii) which kind of reinforcement was given, then c_n is a kind of correlation function (with the decay weighting) of the joint behavior of these quantities. The ordinary, uniformly weighted average has the same general form but with time-dependent θ:

$$c_{n+1} = \left(1 - \frac{1}{N}\right) c_n + \frac{1}{N} Z_n \tag{2}$$

In (1) we have again the situation described in Sec. IIG1; a small value of θ gives fast learning, and the possibility of quick adaptation to a changing environment. A near-unity value of θ gives slow learning, but also smooths away uncertainties due to noise. As noted in Sec. IIG1, the response distribution comes to approximate the probabilities of rewards of the alternative responses. (The importance of this phenomenon has, I think, been overrated; it is certainly not an especially rational strategy. One reasonable alternative is that of computing the numbers p_{ij} as indi-

[13] Properly, the reinforcement functions should depend both on the p's and on the previous reaction—reward should decrease p if our animal has just turned to the left. The notation in the literature is also somewhat confusing in this regard.

cated, but actually playing at each trial the "most likely" choice. Except in the presence of a hostile opponent, there is usually no reason to play a "mixed" strategy.[14])

In Samuel's coefficient-optimizing program (1959*b*) [see Sec. IIIC1], there is a most ingenious compromise between the exponential and the uniform averaging methods: the value of N in (2) above begins at 16 and so remains until $n = 16$, then N is 32 until $n = 32$, and so on until $n = 256$. Thereafter N remains fixed at 256. This nicely prevents violent fluctuations in c_n at the start, approaches the uniform weighting for a while, and finally approaches the exponentially weighted correlation, all in a manner that requires very little computation effort! Samuel's program is at present the outstanding example of a game-playing program which matches average human ability, and its success (in real time) is attributed to a wealth of such elegancies, both in heuristics and in programming.

The problem of extinction or "unlearning" is especially critical for complex, hierarchical, learning. For, once a generalization about the past has been made, one is likely to build upon it. Thus, one may come to select certain properties as important and begin to use them in the characterization of experience, perhaps storing one's memories in terms of them. If later it is discovered that some other properties would serve better, then one must face the problem of translating, or abandoning, the records based on the older system. This may be a very high price to pay. One does not easily give up an old way of looking at things, if the better one demands much effort and experience to be useful. Thus the *training sequences* on which our machines will spend their infancies, so to speak, must be chosen very shrewdly to insure that early abstractions will provide a good foundation for later difficult problems.

Incidentally, in spite of the space given here for their exposition, I am not convinced that such "incremental" or "statistical" learning schemes should play a central role in our models. They will certainly continue to appear as components of our programs but, I think, mainly by default. The more intelligent one is, the more often he should be able to learn from an experience something rather definite; *e.g.*, to reject or accept a hypothesis, or to change a goal. (The obvious exception is that of a truly statistical environment in which averaging is inescapable. But the heart of problem-solving is always, we think, the combinatorial part that gives rise to searches, and we should usually be able to regard the complexities caused by "noise" as mere annoyances, however irritating they may be.) In this connection we can refer to the discussion of memory in Miller, Galanter and Pribram (1960).[15] This seems to be the first major work

[14] The question of just how often one should play a strategy different from the estimated optimum, in order to gain information, is an underlying problem in many fields. See, *e.g.*, Shubik (1960).

[15] See especially chap. 10.

in psychology to show the influence of work in the artificial intelligence area, and its programme is generally quite sophisticated.

B. Secondary Reinforcement and Expectation Models

The simple reinforcement system is limited by its dependence on the Trainer. If the Trainer can detect only the *solution* of a problem, then we may encounter "mesa" phenomena which will limit performance on difficult problems. (See Sec. IC.) One way to escape this is to have the machine learn to generalize on what the Trainer does. Then, in difficult problems, it may be able to give itself partial reinforcements along the way, *e.g.*, upon the solution of relevant subproblems. The machine in Fig. 9 has some such ability. The new unit U is a device that learns which external stimuli are strongly correlated with the various reinforcement signals, and responds to such stimuli by reproducing the corresponding reinforcement signals. (The device U is *not* itself a reinforcement learning device; it is more like a "Pavlovian" conditioning device, treating the Z signals as "unconditioned" stimuli and the S signals as conditioned stimuli.) The heuristic idea is that any signal from the environment which in the past has been well correlated with (say) positive reinforcement is likely to be an indication that something good has just happened. If the training on early problems was such that this is realistic, then the system eventually should be able to detach itself from the Trainer, and become autonomous. If we further permit "chaining" of the "secondary reinforcers," *e.g.*, by admitting the connection shown as a dotted line in Fig. 9, the scheme becomes quite powerful, in principle. There are obvious pitfalls in admitting

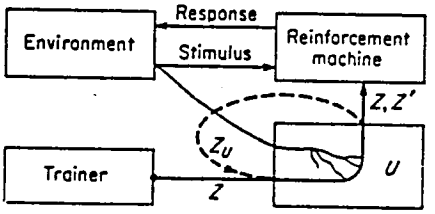


Figure 9. An additional device U gives the machine of Fig. 8 the ability to learn which signals from the environment have been associated with reinforcement. The primary reinforcement signals Z are routed through U. By a Pavlovian conditioning process (not described here), external signals come to produce reinforcement signals like those that have frequently succeeded them in the past. Such signals might be abstract, *e.g.*, verbal encouragement. If the "secondary reinforcement" signals are allowed, in turn, to acquire further external associations (through, *e.g.*, a channel Z_U as shown) the machine might come to be able to handle chains of subproblems. But something must be done to stabilize the system against the positive symbolic feedback loop formed by the path Z_U. The profound difficulty presented by this stabilization problem may be reflected in the fact that, in lower animals, it is very difficult to demonstrate such chaining effects.

such a degree of autonomy; the values of the system may drift to a "non-adaptive" condition.

C. Prediction and Expectation

The evaluation unit U is supposed to acquire an ability to tell whether a situation is good or bad. This evaluation could be applied to *imaginary* situations as well as to real ones. If we could estimate the consequences of a proposed action (without its actual execution), we could use U to evaluate the (estimated) resulting situation. This could help in reducing the effort in search, and we would have in effect a machine with some ability to look ahead, or *plan*. In order to do this we need an additional device P which, given the description of a situation and an action, will predict a description of the likely result. (We will discuss schemes for doing this in Sec. IV*C*.) The device P might be constructed along the lines of a reinforcement learning device. In such a system the required reinforcement signals would have a very attractive character. For the machine must reinforce P positively when the *actual outcome resembles that which was predicted*—accurate expectations are rewarded. If we could further add a premium to reinforcement of those predictions which have a novel aspect, we might expect to discern behavior motivated by a sort of curiosity. In the reinforcement of mechanisms for confirmed novel expectations (or new explanations) we may find the key to simulation of intellectual motivation.[16]

SAMUEL'S PROGRAM FOR CHECKERS

In Samuel's "generalization learning" program for the game of checkers (1959*a*) we find a novel heuristic technique which could be regarded as a simple example of the "expectation reinforcement" notion. Let us review very briefly the situation in playing two-person board games of this kind. As noted by Shannon (1956) such games are in principle finite, and a best strategy can be found by following out all possible continuations—if he goes there I can go there, or there, etc.—and then "backing up" or "minimaxing" from the terminal positions, won, lost, or drawn. But in practice the full exploration of the resulting colossal "move tree" is out of the question. No doubt, some exploration will always be necessary for such games. But the tree must be pruned. We might simply put a limit on depth of exploration—the number of moves and replies. We might also limit the number of alternatives explored from each position—this requires some heuristics for selection of "plausible moves."[17] Now, if the backing-up technique is still to be used (with the incomplete move tree) one has to

[16] See also chap. 6 of Minsky (1954).

[17] See the discussion of Bernstein (1958) and the more extensive review and discussion in the very suggestive paper of Newell, Shaw and Simon (1958*b*).

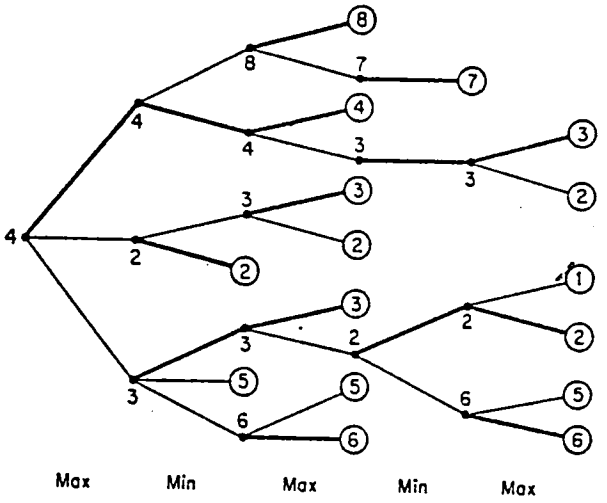


Figure 10. "Backing up" the static evaluations of proposed moves in a game tree. From the vertex at the left, representing the present position in a board game, radiate three branches, representing the *player's* proposed moves. Each of these might be countered by a variety of *opponent* moves, and so on. According to some program, a finite tree is generated. Then the worth to the player of each terminal board position is estimated (see text). If the opponent has the same values, he will choose to minimize the score, while the player will always try to maximize. The heavy lines show how this minimaxing process backs up until a choice is determined for the present position.

The full tree for chess has the order of $.10^{120}$ branches—beyond the reach of any man or computer. There is a fundamental heuristic exchange between the effectiveness of the evaluation function and the extent of the tree. A very weak evaluation (*e.g.*, one which just compares the players' values of pieces) would yield a devastating game if the machine could explore all continuations out to, say, 20 levels. But only 6 levels, roughly within the range of our presently largest computers, would probably not give a brilliant game; less exhaustive strategies, perhaps along the lines of Newell, Shaw, and Simon (1958*b*), would be more profitable.

substitute for the absolute "win, lose, or draw" criterion some other "static" way of evaluating nonterminal positions.[18] (See Fig. 10.) Perhaps the simplest scheme is to use a weighted sum of some selected set of "property" functions of the positions—mobility, advancement, center control, and the like. This is done in Samuel's program, and in most of its predecessors. Associated with this is a multiple-simultaneous-optimizer method for discovering a good coefficient assignment (using the correlation technique noted in Sec. III*A*). But the source of reinforcement signals in

[18] In some problems the backing-up process can be handled in closed analytic form so that one may be able to use such methods as Bellman's "Dynamic Programming" (1957). Freimer (1960) gives some examples for which limited "look-ahead" doesn't work.

Samuel (1959*a*) is novel. One cannot afford to play out one or more entire games for each single learning step. Samuel measures instead *for each move* the difference between what the evaluation function yields *directly* of a position and what it *predicts* on the basis of an extensive continuation exploration, *i.e.*, backing up. The sign of this error, "Delta," is used for reinforcement; thus the system may learn something at *each move*.[19]

D. The Basic Credit-assignment Problem for Complex Reinforcement Learning Systems

In playing a complex game such as chess or checkers, or in writing a computer program, one has a definite success criterion—the game is won or lost. But in the course of play, each ultimate success (or failure) is associated with a vast number of internal decisions. If the run is successful, how can we assign credit for the success among the multitude of decisions? As Newell noted,

> *It is extremely doubtful whether there is enough information in "win, lose, or draw" when referred to the whole play of the game to permit any learning at all over available time scales. . . . For learning to take place, each play of the game must yield much more information. This is . . . achieved by breaking the problem into components. The unit of success is the goal. If a goal is achieved, its subgoals are reinforced; if not they are inhibited. (Actually, what is reinforced is the transformation rule that provided the subgoal.) . . . This also is true of the other kinds of structure: every tactic that is created provides information about the success or failure of tactic search rules; every opponent's action provides information about success or failure of likelihood inferences; and so on. The amount of information relevant to learning increases directly with the number of mechanisms in the chess-playing machine.*[20]

We are in complete agreement with Newell on this approach to the problem.[21]

It is my impression that many workers in the area of "self-organizing" systems and "random neural nets" do not feel the urgency of this prob-

[19] It should be noted that Samuel (1959*a*) describes also a rather successful checker-playing program based on recording and retrieving information about positions encountered in the past, a less abstract way of exploiting past experience. Samuel's work is notable in the variety of experiments that were performed, with and without various heuristics. This gives an unusual opportunity to really find out how different heuristic methods compare. More workers should choose (other things being equal) problems for which such variations are practicable.

[20] See p. 108 of Newell (1955).

[21] See also the discussion in Samuel (p. 22, 1959*a*) on assigning credit for a change in "Delta."

lem. Suppose that one million decisions are involved in a complex task (such as winning a chess game). Could we assign to each decision element one-millionth of the credit for the completed task? In certain special situations we can do just this—*e.g.*, in the machines of Rosenblatt (1958), Roberts (1960), and Farley and Clark (1954), etc., where the connections being reinforced are to a sufficient degree independent. But the problem-solving ability is correspondingly weak.

For more complex problems, with decisions in hierarchies (rather than summed on the same level) and with increments small enough to assure probable convergence, the running times would become fantastic. For complex problems we will have to define "success" in some rich local sense. Some of the difficulty may be evaded by using carefully graded "training sequences" as described in the following section.

FRIEDBERG'S PROGRAM-WRITING PROGRAM

An important example of comparative failure in this credit-assignment matter is provided by the program of Friedberg (1958, 1959) to solve program-writing problems. The problem here is to write programs for a (simulated) very simple digital computer. A simple problem is assigned, *e.g.*, "compute the AND of two bits in storage and put the result in an assigned location." A generating device produces a random (64-instruction) program. The program is run and its success or failure is noted. The success information is used to reinforce *individual instructions* (in fixed locations) so that each success tends to increase the chance that the instructions of successful programs will appear in later trials. (We lack space for details of how this is done.) Thus the program tries to find "good" instructions, more or less independently, for each location in program memory. The machine did learn to solve some extremely simple problems. But it took of the order of 1000 times longer than pure chance would expect. In part II of Friedberg *et al.* (1959), this failure is discussed, and attributed in part to what we called (Sec. IC) the "Mesa phenomena." In changing just one instruction at a time the machine had not taken large enough steps in its search through program space.

The second paper goes on to discuss a sequence of modifications in the program generator and its reinforcement operators. With these, and with some "priming" (starting the machine off on the right track with some useful instructions), the system came to be only a little worse than chance. Friedberg *et al.* (1959) conclude that with these improvements "the generally superior performance of those machines with a success-number reinforcement mechanism over those without does serve to indicate that such a mechanism can provide a basis for constructing a learning machine." I disagree with this conclusion. It seems to me that each of the "improvements" can be interpreted as serving only to increase the step

size of the search, that is, the randomness of the mechanism; this helps to avoid the Mesa phenomenon and thus approach chance behavior. But it certainly does not show that the "learning mechanism" is working—one would want at least to see some better-than-chance results before arguing this point. The trouble, it seems, is with credit-assignment. The credit for a working program can only be assigned to functional groups of instructions, *e.g.*, subroutines, and as these operate in hierarchies we should not expect individual instruction reinforcement to work well.[22] It seems surprising that it was not recognized in Friedberg *et al.* (1959) that the doubts raised earlier were probably justified! In the last section of Friedberg *et al.* (1959) we see some real success obtained by breaking the problem into parts and solving them sequentially. (This successful demonstration using division into subproblems does not use any reinforcement mechanism at all.) Some experiments of similar nature are reported in Kilburn, Grimsdale and Sumner (1959).

It is my conviction that no scheme for learning, or for pattern recognition, can have very general utility unless there are provisions for recursive, or at least hierarchical, use of previous results. We cannot expect a learning system to come to handle very hard problems without preparing it with a reasonably graded sequence of problems of growing difficulty. The first problem must be one which can be solved in reasonable time with the initial resources. The next must be capable of solution in reasonable time by using reasonably simple and accessible combinations of methods developed in the first, and so on. The only alternatives to this use of an adequate "training sequence" are (1) advanced resources, given initially, or (2) the fantastic exploratory processes found perhaps only in the history of organic evolution.[23] And even there, if we accept the general view of Darlington (1958) who emphasizes the heuristic aspects of genetic systems, we must have developed early (in, *e.g.*, the phenomena of meiosis and crossing-over) quite highly specialized mechanisms providing for the segregation of groupings related to solutions of subproblems. Recently, much effort has been devoted to the construction of training sequences in connection with programming "teaching machines." Naturally, the psychological literature abounds with theories of how complex behavior is built

[22] See the introduction to Friedberg (1958) for a thoughtful discussion of the plausibility of the scheme.

[23] It should, however, be possible to construct learning mechanisms which can select for themselves reasonably good training sequences (from an always complex environment) by prearranging a relatively slow development (or "maturation") of the system's facilities. This might be done by prearranging that sequence of goals attempted by the primary Trainer match reasonably well, at each stage, the complexity of performance mechanically available to the pattern-recognition and other parts of the system. One might be able to do much of this by simply limiting the depth of hierarchical activity, perhaps only later permitting limited recursive activity.

up from simpler. In our own area, perhaps the work of Solomonoff (1957), while overly cryptic, shows the most thorough consideration of this dependency on *training sequences*.

IV. Problem-solving and Planning

The solution, by machine, of really complex problems will require a variety of administration facilities. During the course of solving a problem, one becomes involved with a large assembly of interrelated subproblems. From these, at each stage, a very few must be chosen for investigation. This decision must be based on (1) estimates of relative difficulties and (2) estimates of centrality of the different candidates for attention. Following subproblem selection (for which several heuristic methods are proposed), one must choose methods appropriate to the selected problems. But for really difficult problems, even these step-by-step heuristics for reducing search will fail, and the machine must have resources for analyzing the problem structure in the large—in short, for "planning." A number of schemes for planning are discussed, among them the use of models —analogous, semantic, and abstract. Certain abstract models, "Character-Algebras," can be constructed by the machine itself, on the basis of experience or analysis. For concreteness, the discussion begins with a description of a simple but significant system (LT) which encounters some of these problems.

A. The "Logic Theory" Program of Newell, Shaw and Simon

It is not surprising that the testing grounds for early work on mechanical problem-solving have usually been areas of mathematics, or games, in which the rules are defined with absolute clarity. The "Logic Theory" machine of Newell and Simon (1956*a*, 1957*a*), called "LT" below, was a first attempt to prove theorems in logic, by frankly heuristic methods. Although the program was not by human standards a brilliant success (and did not surpass its designers), it stands as a landmark both in heuristic programming and also in the development of modern automatic programming.

The problem domain here is that of discovering proofs in the Russell-Whitehead system for the propositional calculus. That system is given as a set of (five) axioms and (three) rules of inference; the latter specify how certain transformations can be applied to produce new theorems from old theorems and axioms.

The LT program is centered around the idea of "working backward" to find a proof. Given a theorem T to be proved, LT searches among the axioms and previously established theorems for one from which T can be deduced by a single application of one of three simple "Methods" (which

embody the given rules of inference). If one is found, the problem is solved. Or the search might fail completely. But finally, the search may yield one or more "problems" which are usually propositions from which T may be deduced directly. If one of these can, in turn, be proved a theorem the main problem will be solved. (The situation is actually slightly more complex.) Each such subproblem is adjoined to the "subproblem list" (after a limited preliminary attempt) and LT works around to it later. The full power of LT, such as it is, can be applied to each subproblem, for LT can use itself as a subroutine in a recursive fashion.

The heuristic technique of working backward yields something of a teleological process, and LT is a forerunner of more complex systems which construct hierarchies of goals and subgoals. Even so, the basic administrative structure of the program is no more than a nested set of searches through lists in memory. We shall first outline this structure and then mention a few heuristics that were used in attempts to improve performance.

1. Take the next problem from problem list.
   (If there are no more problems, EXIT with total failure.)
2. Choose the next of the three basic Methods.
   (If no more methods, go to 1.)
3. Choose the next member of the list of axioms and previous theorems.
   (If no more, go to 2.)
   Then apply the Method to the problem, using the chosen theorem or axiom.
   If problem is solved, EXIT with complete proof.
   If no result, go to 3.
   If new subproblem arises, go to 4.
4. Try the special (substitution) Method on the subproblem.
   If problem is solved, EXIT with complete proof.
   If no result, put the subproblem *at the end* of the problem list and go to 3.

Among the heuristics that were studied were (1) a *similarity test* to reduce the work in step 4 (which includes another search through the theorem list), (2) a *simplicity test* to select apparently easier problems from the problem list, and (3) a *strong nonprovability test* to remove from the problem list expressions which are probably false and hence not provable. In a series of experiments "learning" was used to find which earlier theorems had been most useful and should be given priority in step 3. We cannot review the effects of these changes in detail. Of interest was the balance between the extra cost for administration of certain heuristics and the resultant search reduction; this balance was quite delicate in some cases when computer memory became saturated. The system seemed to be

quite sensitive to the training sequence—the order in which problems were given. And some heuristics which gave no significant over-all improvement did nevertheless affect the class of solvable problems. Curiously enough, the general efficiency of LT was not greatly improved by any or all of these devices. But all this practical experience is reflected in the design of the much more sophisticated "GPS" system described briefly in Sec. IV*D*2.

Wang (1960) has criticized the LT project on the grounds that there exist, as he and others have shown, mechanized proof methods which, for the particular run of problems considered, use far less machine effort than does LT and which have the advantage that they will ultimately find a proof for any provable proposition. (LT does not have this exhaustive "decision procedure" character and can fail ever to find proofs for some theorems.) The authors of "Empirical Explorations of the Logic Theory Machine," perhaps unaware of the existence of even moderately efficient exhaustive methods, supported their arguments by comparison with a particularly inefficient exhaustive procedure. Nevertheless, I feel that some of Wang's criticisms are misdirected. He does not seem to recognize that the authors of LT are not so much interested in proving these theorems as they are in the general problem of solving difficult problems. The combinatorial system of Russell and Whitehead (with which LT deals) is far less simple and elegant than the system used by Wang.[21] [Note, *e.g.*, the emphasis in Newell, Shaw and Simon (1958*a*, 1958*b*).] Wang's problems, while *logically* equivalent, are *formally* much simpler. His methods do not include any facilities for using previous results (hence they are sure to degrade rapidly at a certain level of problem complexity), while LT is fundamentally oriented around this problem. Finally, because of the very effectiveness of Wang's method on the *particular* set of theorems in question, he simply did not have to face the fundamental heuristic problem of *when to decide to give up on a line of attack*. Thus the formidable performance of his program (1960) perhaps diverted his attention from heuristic problems that must again spring up when real mathematics is ultimately encountered.

This is not meant as a rejection of the importance of Wang's work and discussion. He and others working on "mechanical mathematics" have discovered that there are proof procedures which are much more efficient than has been suspected. Such work will unquestionably help in constructing intelligent machines, and these procedures will certainly be preferred, when available, to "unreliable heuristic methods." Wang, Davis and

[21] Wang's procedure (1960*a*), too, works backward, and can be regarded as a generalization of the method of "falsification" for deciding truth-functional tautology. In Wang (1960*b*) and its unpublished sequel, he introduces more powerful methods (for much more difficult problems).

Putnam, and several others are now pushing these new techniques into the far more challenging domain of theorem proving in the predicate calculus (for which exhaustive decision procedures are no longer available). We have no space to discuss this area,[25] but it seems clear that a program to solve real mathematical problems will have to combine the mathematical sophistication of Wang with the heuristic sophistication of Newell, Shaw and Simon.[26]

B. Heuristics for Subproblem Selection

In designing a problem-solving system, the programmer often comes equipped with a set of more or less distinct "Methods"—his real task is to find an efficient way for the program to decide where and when the different methods are to be used.

Methods which do not dispose of a problem may still transform it to create new problems or subproblems. Hence, during the course of solving one problem we may become involved with a large assembly of interrelated subproblems. A "parallel" computer, yet to be conceived, might work on many at a time. But even the parallel machine must have procedures to allocate its resources because it cannot simultaneously apply all its methods to all the problems. We shall divide this administrative problem into two parts: the selection of those subproblem(s) which seem most critical, attractive, or otherwise immediate, and, in the next section, the choice of which method to apply to the selected problem.

In the basic program for LT (Sec. IV*A*), subproblem selection is very simple. New problems are examined briefly and (if not solved at once) are placed at the end of the (linear) problem list. The main program proceeds along this list (step 1), attacking the problems in the order of their generation. More powerful systems will have to be more judicious (both in generation and selection of problems) for only thus can excessive branching be restrained.[27] In more complex systems we can expect to consider for each subproblem, at least these two aspects: (1) its apparent "centrality"—how will its solution promote the main goal, and (2) its apparent "difficulty"—how much effort is it liable to consume. We need heuristic methods to estimate each of these quantities and, further, to

[25] See Davis and Putnam (1960), and Wang (1960*b*).

[26] All these efforts are directed toward the reduction of search effort. In that sense they are all heuristic programs. Since practically no one still uses "heuristic" in a sense opposed to "algorithmic," serious workers might do well to avoid pointless argument on this score. The real problem is to find methods which significantly delay the apparently inevitable exponential growth of search trees.

[27] Note that the simple scheme of LT has the property that each generated problem will eventually get attention, even if several are created in a step 3. If one were to turn *full* attention to each problem, as generated, one might never return to alternate branches.

select accordingly one of the problems and allocate to it some reasonable quantity of effort.[29] Little enough is known about these matters, and so it is not entirely for lack of space that the following remarks are somewhat cryptic.

Imagine that the problems and their relations are arranged to form some kind of directed-graph structure (Minsky, 1956*b*; Newell and Simon, 1956*b*; Gelernter and Rochester, 1958). The main problem is to establish a "valid" path between two initially distinguished nodes. Generation of new problems is represented by the addition of new, not-yet-valid paths, or by the insertion of new nodes in old paths. Then problems are represented by not-yet-valid paths, and "centrality" by location in the structure. Associate with each connection, quantities describing its current validity state (solved, plausible, doubtful, etc.) and its current estimated difficulty.

1. GLOBAL METHODS

The most general problem-selection methods are "global"—at each step they look over the entire structure. There is one such simple scheme which works well on at least one rather degenerate interpretation of our problem graph. This is based on an electrical analogy suggested to us by a machine designed by Shannon [related to one described in Shannon (1955) which describes quite a variety of interesting game-playing and learning machines] to play a variant of the game marketed as "Hex" (and known among mathematicians as "Nash"). The initial board position can be represented as a certain network of resistors. (See Fig. 11.) One player's goal is to construct a *short-circuit* path between two given boundaries; the opponent tries to open the circuit between them. Each move consists of shorting (or opening), irreversibly, one of the remaining resistors. Shannon's machine applies a potential between the boundaries and selects that resistor which carries the largest current. Very roughly speaking, this resistor is likely to be most critical because changing it will have the largest effect on the resistance of the net and, hence, in the goal direction of shorting (or opening) the circuit. And although this argument is not perfect, nor is this a perfect model of the real combinatorial situation, the machine does play extremely well. (It can make unsound moves in certain artificial situations, but no one seems to have been able to force this during a game.)

The use of such a global method for problem selection requires that the available "difficulty estimates" for related subproblems be arranged to

[29] One will want to see if the considered problem is the same as one already considered, or very similar. See the discussion in Gelernter and Rochester (1958). This problem might be handled more generally by simply *remembering* the (Characters of) problems that have been attacked, and checking new ones against this memory, *e.g.*, by methods of Mooers (1956), looking more closely if there seems to be a match.

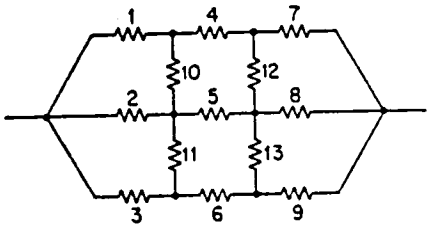


Figure 11. This board game (due to C. E. Shannon) is played on a network of equal resistors. The first player's goal is to open the circuit between the end points; the second player's goal is to short the circuit. A move consists of opening or shortening a resistor. If the first player begins by opening resistor 1, the second player might counter by shorting resistor 4, following the strategy described in the text. The remaining move pairs (if *both* players use that strategy) would be (5, 8) (9, 13) (12, 10 or 2) (2 or 10 *win*). In this game the first player should be able to force a win, and the maximum-current strategy seems always to do so, even on larger networks.

combine in roughly the manner of resistance values. Also, we could regard this machine as using an "analog model" for "planning." (See Sec. IV*D*.)[29]

2. LOCAL, AND "HEREDITARY," METHODS

The prospect of having to study at each step the whole problem structure is discouraging, especially since the structure usually changes only slightly after each attempt. One naturally looks for methods which merely *update* or modify a small fragment of the stored record. Between the extremes of the "first-come-first-served" problem-list method and the full global-survey methods, lie a variety of compromise techniques. Perhaps the most attractive of these are what we will call the *Inheritance* methods—essentially recursive devices.

In an Inheritance method, the effort assigned to a subproblem is determined only by its immediate ancestry; at the time each problem is created it is assigned a certain total quantity Q of time or effort. When a problem is later split into subproblems, such quantities are assigned to them by some local process which *depends only on their relative merits and on what remains of* Q. Thus the centrality problem is managed implicitly. Such schemes are quite easy to program, especially with the new programming systems such as IPL (Newell and Tonge, 1960*c*) and LISP (McCarthy, 1960) (which are themselves based on certain hereditary or recursive operations). Special cases of the Inheritance method arise when one can get along with a simple all-or-none Q, *e.g.*, a "stop condition"—this yields the

[29] A variety of combinatorial methods will be matched against the network-analogy opponent in a program being completed by R. Silver, Lincoln Laboratory, MIT, Lexington, Mass.

exploratory method called "backtracking" by Golumb (1961). The decoding procedure of Wozencraft (1961) is another important variety of Inheritance method.

In the complex exploration process proposed for chess by Newell, Shaw, and Simon (1958*b*) we have a form of Inheritance method with a *nonnumerical stop condition.* Here, the subproblems inherit *sets of goals to be achieved.* This teleological control has to be administered by an additional goal-selection system and is further complicated by a global (but reasonably simple) stop rule of the back-up variety (Sec. IIIC). (*Note:* we are identifying here the move-tree-limitation problem with that of problem selection.) Even though extensive experimental results are not yet available, we feel that the scheme of Newell, Shaw, and Simon (1958*b*) deserves careful study by anyone planning serious work in this area. It shows only the beginning of the complexity sure to come in our development of intelligent machines.[30]

C. "Character-Method" Machines

Once a problem is selected, we must decide which method to try first. This depends on our ability to classify or characterize problems. We first compute the Character of our problem (by using some pattern recognition technique) and then consult a "Character-Method" table or other device which is supposed to tell us which method(s) are most effective on problems of that Character. This information might be built up from experience, given initially by the programmer, deduced from "advice" (McCarthy, 1959), or obtained as the solution to some other problem, as suggested in the GPS proposal (Newell, Shaw and Simon, 1959*a*). In any case, this part of the machine's behavior, regarded from the outside, can be treated as a sort of stimulus-response, or "table look-up," activity.

If the Characters (or descriptions) have too wide a variety of values, there will be a serious problem of filling a Character-Method table. One might then have to reduce the detail of information, *e.g.,* by using only a few important properties. Thus the *Differences* of GPS (see Sec. IVD2) describe no more than is necessary to define a single goal, and a priority scheme selects just one of these to characterize the situation. Gelernter and Rochester (1958) suggest using a property-weighting scheme, a special case of the "Bayes net" described in Sec. IIG.

D. Planning

Ordinarily one can solve a complicated problem only by dividing it into a number of parts, each of which can be attacked by a smaller search (or be further divided). Generally speaking, a successful division will reduce

[30] Some further discussion of this question may be found in Slagle (1961).

the search time not by a mere fraction, but by a *fractional exponent.* In a graph with 10 branches descending from each node, a 20-step search might involve 10^{20} trials, which is out of the question, while the insertion of just four *lemmas* or *sequential subgoals* might reduce the search to only 5×10^4 trials, which is within reason for machine exploration. Thus it will be worth a relatively enormous effort to find such "islands" in the solution of complex problems.[31] Note that even if one encountered, say, 10^6 failures of such procedures before success, one would still have gained a factor of perhaps 10^{10} in over-all trial reduction! *Thus practically any ability at all to "plan," or "analyze," a problem will be profitable,* if the problem is difficult. It is safe to say that all simple, unitary, notions of how to build an intelligent machine will fail, rather sharply, for some modest level of problem difficulty. Only schemes which actively pursue an analysis toward obtaining a set of *sequential goals* can be expected to extend smoothly into increasingly complex problem domains.

Perhaps the most straightforward concept of planning is that of using a *simplified model* of the problem situation. Suppose that there is available, for a given problem, some other problem of "essentially the same character" but with less detail and complexity. Then we could proceed first to solve the simpler problem. Suppose, also, that this is done using a second set of methods, which are also simpler, but in some correspondence with those for the original. *The solution to the simpler problem can then be used as a "plan" for the harder one.* Perhaps each step will have to be expanded in detail. But the multiple searches will *add, not multiply,* in the total search time. The situation would be ideal if the model were, mathematically, a *homomorphism* of the original. But even without such perfection the model solution should be a valuable guide. In mathematics one's proof procedures usually run along these lines: one first assumes, *e.g.,* that integrals and limits always converge, in the planning stage. Once the outline is completed, in this simpleminded model of mathematics, then one goes back to try to "make rigorous" the steps of the proof, *i.e.,* to replace them by chains of argument using genuine rules of inference. And even if the plan fails, it may be possible to patch it by replacing just a few of its steps.

Another aid to planning is the *semantic,* as opposed to the homomorphic, model (Minsky, 1956*a*, 1959*a*). Here we may have an *interpretation* of the current problem within another system, not necessarily simpler, but with which we are more familiar and have already more powerful methods. Thus, in connection with a plan for the proof of a theorem, we will want to know whether the proposed lemmas, or islands in the proof, are actually *true;* if not, the plan will surely fail. We can often easily tell if a proposition is true by looking at an interpretation. Thus the truth of a

[31] See sec. 10 of Ashby (1956).

proposition from plane geometry can be supposed, at least with great reliability, by actual measurement of a few constructed drawings (or the analytic geometry equivalent). The geometry machine of Gelernter and Rochester (1958, 1959) uses such a semantic model with excellent results; it follows closely the lines proposed in Minsky (1956*a*).

1. THE "CHARACTER-ALGEBRA" MODEL

Planning with the aid of a model is of the greatest value in reducing search. Can we construct machines which find their own models? I believe the following will provide a general, straightforward way to construct certain kinds of useful, abstract models. The critical requirement is that we be able to compile a "Character-Method Matrix" (in addition to the simple Character-Method table in Sec. IV*C*). *The CM matrix is an array of entries which predict with some reliability what will happen when methods are applied to problems.* Both of the matrix dimensions are indexed by problem Characters; if there is a method *which usually transforms problems of character* C_i *into problems of character* C_j then let the matrix entry C_{ij} be the name of that method (or a list of such methods). If there is no such method the corresponding entry is null.

Now suppose that there is no entry for C_{ij}—meaning that we have no *direct* way to transform a problem of type C_i into one of type C_j. Multiply the matrix by itself. If the new matrix has a non-null (i,j) entry then there must be a sequence of *two* methods which effects the desired transformation. If that fails, we may try higher powers. Note that [if we put unity for the (i,i) terms] we can reach the 2^n matrix power with just n multiplications. We don't need to define the symbolic multiplication operation; one may instead use arithmetic entries—putting *unity for any non-null entry* and zero for any null entry in the original matrix. This yields a simple connection, or flow diagram, matrix, and its nth power tells us something about its set of paths of length 2^n.[32] [Once a non-null entry is discovered, there exist efficient ways to find the corresponding sequences of methods. The problem is really just that of finding paths through a maze, and the method of Moore (1959) would be quite efficient. Almost any problem can be converted into a problem of finding a chain between two terminal expressions in some formal system.] If the Characters are taken to be abstract representations of the problem expressions, this "Character-Algebra" model can be as abstract as are the available pattern-recognition facilities. See Minsky (1956*a*, 1959*a*).

The critical problem in using the Character-Algebra model for planning is, of course, the *prediction reliability of the matrix entries*. One cannot expect the Character of a result to be strictly determined by the Character of the original and the method used. And the reliability of the pre-

[32] See, *e.g.*, Hohn, Seshu, and Aufenkamp (1957).

dictions will, in any case, deteriorate rapidly as the matrix power is raised. But, as we have noted, any plan at all is so much better than none that the system should do very much better than exhaustive search, even with quite poor prediction quality.

This matrix formulation is obviously only a special case of the character planning idea. More generally, one will have descriptions, rather than fixed characters, and one must then have more general methods to calculate from a description what is likely to happen when a method is applied.

2. CHARACTERS AND DIFFERENCES

In the GPS (General Problem Solver) proposal of Newell, Shaw, and Simon (1959*a*, 1960*a*) we find a slightly different framework: they use a notion of Difference between two problems (or expressions) where we speak of the Character of a single problem. These views are equivalent if we take our problems to be links or connections between expressions. But this notion of Difference (as the Character of a pair) does lend itself more smoothly to teleological reasoning. For what is the goal defined by a problem but to *reduce the "difference" between the present state and the desired state?* The underlying structure of GPS is precisely what we have called a "Character-Method Machine" in which each kind of Difference is associated in a table with one or more methods which are known to "reduce" that Difference. Since the characterization here depends always on (1) the current problem expression and (2) the desired end result, it is reasonable to think, as its authors suggest, of GPS as using "means-end" analysis.

To illustrate the use of Differences, we shall review an example (Newell, Shaw, and Simon, 1960*a*). The problem, in elementary propositional calculus, is to prove that from $S \wedge (- P \supset Q)$ we can deduce $(Q \vee P) \wedge S$. The program looks at both of these expressions with a recursive *matching* process which branches out from the main connectives. The first Difference it encounters is that S occurs on different sides of the main connective "$\wedge$." It therefore looks in the Difference-Method table under the heading "change position." It discovers there a method which uses the theorem $(A \wedge B) \equiv (B \wedge A)$ which is obviously useful for removing, or "reducing," differences of position. GPS applies this method, obtaining $(- P \supset Q) \wedge S$. GPS now asks what is the Difference between this new expression and the goal. This time the matching procedure gets down into the connectives inside the left-hand members and finds a Difference between the connectives "$\supset$" and "$\vee$." It now looks in the CM table under the heading "Change Connective" and discovers the appropriate method using $(- A \supset B) \equiv (A \vee B)$. It applies this method, obtaining $(P \vee Q) \wedge S$. In the final cycle, the difference-evaluating procedure discovers the need for a "change position" inside the left member, and applies a

method using $(A \vee B) \equiv (B \vee A)$. This completes the solution of the problem.[33]

Evidently, the success of this "means-end" analysis in reducing general search will depend on the degree of specificity that can be written into the Difference-Method table—basically the same requirement for an effective Character-Algebra.

It may be possible to *plan* using Differences, as well. One might imagine a "Difference-Algebra" in which the predictions have the form $D = D' D''$. One must construct accordingly a difference-factorization algebra for discovering longer chains $D = D_1 \cdot \cdot \cdot D_n$ and corresponding method plans. We should note that one *cannot* expect to use such planning methods with such primitive Differences as are discussed in Newell, Shaw, and Simon (1960*a*); for these cannot form an adequate Difference-Algebra (or Character-Algebra). Unless the characterizing expressions have many levels of descriptive detail, the matrix powers will too swiftly become degenerate. This degeneracy will ultimately limit the capacity of any formal planning device.

One may think of the general planning heuristic as embodied in a recursive process of the following form. Suppose we have a problem P:

1. Form a plan for problem P.
2. Select first (next) step of the plan.
   (If no more steps, exit with "success.")
3. Try the suggested method(s):
   Success: return to (*b*), *i.e.*, try next step in the plan.
   Failure: return to (*a*), *i.e.*, form new plan, or perhaps change current plan to avoid this step.
   Problem judged too difficult: *Apply this entire procedure to the problem of the current step.*

Observe that such a program schema is essentially recursive; it uses itself as a subroutine (explicitly, in the last step) in such a way that its current state has to be stored, and restored when it returns control to itself.[34]

[33] Compare this with the "matching" process described in Newell and Simon (1956). The notions of "Character," "Character-Algebra," etc., originate in Minsky (1956) but seem useful in describing parts of the "GPS" system of Newell and Simon (1956) and Newell, Shaw, and Simon (1960*a*). The latter contains much additional material we cannot survey here. Essentially, GPS is to be self-applied to the problem of discovering sets of Differences appropriate for given problem areas. This notion of "bootstrapping"—applying a problem-solving system to the task of improving some of its own methods—is old and familiar, but in Newell, Shaw, and Simon (1960*a*) we find perhaps the first specific proposal about how such an advance might be realized.

[34] This violates, for example, the restrictions on "DO loops" in programming systems such as FORTRAN. Convenient techniques for programming such processes were developed by Newell, Shaw and Simon (1960*b*); the program state variables

Miller, Galanter and Pribram[35] discuss possible analogies between human problem-solving and some heuristic planning schemes. It seems certain that, for at least a few years, there will be a close association between theories of human behavior and attempts to increase the intellectual capacities of machines. But, in the long run, we must be prepared to discover profitable lines of heuristic programming which do not deliberately imitate human characteristics.[36]

V. Induction and Models

A. Intelligence

In all of this discussion we have not come to grips with anything we can isolate as "intelligence." We have discussed only heuristics, shortcuts, and classification techniques. Is there something missing? I am confident that sooner or later we will be able to assemble programs of great problem-solving ability from complex combinations of heuristic devices—multiple optimizers, pattern-recognition tricks, planning algebras, recursive administration procedures, and the like. In no one of these will we find the

---

are stored in "pushdown lists" and both the program and the data are stored in the form of "list structures." Gelernter (1959) extended FORTRAN to manage some of this. McCarthy has extended these notions in LISP (1960) to permit *explicit* recursive definitions of programs in a language based on recursive functions of symbolic expressions; here the management of program state variables is fully automatic. See also Orchard-Hays (1960).

[35] See chaps. 12 and 13 of Miller, Galanter, and Pribram (1960).

[36] Limitations of space preclude detailed discussion here of theories of self-organizing neural nets, and other models based on brain analogies. [Several of these are described or cited in *Proceedings of a Symposium on Mechanisation of Thought Processes*, London: H. M. Stationery Office, 1959, and *Self Organizing Systems*, M. T. Yovitts and S. Cameron (eds.), New York: Pergamon Press, 1960.] This omission is not too serious, I feel, in connection with the subject of heuristic programming, because the motivation and methods of the two areas seem so different. Up to the present time, at least, research on neural-net models has been concerned mainly with the attempt to show that certain rather simple heuristic processes, *e.g.*, reinforcement learning, or property-list pattern recognition, can be realized or evolved by collections of simple elements without very highly organized interconnections. Work on heuristic programming is characterized quite differently by the search for new, more powerful heuristics for solving very complex problems, and by very little concern for what hardware (neuronal or otherwise) would minimally suffice for its realization. In short, the work on "nets" is concerned with how far one can get with a small initial endowment; the work on "artificial intelligence" is concerned with using all we know to build the most powerful systems that we can. It is my expectation that, in problem-solving power, the (allegedly brainlike) minimal-structure systems will never threaten to compete with their more deliberately designed contemporaries; nevertheless, their study should prove profitable in the development of component elements and subsystems to be used in the construction of the more systematically conceived machines.

seat of intelligence. Should we ask what intelligence "really is"? My own view is that this is more of an aesthetic question, or one of sense of dignity, than a technical matter! To me "intelligence" seems to denote little more than the complex of performances which we happen to respect, but do not understand. So it is, usually, with the question of "depth" in mathematics. Once the proof of a theorem is really understood its content seems to become trivial. (Still, there may remain a sense of wonder about how the proof was discovered.)

Programmers, too, know that there is never any "heart" in a program. There are high-level routines in each program, but all they do is dictate that "if such and such, then transfer to such and such a subroutine." And when we look at the low-level subroutines, which "actually do the work," we find senseless loops and sequences of trivial operations, merely carrying out the dictates of their superiors. The intelligence in such a system seems to be as intangible as becomes the meaning of a single common word when it is thoughtfully pronounced over and over again.

But we should not let our inability to discern a locus of intelligence lead us to conclude that programmed computers therefore cannot think. For it may be so with *man*, as with *machine*, that, when we understand finally the structure and program, the feeling of mystery (and self-approbation) will weaken.[37] We find similar views concerning "creativity" in Newell, Shaw, and Simon (1958*c*). The view expressed by Rosenbloom (1951) that minds (or brains) can transcend machines is based, apparently, on an erroneous interpretation of the meaning of the "unsolvability theorems" of Godel.[38]

[37] See Minsky (1956, 1959).

[38] On problems of volition we are in general agreement with McCulloch (1954) that our *freedom of will* "presumably means no more than that we can distinguish between what we intend (*i.e.*, our *plan*), and some intervention in our action." See also MacKay (1959) and [the] references; we are, however, unconvinced by his eulogization of "analog" devices. Concerning the "mind-brain" problem, one should consider the arguments of Craik (1952), Hayek (1952), and Pask (1959). Among the active leaders in modern heuristic programming, perhaps only Samuel (1960*b*) has taken a strong position against the idea of machines thinking. His argument, based on the fact that reliable computers do only that which they are instructed to do, has a basic flaw; it does not follow that the programmer therefore has full knowledge (and therefore full responsibility and credit for) what will ensue. For certainly the programmer may set up an evolutionary system whose limitations are for him unclear and possibly incomprehensible. No better does the mathematician know all the consequences of a proposed set of axioms. Surely a machine has to *be* in order to perform. But we cannot assign all the credit to its programmer if the operation of a system comes to reveal structures not recognizable or anticipated by the programmer. While we have not yet seen much in the way of intelligent activity in machines, Samuel's arguments (circular in that they are based on the presumption that machines do not have minds) do not assure us against this. Turing (1956) gives a very knowledgeable discussion of such matters.

B. Inductive Inference

Let us pose now for our machines, a variety of problems more challenging than any ordinary game or mathematical puzzle. Suppose that we want a machine which, when embedded for a time in a complex environment or "universe," will essay to produce a description of that world—to discover its regularities or laws of nature. We might ask it to predict what will happen next. We might ask it to predict what would be the likely consequences of a certain action or experiment. Or we might ask it to formulate the laws governing some class of events. In any case, our task is to equip our machine with *inductive ability*—with methods which it can use to construct general statements about events beyond its recorded experience. Now, there can be no system for inductive inference that will work well in all possible universes. But given a universe, or an ensemble of universes, and a criterion of success, this (epistemological) problem for machines becomes technical rather than philosophical. There is quite a literature concerning this subject, but we shall discuss only one approach which currently seems to us the most promising; this is what we might call the "grammatical induction" schemes of Solomonoff (1957, 1958, 1959*a*), based partly on work of Chomsky and Miller (1957*b*, 1958).

We will take *language* to mean the set of expressions formed from some given set of primitive symbols or expressions, by the repeated application of some given set of rules; the primitive expressions plus the rules is the *grammar* of the language. Most induction problems can be framed as problems in the *discovery of grammars*. Suppose, for instance, that a machine's prior experience is summarized by a large collection of statements, some labelled "good" and some "bad" by some critical device. How could we generate selectively more good statements? The trick is to find some relatively simple (formal) language in which the good statements are grammatical, and in which the bad ones are not. Given such a language, we can use it to generate more statements, and presumably these will tend to be more like the good ones. The heuristic argument is that if we can find a relatively simple way to separate the two sets, the discovered rule is likely to be useful beyond the immediate experience. If the extension fails to be consistent with new data, one might be able to make small changes in the rules and, generally, one may be able to use many ordinary problem-solving methods for this task.

The problem of finding an efficient grammar is much the same as that of finding efficient *encodings*, or programs, for machines; in each case, one needs to discover the important regularities in the data, and exploit the regularities by making shrewd *abbreviations*. The possible importance of Solomonoff's work (1960) is that, despite some obvious defects, it may

point the way toward systematic mathematical ways to explore this discovery problem. He considers the class of all programs (for a given general-purpose computer) which will produce a certain given output (the body of data in question). Most such programs, if allowed to continue, will add to that body of data. By properly weighting these programs, perhaps by length, we can obtain corresponding weights for the different possible continuations, and thus a basis for prediction. If this prediction is to be of any interest, it will be necessary to show some independence of the given computer; it is not yet clear precisely what form such a result will take.

C. Models of Oneself

If a creature can answer a question about a hypothetical experiment, without actually performing that experiment, then the answer must have been obtained from some submachine inside the creature. The output of that submachine (representing a correct answer) as well as the input (representing the question) must be coded descriptions of the corresponding external events or event classes. Seen through this pair of encoding and decoding channels, the internal submachine acts like the environment, and so it has the character of a "model." The inductive inference problem may then be regarded as the problem of constructing such a model.

To the extent that the creature's actions affect the environment, this internal model of the world will need to include some representation of the creature itself. If one asks the creature "why did you decide to do such and such" (or if it asks this of itself), any answer must come from the internal model. Thus the evidence of introspection itself is liable to be based ultimately on the processes used in constructing one's image of one's self. Speculation on the form of such a model leads to the amusing prediction that intelligent machines may be reluctant to believe that they are *just* machines. The argument is this: our own self-models have a substantially "dual" character; there is a part concerned with the physical or mechanical environment—with the behavior of inanimate objects—and there is a part concerned with social and psychological matters. It is precisely because we have not yet developed a satisfactory mechanical theory of mental activity that we have to keep these areas apart. We could not give up this division even if we wished to—until we find a unified model to replace it. Now, when we ask such a creature what sort of being it is, it cannot simply answer "directly"; it must inspect its model(s). And it must answer by saying that it seems to be a dual thing—which appears to have two parts—a "mind" and a "body." Thus, even the robot, unless equipped with a satisfactory theory of artificial intelligence, would have to maintain a dualistic opinion on this matter.[39]

[39] There is a certain problem of infinite regression in the notion of a machine

Conclusion

In attempting to combine a survey of work on "artificial intelligence" with a summary of our own views, we could not mention every relevant project and publication. Some important omissions are in the area of "brain models"; the early work of Farley and Clark (1954) [also Farley's paper in Yovitts and Cameron (1960), often unknowingly duplicated, and the work of Rochester (1956) and Milner (1960)]. The work of Lettvin *et al.* (1959) is related to the theories in Selfridge (1959). We did not touch at all on the problems of logic and language, and of information retrieval, which must be faced when action is to be based on the contents of large memories; see, *e.g.*, McCarthy (1959). We have not discussed the basic results in mathematical logic which bear on the question of what can be done by machines. There are entire literatures we have hardly even sampled—the bold pioneering work of Rashevsky (c. 1929) and his later co-workers (Rashevsky, 1960); Theories of Learning, *e.g.*, Gorn (1959); Theory of Games, *e.g.*, Shubik (1960); and Psychology, *e.g.*, Bruner *et al.* (1956). And everyone should know the work of Polya (1945, 1954) on how to solve problems. We can hope only to have transmitted the flavor of some of the more ambitious projects *directly* concerned with getting machines to take over a larger portion of problem-solving tasks.

One last remark: we have discussed here only work concerned with more or less self-contained problem-solving programs. But as this is written, we are at last beginning to see vigorous activity in the direction of constructing usable *time-sharing* or *multiprogramming* computing systems. With these systems, it will at last become economical to match human beings in real time with really large machines. This means that we can work toward programming what will be, in effect, "thinking aids." In the years to come, we expect that these man-machine systems will share, and perhaps for a time be dominant, in our advance toward the development of "artificial intelligence."

---

having a *good* model of itself: of course, the nested models must lose detail and finally vanish. But the argument, *e.g.*, of Hayek (see 8.69 and 8.79, 1952) that we cannot "fully comprehend the unitary order" (of our own minds) ignores the power of recursive description as well as Turing's demonstration that (with sufficient external writing space) a "general-purpose" machine can answer any question about a description of itself that any larger machine could answer.

Appendix C

Allen Newell. "Heuristic Programming: Ill-Structured Problems," chapter 10, pp 361-414, in Publications in Operations Research, *Progress in Operations Research: Relationship Between Operations Research and the Computer*, Volume III, edited by Julius S. Aronofsky, of Operations Research Society of America, Publications in Operations Research Number 16, David B. Hertz, editor, copyrighted 1969. Reprinted by permission of John Wiley & Sons, Inc., New York.

Appendix C

Chapter 10

HEURISTIC PROGRAMMING: ILL-STRUCTURED PROBLEMS

ALLEN NEWELL

Carnegie-Mellon University,
Pittsburgh, Pennsylvania

Reprinted from J. S. Aronofsky, (ed.)
Progress in Operations Research, Volume III,
John Wiley & Sons, 1969.

Contents

We observe that on occasion expressions in some language are put forward that purport to state "a problem." In response a method (or algorithm) is advanced that claims to solve the problem. That is, if input data are given that meet all the specifications of the problem statement, the method produces another expression in the language that is the solution to the problem. If there is a challenge as to whether the method actually provides a general solution to the problem (i.e., for all admissible inputs), a proof may be forthcoming that it does. If there is a challenge to whether the problem statement is well defined, additional formalization of the problem statement may occur. In the extreme this can reach back to formalization of the language used to state the problem, until a formal logical calculus is used.

We also observe that for other problems that people have and solve there seems to be no such formalized statement and formalized method. Although usually definite in some respects problems of this type seem incurably fuzzy in others. That there should be ill-defined problems around is not very surprising. That is, that there should be expressions that have some characteristics of problem statements but are fragmentary seems not surprising. However, that there should exist systems (i.e., men) that can solve these problems without the eventual intervention of formal statements and formal methods does pose an issue. Perhaps there are two domains of problems, the well structured and the ill structured. Formalization always implies the first. Men can deal with both kinds. By virtue of their capacity for working with ill-structured problems, they can transmute some of these into well-structured (or formalized) problems. But the study of formalized problems has nothing to say about the domain of ill-structured problems. In particular, there can never be a formalization of ill-structured problems, hence never a theory (in a strict sense) about them. All that is possible is the conversion of particular problems from ill structured to well structured via the one transducer that exists, namely, man.

Perhaps an analog is useful: well-structured problems are to ill-structured problems as linear systems are to nonlinear systems, or as

I wish to acknowledge the help of J. Moore in clarifying the nature of the methods of artificial intelligence and also my discussions with my colleague H. A. Simon. The research was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-146).

stable systems are to unstable systems, or as rational behavior is to non-rational behavior. In each case, it is not that the world has been neatly divided into two parts, each with a theory proper to it. Rather, one member of the pair is a very special case about which much can be said, whereas the other member consists of all the rest of the world—uncharted, lacking uniform approach, inchoate, and incoherent.

This is not the only view, of course. Alternatively, one can assert that all problems can be formulated in the same way. The formalization exists because there is some symbolic system, whether on paper or in the head, that holds the specific, quite definite information in the problem statement. The fragmentation of problem statement that occurs when an attempt is made to explicate the problem only shows that there are serious (perhaps even profound) communication problems. But it does not say that ill-structured problems are somehow different in nature.

Thus we have an issue—somewhat ill structured, to be sure—but still an issue. What are ill-structured problems and are they a breed apart from well-structured ones? This chapter is essentially an essay devoted to exploring the issue, as it stands in 1968.

We have an issue defined. What gives life to it are two concerns, one broad, one narrow. At root, there is the long-standing concern over the rationalization of human life and action. More sharply stated, this is the challenge of art by science. In terms of encounters long resolved, it is whether photography will displace painting, or whether biology and physiology can contribute to the practice of medicine. In terms of encounters now in twilight, it is whether new products come from applied science or from lone inventors. In terms of encounters still active, it is whether the holistic diagnostic judgment of the clinical psychologist is better than the judgments of a regression equation [12]. For the purpose of this essay, of course, it is to what extent management science can extend into the domain of business judgment.

When put in this context, the issue has a charge. The concern flows partly from economics, where it is now usually labeled the problem of automation. Concern also flows from a problem of identity, in which some are compelled to ask what attributes man can uniquely call his own. As has been pointed out, probably most thoroughly by Ellul [3], it makes no sense to separate hardware and software, that is, to separate machines from procedures, programs, and formalized rules. They are all expressions of the rationalization of life, in which human beings become simply the agents or carriers of a universalistic system of orderly relations of means to ends.

Thus, viewed broadly, the issue is emotionally toned. However, this

fact neither eliminates nor affects the scientific questions involved, although it provides reasons for attending to them. Our aim in this essay is essentially scientific, a fact which leads to the second, more narrow context.

Within management science the nature of rationalization has varied somewhat over time. In the early days, those of Frederick Taylor, it was expressed in explicit work procedures, but since World War II it has been expressed in the development of mathematical models and quantitative methods. In 1958 we put it as follows:

> A problem is well structured to the extent that it satisfies the following criteria:
>
> 1. It can be described in terms of numerical variables, scalar and vector quantities.
> 2. The goals to be attained can be specified in terms of a well-defined objective function—for example, the maximization of profit or the minimization of cost.
> 3. There exist computational routines (*algorithms*) that permit the solution to be found and stated in actual numerical terms. Common examples of such algorithms, which have played an important role in operations research, are maximization procedures in the calculus and calculus of variations, linear-programming algorithms like the stepping-stone and simplex methods, Monte Carlo techniques, and so on [21, pp. 4–5].

Ill-structured problems were defined, as in the introduction of this essay, in the negative: all problems that are not well structured. Now the point of the 1958 paper, and the reason that it contrasted well- and ill-structured problems, was to introduce heuristic programming as relevant to the issue:

> With recent developments in our understanding of heuristic processes and their simulation by digital computers, the way is open to deal scientifically with ill-structured problems—to make the computer co-extensive with the human mind [21, p. 9].

That is, before the development of heuristic programming (more generally, artificial intelligence) the domain of ill-structured problems had been the exclusive preserve of human problem solvers. Now we had other systems that also could work on ill-structured problems and that could be studied and used.

This 1958 paper is a convenient marker for the narrow concern of the present essay. It can symbolize the radical transformation brought by the computer to the larger, almost philosophical concern over the nature and possibilities for rationalization. The issue has become almost technical, although now it involves three terms, where before it involved only two:

- What is the nature of problems solved by formal algorithms?
- What is the nature of problems solved by computers?
- What is the nature of problems solved by men?

We have called the first well-structured problems; the last remains the residual keeper of ill-structured problems; and the middle term offers the opportunity for clarification.

Our course will be to review the 1958 paper a little more carefully, leading to a discussion of the nature of problem solving. From this will emerge an hypothesis about the nature of generality in problem solving, which will generate a corresponding hypothesis about the nature of ill-structured problems. With theses in hand, we first consider some implications of the hypotheses, proceed to explore these a little, and finally bring out some deficiencies.

The 1958 paper asserted that computers (more precisely, computers appropriately programmed) could deal with ill-structured problems, where the latter was defined negatively. The basis of this assertion was twofold. First, there had just come into existence the first successful heuristic programs, that is to say, programs that performed tasks requiring intelligence when performed by human beings. They included a theorem-proving program in logic [15], a checker-playing program [19], and a pattern recognition program [20]. These were tasks for which algorithms either did not exist or were so immensely expensive as to preclude their use. Thus, there existed some instances of programs successfully solving interesting ill-structured problems. The second basis was the connection between these programs and the nature of human problem solving [16]. Insofar as these programs reflected the same problem-solving processes as human beings used, there was additional reason to believe that the programs dealt with ill-structured problems. The data base for the assertion was fairly small, but there followed in the next few years additional heuristic programs that provided support. There was one that proved theorems in plane geometry, one that did symbolic indefinite integration, a couple of chess programs, a program for balancing assembly lines, and several pattern recognition programs [5].

The 1958 paper provided no positive characterization of ill-structured problems. Although it could be said that some ill-structured problems were being handled, these might constitute a small and particularly "well-formed" subset. This was essentially the position taken by Reitman, in one of the few existing direct contributions to the question of ill-formed problems [17, 18]. He observed, as have others, that all of the heuristic programs, although lacking well-specified algorithms, were otherwise quite

precisely defined. In particular, the test whereby one determined whether the problem was solved was well specified, as was the initial data base from which the problem started. Thus, he asserted that all existing heuristic programs were in a special class by virtue of certain aspects being well defined, and thus shed little light on the more general case.

Stating this another way, it is not enough for a problem to become ill structured only with respect to the methods of solution. It is required also to become ill structured with respect to both the initial data and the criteria for solution. To the complaint that one would not then really know what the problem was, the rejoinder is that almost all problems dealt with by human beings are ill structured in these respects. To use an example discussed by Reitman, in the problem of making a silk purse from a sow's ear, neither "silk purse" nor "sow's ear" is defined beyond cavil. To attempt really to solve such a problem, for instance, would be to search for some ways to stretch the implicit definitions to force acceptance of the criteria, for example, chemical decomposition and resynthesis.

Reitman attempted a positive characterization of problems by setting out the possible forms of uncertainty in the specification of a problem: the ways in which the givens, the sought-for transformation, or the goal could be ill defined. This course has the virtue, if successful, of defining "ill structured" independently of problem solving and thus providing a firm base on which to consider how such problems might be tackled. I will not follow him in this approach, however. It seems more fruitful here to start with the activity of problem solving.

1. THE NATURE OF PROBLEM SOLVING

A rather general diagram, shown in Fig. 10.1, will serve to convey a view of problem solving that captures a good deal of what is known, both casually and scientifically. A problem solver exists in a task environment, some small part of which is the immediate stimulus for evoking the problem and which thus serves as the initial problem statement.[1] This external representation is translated into some internal representation (a condition, if you please, for assimilation and acceptance of the problem by the problem solver). There is located within the memory of the problem solver a collection of methods. A method is some organ-

[1] Its statement form is clear when given linguistically, as in "Where do we locate the new warehouse?" Otherwise, "statement" is to be taken metaphorically as comprising those clues in the environment attended to by the problem solver that indicate to him the existence of the problem.

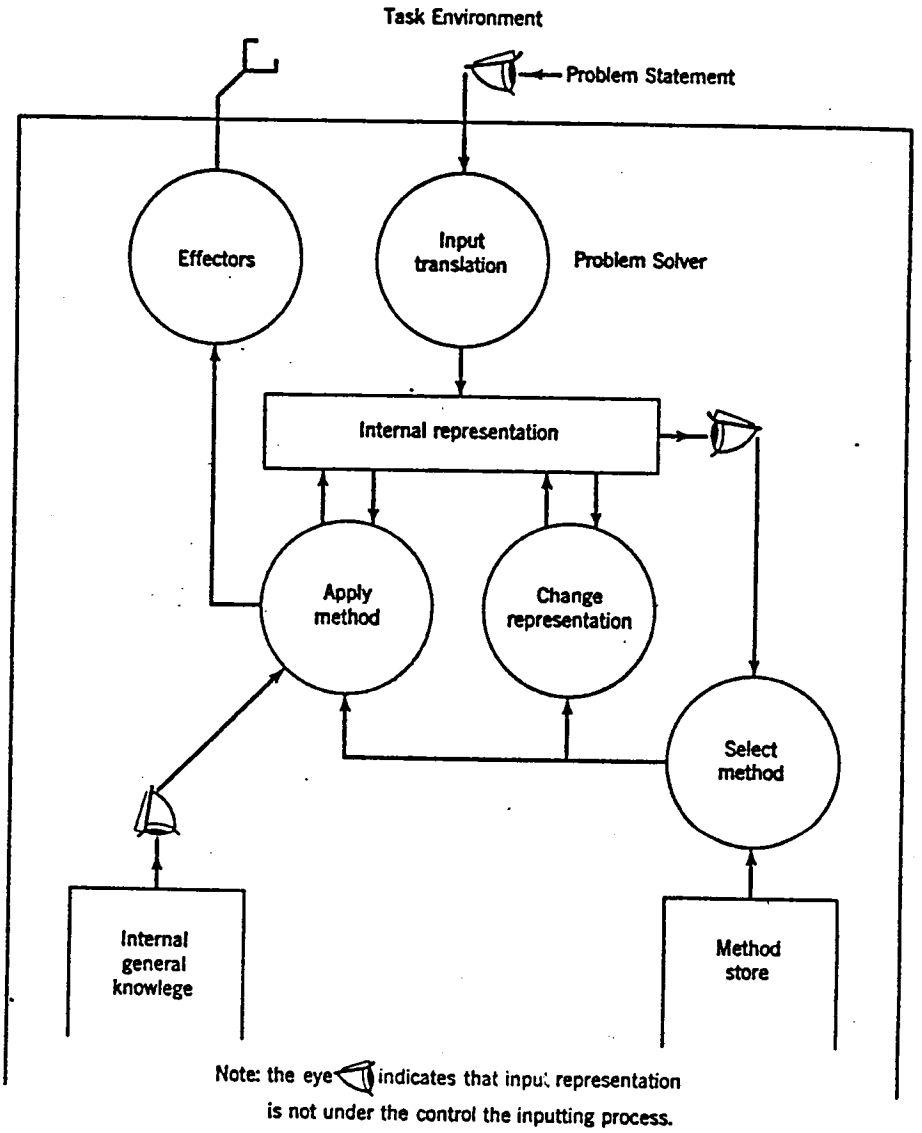


Figure 10.1. General schema of a problem solver.

ized program or plan for behavior that manipulates the internal representation in an attempt to solve the problem. For the type of problem solvers we have in mind—business men, analysts, etc.—there exist many relatively independent methods, so that the total behavior of the problem

solver is made up as an iterative cycle in which methods are selected on the basis of current information (in the internal representation) and tried with consequent modification of the internal representation, and a new method is selected.

Let us stay with this view of problem solving for a while, even though it de-emphasizes some important aspects, such as the initial determination of an internal representation, its possible change, and the search for or construction of new methods (by other methods) in the course of problem solving. What Fig. 10.1 does emphasize is the method—the discrete package of information that guides behavior in an attempt at problem solution. It prompts us to inquire about its anatomy.

1.1. The Anatomy of a Method

Let us examine some method in management science. The simplex method of linear programming will serve admirably. It is well known, important, and undoubtedly a method. It also works on well-structured problems, but we will take due account of this later. The basic linear programming problem is as follows.

Given: a set of positive real variables

$$x_j \geqslant 0, \qquad j = 1, \ldots, n$$

and real constants

$$a_{ij}, b_i, c_j, \qquad i = i, \ldots, m; j = 1, \ldots, n$$

maximize

$$z = \sum_j c_j x_j$$

subject to

$$\sum_j a_{ij} x_j \leqslant b_i, \qquad i = 1, \ldots, m$$

Figure 10.2 shows the standard data organization used in the simplex method and gives the procedure to be followed. We have left out the procedure for getting an initial solution, that is, we assume that the tableau of Fig. 10.2 is initially filled in. Likewise, we have ignored the detection of degeneracy and the procedure for handling it.

There are three parts to the simplex method. The first is the *statement of the problem.* This determines the entities you must be able to identify in a situation and what you must know about their properties and mutual interrelationships. Unless you can find a set of nonnegative numerical quantities, subject to linear constraints and having a linear objective function to be maximized, you do not have a LP problem, and

the method cannot help you. The second part of the method is the *procedure* that delivers the solution to the problem. It makes use only of information available in the problem statement. Indeed, these two are coupled together as night and day. Any information in the problem statement which is known not to enter into the procedure can be discarded, thereby making the problem just that much more general.

Simplex tableau

Basis	b_i	c_1 x_1	c_2 x_2	...	c_n x_n	0 x_{n+1}	...	0 x_{n+m}
x_{j_1}								
x_{j_2}								
.				t_{ij}				
.								
x_{j_m}								
z_j								
$z_j - c_j$								

Procedure

1. Let j_0 = column with min $\{z_j - c_j | z_j - c_j < 0\}$;
   if no $z_j - c_j < 0$, then at maximum z, end.
2. Let i_0 = row with min $\{b_i/t_{ij_0} | b_i/t_{ij_0} > 0\}$;
   if no $b_i/t_{ij_0} > 0$, then z unbounded, end.
3. For row i_0, $t_{i_0 j} \leftarrow t_{i_0 j}/t_{i_0 j_0}$.
4. For row $i \neq i_0$, $t_{ij} \leftarrow t_{ij} - t_{i_0 j} t_{ij_0}$
   ($t_{i_0 j}$ is the value from step 3).

Define

$$x_{n+i} = b_i - \sum_j a_{ij} x_j, \qquad i = 1, \ldots, m$$

$$c_{n+i} = 0$$

$$a_{i,n+k} = 1 \text{ if } i = k, \text{ 0 otherwise}$$

Figure 10.2. Simplex method.

The third part of the method is the proof or *justification* that the procedure in fact delivers the solution to the problem (or delivers it within certain specified limits). The existence of this justification has several consequences. One, already noted, is the complete adaptation of means to ends—of the shaping of the problem statement so that it is as general as possible with respect to the procedure. Another consequence is a toleration of apparent meaninglessness in the procedure. It makes no difference that there seems to be neither rhyme nor reason to the steps of the method in Fig. 10.2. Careful analysis reveals that they are in fact just those steps necessary to the attainment of the solution. This feature is characteristic of mathematical and computational methods generally and sometimes is even viewed as a hallmark.

An additional part of the simplex method is a *rationale* that can be used to make the method understandable. The one usually used for the simplex is geometrical, with each constraint being a (potential) boundary plane of the space of feasible solutions. Then the simplex procedure is akin to climbing from vertex to vertex until the maximal one is reached. This fourth part is less essential than the other three.

The first three parts seem to be characteristic of all methods. Certainly, examples can be multiplied endlessly. The quadratic formula provides another clear one:

Problem statement:	Find x such that $ax^2 + bx + c = 0$.
Procedure:	compute $x = b/2a \pm \frac{1}{2}a\sqrt{b^2 - 4ac}$.
Justification:	(substitute formula in $ax^2 + bx + c$ and show by algebraic manipulation that 0 results).

In each case a justification is required (and forthcoming) that establishes the relation of method to problem statement. As we move toward more empirical methods, the precision of both the problem statement and the procedure declines, and concurrently the precision of the justification; in fact, justification and plausible rationale merge into one.

1.2. Generality and Power

We need to distinguish the generality of a method from its power. A method lays claim via its problem statement to being applicable to a certain set of problems, namely, to all those for which the problem statement applies. The generality of a method is determined by how large the set of problems is. Even without a well-defined domain of all problems, or any suitable measure on the sets of problems, it is still often possible to compare two problem statements and judge one to be more inclusive than another, hence one method more general than the other.

A method that is applicable only to locating warehouses is less general than one that is applicable to problems involving the location of all physical resources. But nothing interesting can be said about the relative generality of a specific method for inventory decisions versus one for production scheduling.

Within the claimed domain of a method we can inquire after its ability to deliver solutions: the higher this is, the more powerful the method. At least three somewhat independent dimensions exist along which this ability can be measured. First, the method may or may not solve every problem in the domain; and we may loosely summarize this by talking of the probability of solution. Second, there may exist a dimension of quality in the solution, such as how close an optimizing method gets to the peak. Then methods can differ on the quality of their solutions. (To obtain a simple characteristic for this requires some summarization over the applicable domain, but this feature need not concern us here.) Third, the method may be able to use varying amounts of resources. Then, judgments of probability of solution and of quality are relative to the amount of resources used. Usually the resource will be time, but it can also be amount of computation, amount of memory space, number of dollars to acquire information, and so on. For example, most iterative methods for solving systems of equations do not terminate in a finite number of iterations, but produce better solutions if run longer; the rate of convergence becomes a significant aspect of the power of such methods.

In these terms the simplex method would seem to rank as one of limited generality but high power. The restrictions to linearity, both in the constraints and the objective function, and to a situation describable by a set of real numbers, all constrict generality. But the simplex method is an algorithm within its domain and guarantees delivery of the complete solution. It is not the least general method, as is indicated by the transportation problem with its more specialized assumptions; nor is it the most powerful method for its domain, since it can be augmented with additional schemes that obtain solutions more expeditiously.

Evidently there is an inverse relationship between the generality of a method and its power. Each added condition in the problem statement is one more item that can be exploited in finding the solution, hence in increasing the power. If one takes a method, such as the simplex method, and generalizes the problem statement, the procedure no longer solves every problem in the wider domain, but only a subset of these. Thus the power diminishes. The relationship is not one-one, but more like a limiting relationship in which the amount of information in the problem statement puts bounds on how powerful the method can be. This re-

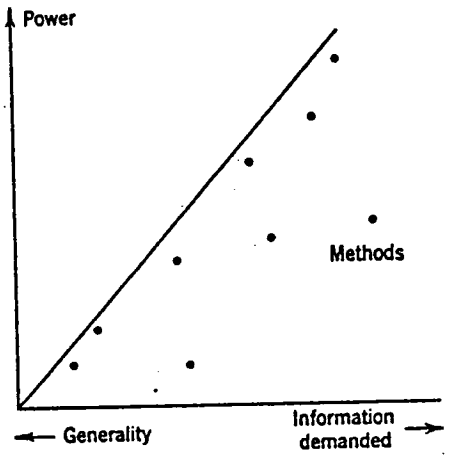


Figure 10.3. Generality versus power.

lationship is important enough to the argument of this essay that we indicate it symbolically in Fig. 10.3. The abcissa represents increasing information in the problem statement, that is, decreasing generality. The ordinate represents increasing power. For each degree of generality an upper bound exists on the possible power of a method, though there are clearly numerous methods which do not fully exploit the information in the problem statement.

2. TWO HYPOTHESES: ON GENERALITY AND ON ILL-STRUCTURED PROBLEMS

With this view of method and problem solver we can move back toward the nature of ill-structured problems. However, we need to address one intermediate issue: the nature of a general problem solver. The first heuristic programs that were built laid claims to power, not to generality. A chess or checker program was an example of artificial intelligence because it solved a problem difficult by human standards; there was never a pretense of its being general. Today's chess programs cannot even play checkers, and vice versa.

Now this narrowness is completely consistent with our general experience with computer programs as highly special methods for restricted tasks. Consider a typical subroutine library, with its specific routines for inverting matrices, computing the sine, carrying out the simplex method, and so on. The only general "programs" are the higher-level

programming languages, and these are not problem solvers in the usual sense, but only provide means to express particular methods.[2] Thus the view has arisen that, although it may be possible to construct an artificial intelligence for any highly specific task domain, it will not prove possible to provide a general intelligence. In other words, it is the ability to be a general problem solver that marks the dividing line between human intelligence and machine intelligence.

The formulation of method and problem solver given earlier leads rather directly to a simple hypothesis about this question:

Generality Hypothesis. A general problem solver is one that has a collection of successively weaker methods that demand successively less of the environment in order to be applied. Thus a good general problem solver is simply one that has the best of the weak methods.

This hypothesis, although itself general, is not without content. (To put it the way that philosophers of science prefer, it is falsifiable.) It says that there are no special mechanisms of generality—nothing beyond the willingness to carry around specific methods that make very weak demands on the environment for information. By the relationship expressed in Fig. 10.3 magic is unlikely, so that these methods of weak demands will also be methods of low power. Having a few of them down at the very low tip in the figure gives the problem solver the ability to tackle almost any kind of problem, even if only with marginal success.

There are some ways in which this generality hypothesis is almost surely incorrect or at least incomplete, and we will come to these later; but let us remain with the main argument. There is at least one close association between generality and ill-structured problems: it is man that can cope with both. It is also true that ill-structured problems, whatever else may be the case, do not lend themselves to sharp solutions. Indeed, their lack of specificity would seem to be instrumental in prohibiting the use of precisely defined methods. Since every problem does present some array of available information—something that could meet the conditions of a problem statement of some method—the suspicion arises that lack of structure in a problem is simply another indication that there are not methods of high power for the particular array of information available. Clearly this situation does not prevail absolutely, but only with respect to a given problem solver and his collection of methods (or, equally, a population of highly similar problem solvers). We can phrase this suspicion in sharper form:

[2] The relationship of programming languages to problem solvers, especially as the languages become more problem-oriented, is unexplored territory. Although relevant to the main question of this essay, it cannot be investigated further here.

Ill-structured Problem Hypothesis. A problem solver finds a problem ill structured if the power of his methods that are applicable to the problem lies below a certain threshold.

The lack of any uniform measure of power, with the consequent lack of precision about a threshold on this power, is not of real concern: the notion of ill-structuredness is similarly vague. The hypothesis says that the problem of locating a new warehouse will look well structured to a firm that has, either by experience, analysis, or purchase, acquired a programmed procedure for locating warehouses, providing it has decided that the probability of obtaining an answer of suitable quality is high enough simply to evoke the program in the face of the new location problem. The problem will look ill structured to a firm that has only its general problem-solving abilities to fall back on. It can only have the most general faith that these procedures will discover appropriate information and use it in appropriate ways in making the decision.

My intent is not to argue either of these two hypotheses directly, but rather to examine some of their implications. First, the weak methods must be describable in more concrete terms. This we will do in some detail, since it has been the gradual evolution of such methods in artificial intelligence that suggested the hypotheses in the first place. Second, the picture of Fig. 10.3 suggests not only that there are weak methods and strong ones, but that there is continuity between them in some sense. Phrased another way, at some level the methods of artificial intelligence and those of operations research should look like members of the same family. We will also look at this implication, although somewhat more sketchily, since little work has been done in this direction. Third, we can revisit human decision makers in ill-structured situations. This we do in an even more sketchy manner, since the main thrust of this essay stems from the more formal concerns. Finally, after these (essentially positive) explications of the hypotheses, we will turn to discussion of some difficulties.

3. THE METHODS OF HEURISTIC PROGRAMMING

There has been continuous work in artificial intelligence ever since the article quoted at the beginning of this chapter [21] took note of the initial efforts. The field has had two main branches. We will concentrate on the one called heuristic programming. It is most closely identified with the programmed digital computer and with problem solving. Also, almost all the artificial intelligence efforts that touch management science are included within it. The other branch, identified with pattern

recognition, self-organizing systems, and learning systems, although not exempt from the observations to be made here, is sufficiently different to preclude its treatment.

A rather substantial number of heuristic programs have been constructed or designed and have gone far enough to get into the literature. They cover a wide range of tasks: game playing, mostly chess, checkers, and bridge; theorem proving, mostly logic, synthetic geometry, and various elementary algebraic systems; all kinds of puzzles; a range of management science tasks, including line balancing, production scheduling, and warehouse location; question-answering systems that accept quasi-English of various degrees of sophistication; and induction problems of the kind that appear on intelligence tests. The main line of progress has constituted a meandering tour through new task areas which seemed to demand new analyses. For example, there is considerable current work on coordinated effector-receptor activity (e.g., hand-eye) in the real world—a domain of problems requiring intelligence that has not been touched until this time.

Examination of this collection of programs reveals that only a few ideas seem to be involved, despite the diversity of tasks. These ideas, if properly expressed, can become a collection of methods in the sense used earlier. Examination of these methods shows them to be extraordinarily weak compared with the methods, say, of linear programming. In compensation, they have a generality that lets them be applied to tasks such as discovering proofs of theorems, where strong methods are unknown.[3]

It thus appears that the work in heuristic programming may provide a first formalization of the kind of weak methods called for by our two hypotheses. (To be sure, as already noted, psychological invention runs the other way: the discovery that there seems to be a small set of methods underlying the diversity of heuristic programs suggested the two hypotheses.)

It might be claimed that the small set of methods shows, not parsimony, but the primitive state of development of the field and that investigators read each other's papers. Although there is clearly some force to this argument, to an important degree each new task attempted in heuristic programming represents an encounter with an unknown set of

[3] Strong methods of proof discovery do get developed in the course of mathematical progress, but their effect is to reduce whole areas to a calculus. The development of the operational calculus—later the Laplace and Fourier transforms—is a case in point. But the present theorem-proving programs and the methods that lie behind them do not involve mathematical advances; rather they appear to capture methods available for proof discovery within the existing state of ignorance.

demands that have to be met on their own terms. Certainly the people who have created heuristic programs have often felt this way. In fact, the complaint is more often the opposite to the above caveat—that artificial intelligence is a field full of isolated cases with no underlying coherency.

In fact, the view expressed in this chapter is not widely held. There is some agreement that all heuristic theorem provers and game players make use of a single scheme, called heuristic search. But there is little acknowledgment that the remainder of the methods listed below constitute some kind of basic set.

With this prelude, let us describe briefly some methods. An adequate job cannot be done in a single chapter; it is more an undertaking for a textbook. Hopefully, however, some feeling for the essential characteristics of generality and power can be obtained from what is given. The first three, generate-and-test, match, and hill climbing, rarely occur as complete methods in themselves (although they can), but are rather the building blocks out of which more complex methods are composed.

3.1. Generate-and-Test

This is the weak method par excellence. All that must be given is a way to generate possible candidates for solution plus a way to test whether they are indeed solutions. Figure 10.4 provides a picture of generate-and-test that permits us to view it as a method with a problem statement and a procedure. The flow diagram in the figure adopts some conventions that will be used throughout. They allow expresson of the central idea of a method without unnecessary detail. The lines in the diagram show the flow of data, rather than the flow of control—more in the style of an analog computer flow diagram than a digital computer flow diagram. Thus the nodes represent processes that receive inputs and deliver outputs. If a node is an item of data, as in the predicate P or the set $\{x\}$ (braces are used to indicate sets), it is a memory process that makes the data item available. A process executes (or fires) when it receives an input; if there are several inputs, it waits until all appropriate ones have arrived before firing.

A generator is a process that takes information specifying a set and produces elements of that set one by one. It should be viewed as autonomously "pushing" elements through the system. Hence there is a flow of elements from generate to the process called test. Test is a process that determines whether some condition or predicate is true of its input and behaves differentially as a result. Two different outputs are possible: satisfied (+) and unsatisfied (−). The exact output behavior depends

Problem statement

Given: a generator of the set $\{x\}$;
a test of the predicate P defined on elements of $\{x\}$;
Find: an element of $\{x\}$ that satisfies $P(x)$.

Procedure

$$\{x\} \dashrightarrow \text{generate} \xrightarrow{\text{element input}} \overset{P}{\underset{\vdots}{\text{test}}} \xrightarrow{+ \text{ solution}}$$

Justification

To show y is a solution if and only if $y \in \{x\}$ and $P(y)$.
Notation: $\pi \rightarrow \alpha$ means that process π produces α;
α/β means that α is on line labeled β.
Test has associated with it a predicate P on one variable, such that:

$\text{test} \rightarrow \alpha/+$ if and only if $P(\alpha)$ and α/input;
$\text{test} \rightarrow \alpha/-$ if and only if $\neg P(\alpha)$ and α/input.

Generate has associated with it a set $\{x\}$ such that:
generate $\rightarrow \alpha/\text{element}$ only if $\alpha \in \{x\}$;
$\alpha \in \{x\}$ implies there exists a time when generate $\rightarrow \alpha/\text{element}$.
Working backward from the flow line labeled solution, we get:

1. $y/\text{solution}$ if and only if test $\rightarrow y/+$.
2. test $\rightarrow y/+$ if and only if $P(y)$ and y/input.

Now we need only show that y/input if and only if $y \in \{x\}$.

3. y/input if and only if generate $\rightarrow y/\text{element}$.
4. generate $\rightarrow y/\text{element}$ only if $y \in \{x\}$.

Now we need only show that $y \in \{x\}$ implies generate $\rightarrow y/\text{element}$; however, the best we can do is:

5. $y \in \{x\}$ implies there exists a time when generate $\dashrightarrow y/\text{element}$.

Figure 10.4. Generate-and-test method.

on the needs of the rest of the processing. The input can be passed through on one condition and nothing done on the other, in which case test acts as a filter or gate. The input can be passed through in both cases, but on different output lines, in which case test acts as a binary switch.

The set associated with a generator and the predicate associated with a test are not inputs. Rather, they are constructs in terms of which the behavior of the process can be described. This is done in Fig. 10.4 by listing a set of propositions for each process. The single arrow (→) indi-

cates production of an output, and the slash (/) indicates that a data item is on the line with the given label. Thus the first proposition under test says that test produces an item on its + output line if and only if that item was input and also satisfies the associated predicate. For any particular generator the associated set must be fully specified, but clearly that specification can be shared in particular ways between the structure of the generator and some actual inputs; for example, a generator could take an integer as input and produce the integers greater than the input, or it could have no input at all and simply generate the positive integers. The same situation holds for test or any other process: its associated constructs must be fully specified, but that specification can be shared in different ways between the structure of the process and some of its inputs. Sometimes we will put the associated construct on the flow diagram, as we have in Fig. 10.4, to show the connection between the processes in the flow diagram and the constructs used in the statement of the problem. We use dotted lines to show that these are not really inputs, although inputs could exist that partially specify them.

We have provided a sketch of a justification that the procedure of the method actually solves the problem. In substance the proof is trivial. To carry it through in detail requires formalization of both the procedure and the language for giving the problem statements and the properties known to hold if a process is executed [6]. The handling of time is a bit fussy and requires more formal apparatus than is worthwhile to present here. Note, for instance, that if the generator were not allowed to go to conclusion, generate-and-test would not necessarily produce a solution. Similar issues arise with infinite sets. Justifications will not be presented for the other methods. The purpose of doing it for this (simplest) one is to show that all the components of a method—problem statement, procedure, and justification—exist for these methods of artificial intelligence. However, no separate rationale is needed for generate-and-test, partly because of its simplicity and partly because of the use of a highly descriptive procedural language. If we had used a machine code, for instance, we might have drawn the procedure of Fig. 10.4 as an informal picture of what was going on.

Generate-and-test is used as a complete method, for instance, in opening a combination lock (when in desperation). Its low power is demonstrated by the assertion that a file with a combination lock is a "safe." Still, the method will serve to open the safe eventually. Generate-and-test is often used by human beings as a second method for finding lost items, such as a sock or a tiepin. The first method relies on recollections about where the item was left or last seen. After this has failed,

generate-and-test is evoked, generating the physical locations in the room one by one, and looking in each.

The poor record of generate-and-test as a complete method should not blind one to its ubiquitous use when other information is absent. It is used to scan the want ads for neighborhood rentals after the proper column is discovered (to the retort "What else?", the answer is, "Right! That's why the method is so general"). In problem-solving programs it is used to go down lists of theorems or of subproblems. It serves to detect squares of interest on chessboards, words of interest in expressions, and figures of interest in geometrical displays.

3.2. Match

We are given the following expression in symbolic logic:

$$e\text{: } (p \vee q) \supset ((p \vee q) \vee (r \supset p))$$

A variety of problems arise from asking whether e is a member of various specified sets of logic expressions. Such problems can usually be thrown into the form of a generate-and-test, at which point the difficulty of finding the solution is directly proportional to the size of the set.

If we know more about the structure of the set, better methods are available. For instance, consider the following two definitions of sets:

S_1: $x \supset (x \vee y)$, where x and y are any logic expressions.

Examples: $p \supset (p \vee q)$, $q \supset (q \vee q)$, $(p \vee p) \supset ((p \vee p) \vee p)$, ...

S_2: α, where α may be replaced (independently at each occurrence) according to the following schemes:

$\alpha \leftarrow q$, $\alpha \leftarrow (p \vee \alpha)$, $\alpha \leftarrow \alpha \supset \alpha$.

Examples: q, $p \vee q$, $q \supset q$, $p \vee (p \vee q)$, $(p \vee q) \supset (p \vee q)$, ...

In S_1, x and y are variables in the standard fashion, where each occurrence of the variable is to be replaced by its value. In S_2 we have defined a replacement system, where each separate occurrence of the symbol α may be replaced by any of the given expressions. These may include α, hence lead to further replacements. A legal logic expression exists only when no α's occur.

It is trivial to determine that e is a member of the set of expressions defined by S_1, and not so trivial to determine that it is not a member of the set defined by S_2. The difference is that for S_1 we could simply match the expressions against the form and determine directly the values of the variables required to do the job. In the case of S_2 we had essentially to generate-and-test. (Actually, the structure of the replacement system

permits the generation to be shaped somewhat to the needs of the task, so it is not pure generate-and-test, which assumes no knowledge of the internal structure of the generator.)

Figure 10.5 shows the structure of the match method, using the same symbolism as in Fig. 10.4 for the generate-and-test. A key assumption, implicit in calling X and F expressions, is that it is possible to generate the subparts of X and F, and that X and F are equal if and only if corresponding subparts are equal. Thus there are two generators, which produce corresponding subparts of the two expressions as elements. These are compared: if equal, the generation continues; if not equal, a test is made if the element from the form is a variable. If it is, a substitution of the corresponding part of X for the variable is possible, thus making the two expressions identical at that point, and permitting generation to continue. The generators must also produce some special end signal, whose co-occurrence is detected by the compare routine to determine that a solution has been found.

The match procedure sketched in Fig. 10.5 is not the most general one possible. Operations other than substitution can be used to modify the form (more generally, the kernel structure) so that it is equal to X. There can be several such operations with the type of difference between the two elements selecting out the appropriate action. This action can

Problem statement

Given: expressions made up of parts from a set S;
a set of variables $\{v\}$ with values in S;
a form F, which is an expression containing variables;
an expression X.
Find: if X is in the set defined by F; that is,
Find: values for $\{v\}$ such that $X = F$ (with values substituted).

Procedure

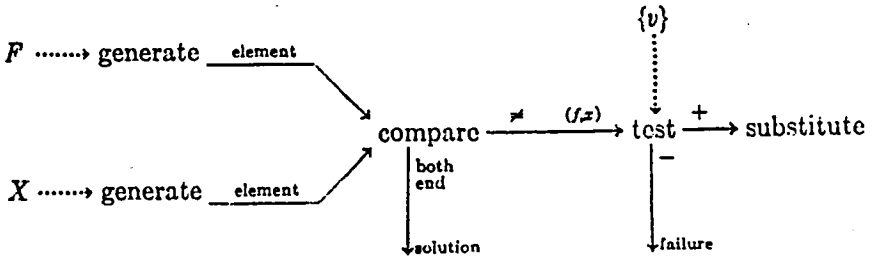


Figure 10.5. Match method.

result in modification of X as well as F. It is possible to write a single procedure that expresses these more general possibilities, but the detail does not warrant it. The essential point is that generation occurs on the parts of the expressions, and when parts fail to correspond it is possible to make a local decision on what modifying operation is necessary (though perhaps not sufficient) for the two expressions to become equal.

Matching is used so pervasively in mathematical manipulation, from algebraic forms to the conditions of a theorem, that our mathematical sophistication leads us not to notice how powerful it is. Whenever a set of possible solutions can be packaged as a form with variables, the search for a solution is no longer proportional to the size of the set of all possible solutions, but only to the size of the form itself. Notice that the generate process in generate-and-test (Fig. 10.4) operates on quite a different set from the generate of the match (Fig. 10.5).

Beside the obvious uses in proving theorems and doing other mathematics, matching shows up in tasks that seem remote from this discipline. One of them, as shown below, is inducing a pattern from a part. Another use is in answering questions in quasi-natural language. In the latter, information is extracted from the raw text by means of forms, with the variables taking subexpressions in the language as values.

3.3. *Hill Climbing*

The most elementary procedure for finding an optimum is akin to generate-and-test, with the addition that the candidate element is compared against a stored element—the best so far—and replaces it if higher. The element often involves other information in addition to the position in the space being searched, for example, a function value. With just a little stronger assumptions in the problem statement, the problem can be converted into an analog of climbing a hill. There must be available a set of operators that find new elements on the hill, given an existing element. That is, new candidate elements are generated by taking a step from the present position (one is tempted to say a "nearby" step, but it is the operators themselves that define the concept of nearness). Thus the highest element so far plays a dual role, both as the base for generation of new elements and as the criterion for whether they should be kept.

Figure 10.6 provides the capsule formulation of hill climbing. Generation is over the set of operators, which are then applied to the best x so far, until a better one is found. This method differs from the various forms of steepest ascent in not finding the best step from the current position before making the next step.

Problem statement

Given: a comparison of two elements of a set $\{x\}$ to determine which is greater;
a set of operators $\{q\}$ whose range and domain is $\{x\}$ [i.e., $q(x) = x'$, another element of $\{x\}$].

Find: the greatest $x \in \{x\}$.

Procedure

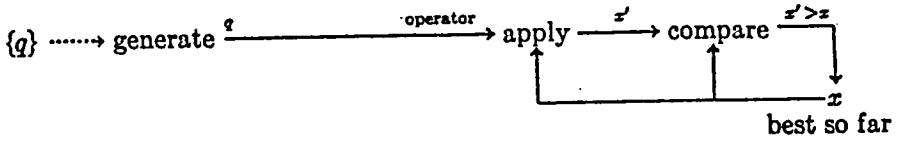


Figure 10.6. Hill climbing.

A great deal has been written about hill climbing, and the interested reader will find a thorough discussion within the context of more elaborate methods for finding optima in Chapter 9 on structured heuristic programming. Here we note only the familiar fact that the method does not guarantee to find the best element in the space. An additional condition, unimodality, is required; otherwise the procedure may end up on a local peak, which is lower than the highest peak. Actually, unimodality is not easy to define in the most general situation to which hill climbing applies, since the underlying space (which is "seen" only through the operators) need not have neighborhoods in the sense required to define a local peak.

Hill climbing shows up in a subordinate way in many heuristic programs, especially in the adaptive setting of parametric values. For example, in one program that attempted to construct programs satisfying certain criteria [9], the various basic instructions were selected at random and used to extend the program built so far. The entire program was an elementary form of heuristic search, discussed in Section 3.4. But superimposed on it was a hill-climbing program that gradually modified the probabilities of selecting the basic instructions so as to maximize the yield of programs over the long haul. The operators randomly jiggled the selection probabilities around (always maintaining their sum equal to one). The comparison was made on a statistical basis after observing the performance with the new probabilities for, say, 100 randomly selected problems.

Management science is much concerned with seeking optima, although, as mentioned above, the methods used are more elaborate. This can be

illustrated by a heuristic program developed by Kuehn and Hamburger [11] for locating warehouses so as to balance the costs of distribution (which decrease with more warehouses) and the costs of operation (which increase with more warehouses). The program consists of three separate optimizers: a cascade of a generate-and-test optimizer and a steepest ascent optimizer, followed by a simple hill climber, followed by a set of simple generate-and-test optimizers. Figure 10.7 gives the problem statement (leaving out details on the cost functions) and an indication of how the problem is mapped into the problem space[4] for optimization. Three operators are defined, corresponding to the three separate stages already mentioned. The procedure is given for the first stage (called the Main Routine), but not for the other two (called the Bump and Shift Routine).

The elements of the problem space consist of all subsets of warehouses, taken from a list of possible sites (which is a subset of the total set of sites with customer demand). The program builds up the set of warehouses by the operation of adding one warehouse at a time. The actual data structure corresponding to the element consists not only of the list of warehouses but also the assignment of each customer to a warehouse, the partial cost borne by that warehouse, and the total cost of operating (TC). (That incremental cost calculations are easier to make than calculations starting from scratch is an important aspect of the efficiency of programs such as this one.) The main part of the program simply considers adding new warehouses (i.e., taking steps in the problem space) and comparing these against the current position on total cost. It is a steepest ascent scheme, since it goes through the whole set and then picks the best one. The additional wrinkle is to eliminate from the set of unused warehouses any whose costs are less than the current position, thus depleting the set to be considered. In fact, the first stage terminates when this set becomes empty.

The operator generator delivers only a fixed subset of all possible warehouse sites. It does this by a simple hill-climbing scheme whereby the best n sites are chosen from the total set on the basis of local cost (LC), which is the cost savings to be made from handling the local demand at the same site as the warehouse (n is a parameter of the program). This cascading of two optimizers keeps the second one from becoming excessively costly.

The next two stages (the Bump and Shift Routine) make minor

[4] We often use the term problem space to refer to the set of potential solutions as defined by the problem statement of a method. It includes the associated operations for moving around in the space.

Problem statement

Given: a set of customers, $\{c\}$, with locations and sales volume;
a set of factories, $\{f\}$, with locations;
a set of warehouse sites, $\{w\}$, with transportation costs to customers and to factories, and operating costs.

Find: a set of warehouses that minimizes total costs.

Problem space for hill climbing

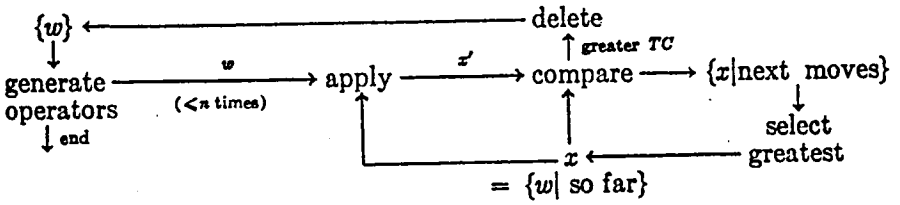
Elements: $\{x|x$ is a subset of $\{w\}\}$,
for each x can compute $TC(x)$.

Initial element: the null set.

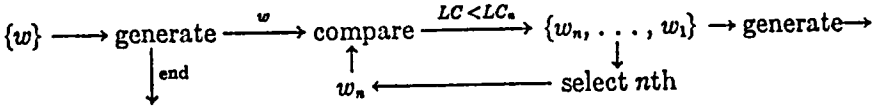
Desired element: the element with lowest TC.

Operators: 1. Add w to x;
2. delete w from x;
3. move $w \in x$ to location of customer of w.
Note: all these permit incremental calculation of TC, since only paired comparisons with existing w for each customer affected are required.

Procedure for stage 1 (operator 1)



Generate operators:



$TC(x)$ = total cost of x to supply all customers.
$LC(w)$ = local cost of w to supply customers at location of w.

Figure 10.7. Warehouse location heuristic program.

adjustments in the element (the set of warehouses) that results from the first phase. Warehouses are successively eliminated if they fail to pay for themselves. This is hill climbing if the order of elimination affects subsequent costs; otherwise it is simply generating through the parts of the system, making local changes. Note that no new warehouses are

added to the solution element after deletion. Finally, as the last stage, each warehouse is moved around locally in its own territory to find the most advantageous location. This stage constitutes a series of independent generate-and-test optimizations, since no interaction between warehouses is involved.

3.4. Heuristic Search

The best-known method in heuristic programming is the one whereby the problem is cast as a search through an exponentially expanding space of possibilities—as a search which must be controlled and focused by the application of heuristics. All of the game-playing and theorem-proving programs make use of this method, as well as many of the management science applications [13].[5]

Figure 10.8 gives the most elementary variant of the method. It assumes a space of elements, the problem space, which contains one element representing the initial position, and another representing the final or desired position. Also available is a fixed set of operators, which when applied to elements in space produce new elements. (Operators need not always be applicable.) The problem is to produce the final desired position, starting at the initial one.

With only this information available, the method involves a search that expands in a tree-like fashion. The initial element x_0 is the initial current position; operators are selected and applied to it; each new element is compared with x_d to see whether the problem is solved; if not, it is added to a list of obtained positions (also called the "try list" or the "subproblem list"); and one of these positions is selected from which to continue the search. If about B of the operators applied are applicable to an obtained position, about B^D elements will have been reached after D steps.

The search is guided (i.e., the tree pruned) by appropriate selection and rejection of operators and elements. The flow diagram provides a scheme upon which the various possibilities can be localized. The most elementary ones are unconditional: a rule for operator selection or for element selection. The latter is often accomplished by keeping an ordered list of elements and simply selecting the first one on the list; hence the order is dictated by the insertion process. The simplest rules have been given names. Thus, if the list is last-in-first-out (so that insertion is al-

[5] A few game players are exceptions. They use a recognition method that learns to associate to each game position (or class of positions) a good move. Although theoretically capable of handling complex games through the development of an appropriate classification, this method has not been used in any but simple games.

Problem statement

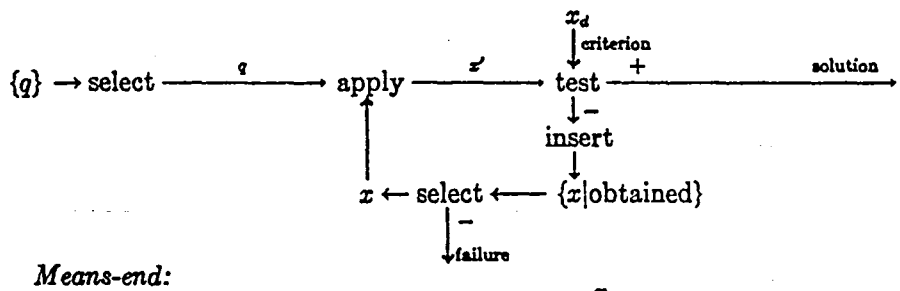
Given: a set $\{x\}$, the problem space;
a set of operators $\{q\}$ with range and domain in $\{x\}$;
an initial element, x_0;
a desired element, x_d.

Find: a sequence of operators, $q_1, q_2, \ldots, q_n$, such that they transform x_0 into x_d:

$$q_n[q_{n-1} \ldots q_1(x_0) \ldots] = x_d$$

Procedure

Basic:



Means-end:

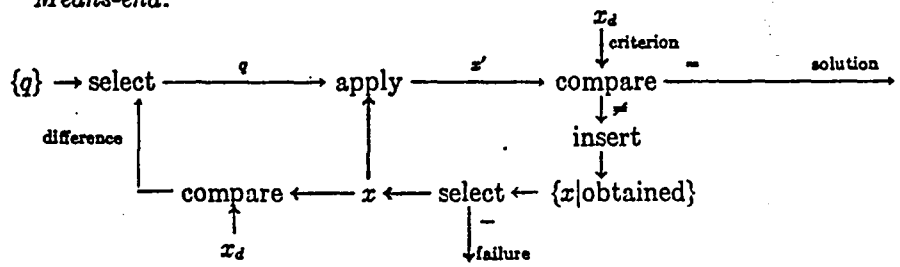


Figure 10.8. Heuristic search method.

ways at the front), the resulting search is *depth first*. This scheme was used by almost all early game-playing programs. If the list is first-in-first-out (so that insertion is always at the end), the resulting search is *breadth first*. This scheme has been used by some theorem provers.

An element may be completely rejected at the time of insertion. One of the most frequent heuristics is to reject if the element is already in the obtained list, that is, to avoid duplication. (This is a heuristic, since, though always beneficial, it requires extra effort and memory; thus it may not pay compared to, say, additional search.) Information already available in the scheme can be used; for instance, a rather important

variant is called means-ends analysis. The current element x is compared with the desired element x_d, and the resulting difference is used to select the operator, that is, the operator (means) is selected with a view toward the end. The flow diagram for this variant is given in the figure below the basic heuristic search.

The situation described in Fig. 10.8 is overly simple in several respects. Initially, a set of elements may be given, rather than just one, with the search able to start from any of them. Likewise, finding any one of a set of elements can be desired, rather than just finding a single one. This final element (or set) can be given by a test instead of by an element, although this has consequences for variants such as means-ends analysis. More important, the operators need not involve only a single input and a single output. In logic, for instance, important rules of inference, such as *modus ponens*, take two inputs and deliver a single output: from a and $a \supset b$ infer b. Thus we can have multiple inputs for an operator. Similarly, if one is working backwards in logic (that is, from the conclusion to premises that make this conclusion follow), *modus ponens* shows up as an operator that has a single input but a set of outputs: to get b, prove a and $a \supset b$. Furthermore, *all* of the outputs must be obtained; thus independent subproblems must radiate from each element of the output set in order to solve the problem.

Figure 10.9 shows one of the early theorem provers, LT (the Logic Theorist), which worked on elementary symbolic logic [15]. It is a heuristic search, using a breadth first strategy. However, it also uses generate-and-test and match, so that, like the warehouse location program, it has a composite structure. Comparison of LT with the basic heuristic search method will show that there are two unexpected twists to formulating the task of theorem proving for heuristic search. First, the problem has to be turned around, so that LT works backward from the original goal toward the given theorems. Thus the rules of inference must be expressed in an inverse sense. Second, the assumed theorems enter into the task both in the generation of operations and in the test. This actually reflects a restriction to the generality of LT, since it insists that one of the two expressions in the backward rules of inference be tied immediately to a theorem, rather than being a subproblem which need only make contact eventually.

The only elaborations of LT from the basic heuristic search procedure are the insertion of a test for similarity between t and x before trying to apply the operator, and the rejection of duplicate elements, which requires keeping a list of the elements already tried. The test for solution is elaborated in the minimal generate-and-test way to take into account

Problem statement

Given: the rules of inference of propositional logic;
a set of theorems, $\{t\}$, assumed valid;
a theorem, x_0.

Find: a proof of x_0 from the assumed theorems.

Problem space for heuristic search

Elements: logic expressions, $\{x\}$.

Initial element: theorem to be proved, x_0.

Desired elements: any of the assumed theorems, $\{t\}$.

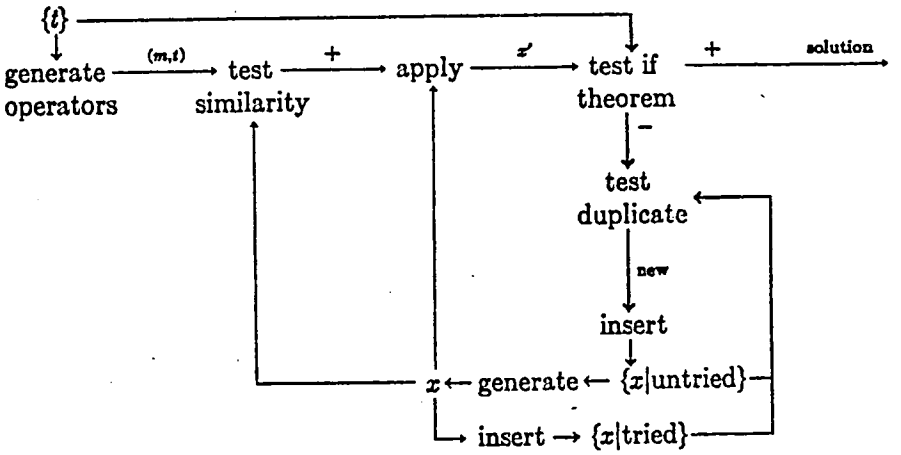
Operators: pairs (m, x), where t is any assumed theorem and m is an expression of the rules of inference, working backwards:

MDt (Detachment): $(t{:}a \supset b, x{:}b) \rightarrow a$

MChF (Chaining forward): $(t{:}a \supset b, x{:}a \supset c) \rightarrow b \supset c$

MChB (Chaining backward): $(t{:}a \supset b, x{:}d \supset b) \rightarrow d \supset a$

Procedure



Generate operators: {MChB, MChF, MDt} → generate $\xrightarrow{m}$ generate $\xrightarrow{(m,t)}$ (with $\{t\}$)

Test if theorem: $\{t\} \rightarrow$ generate $\xrightarrow{t}$ match $\xrightarrow{+}$; x' ; $-$

Apply: $\xrightarrow{(t,m)}$ match $\xrightarrow{+}$ construct $\xrightarrow{x'}$; x ; output

Match: operations are substitution and the definition: $a \supset b \equiv \sim a \vee b$

Figure 10.9. LT: Logic Theorist.

that any of the set of theorems will do. Although we have not shown the internal structure of the match, it does use definitions as well as substitutions to make a theorem and an expression the same.

3.5. Induction

Figure 10.10 shows a task that clearly involves inductive reasoning: you are to determine which of the figures 1–5 bears the same relationship to figure C as figure B does to figure A [4]. Similar problems exist in extrapolating series [22]: for example, what is the blank in *abcbcdcd*_?

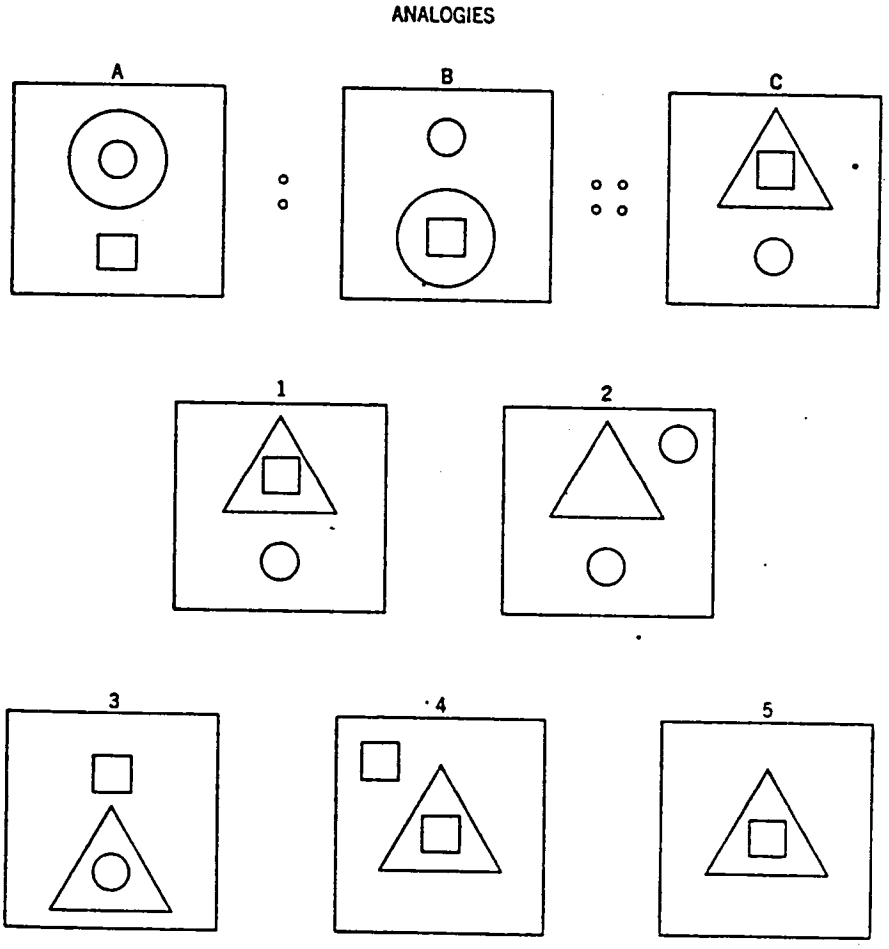


Figure 10.10. Analogies task.

Another similar task is to discover a concept, given a sequence of items, some of which exemplify the concept whereas others do not [8]: for example, if xoxox, xoxxo, oxoxo are positive instances and xooxo, xxoox, xoooo are negative instances, what is oxxxo?

Computer programs have been constructed for these tasks. They show a certain diversity due to the gross shape of the task; that is, the task of Fig. 10.10 gives one instance of the concept (A:B) and five additional possible ones {C:1, C:2, . . . , C:5}, whereas the series provides a long sequence of exemplars if one assumes that each letter can be predicted from its predecessors. However, most of the programs use a single method, adapted to the particular top-level task structure.[6] Figure 10.11 gives the method, although somewhat more sketchily than for the others.

The first essential feature of the method is revealed in the problem statement, which requires the problem to be cast as one of finding a function or mapping of the given data into the associated (or predicted) data. The space of functions is never provided by the problem poser—certainly not in the three examples just presented. Often it is not even clear what the range and domain of the function should be. For the series extrapolation task, to view $\{x{:}y\}$ as $\{a{:}b,\ ab{:}c,\ abc{:}b, \ldots,$ *abcbcdcd*:_} is already problem solving. Thus the key inductive step is the assumption of some space of functions. Once this is done the problem reduces to finding in this space one function (or perhaps the simplest one) that fits the exemplars.

The second essential feature of the method is the use of a form or kernel for the function. This can be matched (in the sense of the match method) against the exemplars. Evidence in the items then operates directly to specify the actual function from the kernel. Implicit in the procedure in Fig. 10.11 is that, inside the match, generation on the kernel (refer back to Fig. 10.5) produces, not the parts of the kernel itself, but the predictions of the y associated with the presented x. However, parts of the kernel expression must show through in these predictions, so that the modification operations of the match can specify or modify them in the light of differences. When the kernel expression actually has variables in it, the prediction from the kernel is sometimes a variable. Its value can be made equal to what it should be from the given $x{:}y$, and thus the kernel expression itself specified. Often the modification operations are like linguistic replacement rules, and then the matter is somewhat more complex to describe.

[6] Most, but not all. Several adapt the paradigm used for pattern recognition programs. In addition, a method called the method of successive differences is applicable to series extrapolation where the terms of the series are expressed as numbers.

Problem statement

Given: a domain $\{x\}$;
a range $\{y\}$;
a generator of associated pairs $\{x;y\}$.

Find: a function f with domain $\{x\}$ and range $\{y\}$ such that $f(x) = y$ for all $\{x:y\}$.

Additional assumption (almost never given with the problem statement, and therefore constituting the actual inductive step):

Given: a set of functions $\{f\}$ constructable from a set of kernel forms $\{k\}$.

Procedure

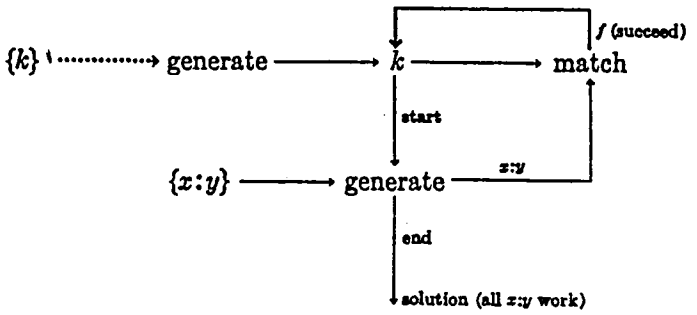


Figure 10.11. Induction method.

It is not often possible to express the entire space of functions as a single form (whence a single match would do the job). Consequently a sequential generation of the kernels feeds the match process. Sometimes clues in the exemplars are used to order the generation; more often, generation is simply from the simplest functions to the more complex.

This method is clearly a version of "hypothesis-and-test." However the latter term is used much more generally than to designate the class of induction problems handled by this method. Furthermore, there is nothing in hypothesis-and-test which implies the use of match; it may be only generate-and-test. Consequently, we choose to call the method simply the induction method, after the type of task it is used for.

3.6. Summary

The set of methods just sketched—generate-and-test, hill climbing, match, heuristic search, and induction—constitutes a substantial fraction of all methods used in heuristic programming. To be sure, this is only a judgment. No detailed demonstration yet exists. Also, one or two im-

portant methods are missing; for example, an almost universally used paradigm for pattern recognition.

Two characteristics of the set of methods stand out. First, explicit references to processes occur in the problem statement, whereas this is not true of mathematical methods, such as the simplex method. Thus generate-and-test specifies that you must have a generator and you must have a test; then the procedure tells how to organize these. This feature seems to be related to the strength of the method. Methods with stronger assumptions make use of known processes whose existence is implied by the assumptions. In the simplex method generation on a set of variables is done over the index, and the tests used are equality and inequality on real numbers. Hence there is no need to posit directly, say, the generator of the set.

The second characteristic is the strong similarity of the methods to each other. They give the impression of ringing the various changes on a small set of structural features. Thus there appear to be only two differences between heuristic search and hill climbing. First, it is necessary to compare for the greater element in hill climbing; heuristic search needs only a test for solution (although it can use the stronger comparison, as in means-ends analysis). Second, hill climbing keeps only the best element found so far, that is, it searches the problem space from where it is. Heuristic search, on the other hand, keeps around a set of obtained elements and selects from it where next to continue the search. In consequence, it permits a more global view of the space than hill climbing—and pays for it, not only by extra memory and processing, but also by the threat of exponential expansion.

Similarly, the difference between match and heuristic search is primarily one of memory for past actions and positions. Our diagram for match does not reveal this clearly, since it shows only the case of a single modification operation, substitution; but with a set of modification operations (corresponding to the set of operators in heuristic search) the match looks very much like a means-ends analysis that never has to back up.

Finally, the more complex processes, such as LT and the warehouse program, seem to make use of the more elementary ones in recognizable combinations. Such combination does not always take the form of distinct units tied output to input (i.e., of closed subroutines), but a flavor still exists of structures composed of building blocks.

In reviewing these methods instruction in the details of artificial intelligence has not been intended. Hopefully, however, enough information has been given to convince the reader of two main points: (1) there is in

heuristic programming a set of methods, as this term was used in the beginning of the paper; and (2) these methods make much weaker demands for information on the task environment than do methods such as the simplex, and hence they provide entries toward the lower, more general end of the graph in Fig. 10.3.

4. THE CONTINUNITY OF METHODS

If the two hypotheses that we have stated are correct, we should certainly expect there to be methods all along the range exhibited in Fig. 10.3. In particular, the mathematical methods of management science should not be a species apart from the methods of artificial intelligence, but should be distinguished more by having additional constraints in the problem statement. Specific mathematical content should arise as statements strong enough to permit reasonable mathematical analysis are introduced.

Evidence for this continuity comes from several sources. One is the variety of optimization techniques, ranging from hill climbing to the calculation methods of the differential calculus, each with increasing amounts of specification. Another is the existence of several methods, such as so-called branch and bound techniques, that seem equally akin to mathematical and heuristic programming. Again, dynamic programming, when applied to tasks with little mathematical structure, leads to procedures which seem not to differ from some of the methods in heuristic programming, for example, the minimax search techniques for playing games such as chess and checkers.

What we should like most of all is that each (or at least a satisfactory number) of the mathematical methods of management science would lie along a line of methods that extends back to some very weak but general ancestors. Then, hopefully, the effect on the procedure of the increasing information in the problem statement would be visible and we could see the continuity directly.

As a simple example, consider inverting a matrix. The normal algorithms for this are highly polished procedures. Yet one can look at inversion as a problem—as it is to anyone who does not know the available theory and the algorithms based on it. Parts of the problem space are clear: the elements are matrices (say of order n), hence include both the given matrix, A, the identity matrix, I, and the desired inverse, X. The problem statement is to find X such that $AX = I$. Simply generating and testing is not a promising way to proceed, nor is expressing X as a form, multiplying out, and getting n^2 equations to solve. Not only are these

poor approaches, but also they clearly are not the remote ancestors of the existing algorithms.

If the inverse is seen as a transformation on A, carrying it into I, a more interesting specification develops. The initial object is A, the desired object is I, and the operators consist of premultiplication (say) by some basic set of matrices. Then, if operators $E_1, E_2, \ldots, E_k$ transform A into I, we have $E_k \cdots E_2 E_1 A = I$, hence $E_k \cdots E_2 E_1 I = A^{-1}$. If the basic operators are the elementary row operations (permute two rows, add one row to another, multiply a row by a constant), we have the basic ingredients of several of the existing direct algorithms (those that use elimination rather than successive approximation). These algorithms prescribe the exact transformations to be applied at each stage, but if we view this knowledge as being degraded we can envision a problem solver doing a heuristic search (or perhaps hill climbing if the approach were monotone). Better information about the nature of the space should lead to better selection of the transformation until existing algorithms are approached.

4.1. An Example: the Simplex Method

The simplex method clearly involves both optimization and search, hence should eventually show kinship with the methods that we have been describing. We should be able to construct a sequence of methods, each with somewhat less stringent conditions and therefore with more general applicablilty but less power. Power can be measured here by applying the method to the original linear programming (LP) problem, where the true nature of the problem is known.

Figure 10.12 reformulates the LP problem and the simplex algorithm in the present notation. Indices have been suppressed as much as possible, partly by using the scalar product, in the interests of keeping the figures uncluttered. The problem statement for the simplex method, which we call SM from now on, is a special case of the LP problem, having equalities for constraints rather than inequalitities, but involving $n + m$ variables rather than just n. The transformation between problems is straightforward (although the original selection of this specialization is not necessarily so).

The elements of the problem space (called bases) are all the subsets of m out of the $n + m$ variables. An element consists of much more information than just the subset of variables, of course, namely, of the entire tableau shown in Fig. 10.2. The operators theoretically could be any rules that replace one subset of m variables with another. In fact, they involve adding just a single variable, hence removing one. This

Problem statement for LP problem

Given: a set of n variables, $\{x\}$, where each $x \geqslant 0$:
let $\bar{x}$ be the n-tuple $(x_1, x_2, \ldots, x_n)$;
a set of m constraints, $\{g = b - \bar{a}\bar{x}\}$:
let the feasible region be $\{\bar{x} | g \geqslant 0\}$;
an objective function, $z = \bar{c}\bar{x}$.

Find: $\bar{x}$ in the feasible region such that z is maximum.

Problem statement for SM, the simplex method

Given: a set of $n + m$ variables, $\{x\}$, where each $x \geqslant 0$:
let $\bar{x}$ be the $(n + m)$-tuple $(x_1, x_2, \ldots, x_{n+m})$;
a set of m constraints, $\{g = b - \bar{a}\bar{x}\}$:
let the feasible region be $\{\bar{x} | g = 0\}$;
an objective function, $z = \bar{c}\bar{x}$.

Find: $\bar{x}$ in the feasible region such that z is maximum.

Note: any LP problem can be solved if this one can. It is a separate problem to provide the translation between them (define $x_{n+i} = g_i$ and determine $\bar{c}$ and the $\bar{a}$ accordingly).

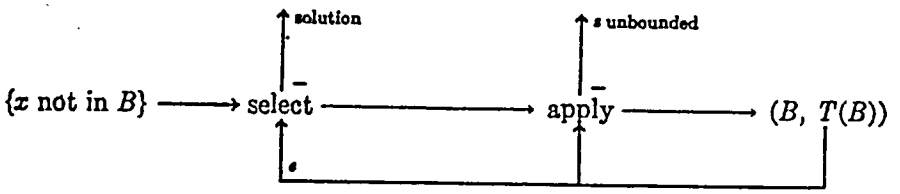
Problem space for SM

Elements: {B (bases), the $\binom{n+m}{m}$ subsets of m variables from $\{x\}$};
with B is associated $T(B)$ (the tableau) containing:
a feasible x such that $x \in B$ implies $x > 0$; otherwise $x = 0$;
the current value of z for $\bar{x}$;
the exchange rate (e) for each x [$-(z - c)$ in tableau];
auxiliary data to permit application of any operator.

Initial element: B_0, a feasible basis (not obtained by SM).

Operators: {x not in B}.

Procedure



Select (pick x with maximum e):

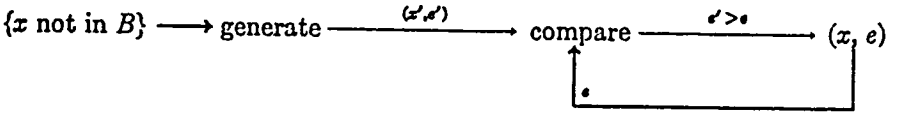


Figure 10.12. SM: reformulation of simplex method.

would still leave $(n-m)m$ operators (any of $n-m$ in, any of m out), except that no choice exists on the one to be removed. Hence, there are just $n-m$ operators, specified by the $n-m$ variables not in the current basis. Applying these operators to the current element consists of almost the entire calculation of the simplex procedure specified in Fig. 10.2 (actually steps 2, 3, and 4), which amounts to roughly $m(n+m)$ multiplications (to use a time-honored unit of computational effort).

The procedure in Fig. 10.12 looks like a hill climber with the comparison missing: as each operator is selected, the new (B,T) is immediately calculated (i.e., the tableau updated) and a new operator selected. No comparison is needed because the selection produces only a single operator, and this is known to advance the current position. The procedure for selecting the operator reveals that the process generates over all potential operators—over all x not in the current basis—and selects the best one with respect to a quantity called the exchange rate (e). Thus the select is a simple optimizer, with one exception (not indicated in the figure): it is given an initial bias of zero, so that only operators with $e > 0$ have a chance.

The exchange rate is the rate of change of the objective function (z) with a change in x, given movement along a boundary of the constraints, where the only variables changing are those already in the basis. Given this kind of exploration in the larger space of the $n+m$ variables, e measures how fast z will increase or decrease. Thus, $e > 0$ guarantees that the compare routine is not needed in the main procedure.

The selection of an x with the maximum exchange rate does not guarantee either the maximum increase from the current position or the minimum number of steps to the optimum. The former could be achieved by inserting a compare routine and trying all operators from the current position; but this would require many times as much effort for (probably) small additional gain. However, since the space is unimodal in the feasible region, the procedure does guarantee that eventually an optimum will be reached.

We need a method at the general end of the scale against which the SM can be compared. Figure 10.13 gives the obvious one, which we call M1. It retains the shape of the problem but without any specific content. The problem is still to optimize a function, f, of n positive variables, subject to m inequality constraints, $\{g\}$. But the only thing known about f and the g is that f is unimodal in the feasible set (which accounts for its descriptive title). This justifies using hill climbing as the procedure. The operators must be any increments to the current position, either positive or negative. Many will produce a location outside the feasible

Problem statement for M1, the unimodal objective method

Given: a set of n variables, $\{x\}$, where each $x \geqslant 0$:
let $\bar{x}$ be the n-tuple $(x_1, x_2, \ldots x_n)$;
a set of m constraints, $\{g(\bar{x}) \geqslant 0\}$:
let the feasible region be $\{\bar{x} | g \geqslant 0\}$;
an objective function $z = f(\bar{x})$;
f is unimodal in the feasible region.
Find: $\bar{x}$ in the feasible region such that z is maximum.

Problem space PS1, for hillclimbing

Elements: $\{\bar{x}\}$.
Initial element: x_0, a feasible solution (not obtained by M1).
Operators: $\{\Delta\bar{x}$, where each Δx is any real number$\}$,
and $\bar{x}' = \bar{x} + \Delta\bar{x}$.

Procedure

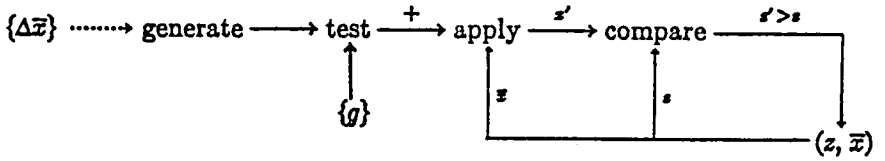


Figure 10.13. M1: unimodal objective method.

region, but these can be rejected by testing against the g. The procedure in the figure does not provide any information on how to generate the operators.

If M1 is compared to SM, several differences are apparent. First, and most striking, the problem space for M1 is n-dimensional Euclidean space, whereas for SM it is a finite set of $\binom{n+m}{m}$ points in $(n+m)$-dimensional space. Thus the search space has been drastically reduced, independently of what techniques are used to search it. Second, and almost as striking, M1 has all of $\{\Delta\bar{x}\}$ as operators (i.e., all of n-dimensional space again), whereas SM has only $n - m$ operators. Third, a unique operator is selected for application on the basis of partial information; it always both applies and improves the position. In M1 there is no reason to expect an operator either to produce a feasible solution or, if it does, to obtain an improvement; thus, extensive testing and comparing must be done. Finally, we observe that the cost per step in M1 (when applied to the same LP problem as SM) is mk, where k is the number of variables

in $\Delta\bar{x}$ and would normally be rather small. Compared to $m(m+n)$ for SM, this yields the one aspect favoring M1. One can take about $(m+n)/k$ steps in the space of M1 for each step in the space of SM. However, the thrashing around necessary at each point to obtain a positive step will largely nullify this advantage.

This list of differences suggests constructing a sequence of methods that extend from M1 to SM, with decreasing spaces and operators and increasing cost per step (to pay for the additional sophistication). Much of the gain, of course, will come without changing problem spaces, but from acquiring better operator selection. There may be relatively few substantial changes of problem space. Figures 10.14 and 10.15 provide

Problem statement for M2, the monotone objective method

Given: the conditions of M1, plus
f is monotone in the feasible region.
Find: $\bar{x}$ in the feasible region such that z is maximum.

Problem space PS2, main hill climbing

Elements: $\{\bar{x}$ on boundary of feasible region (at least one $x = 0$ or $g = 0)\}$.
Initial element: x_0 in feasible region (not obtained by M2).
Operators: $\{x\}$, where $\bar{x}'$ = the point on boundary given by M2*.

Problem statement for M2°, M2-operator method

Given: the conditions of M2, plus
$\bar{x}$ is on the boundary;
$x \in \{x\}$.
Find: $\bar{x}$ on the boundary such that
Δx to increase z not feasible;
all other x unchanged;
z increased.
Additional assumption for efficiency:
$g(\bar{x}) = 0$ can be solved for any x with all other x fixed.

Problem space for M2°, hill climbing

Elements: $\{\bar{x}\}$.
Initial element: x, given by M2 operator.
Operators: $\{\Delta x$, with appropriate sign$\}$, where $\bar{x}' = \bar{x} + \Delta x$.

Problem space PS1 for M1

Used as backup when PS2 terminates without optimum.

Figure 10.14. M2: monotone objective method.

Problem statement for M3, the consistent exchange problem

Given: the conditions of M2, plus
if an x is exchanged for other variables by moving along a maximal boundary, then Δz has a consistent sign.

Find: $\bar{x}$ in the feasible region such that z is a maximum.

Problem space PS3, main hill climbing

Elements: {$\bar{x}$ on the maximal boundary of feasible set; i.e., no x can be changed to increase z, holding other x fixed}.

Initial element: x_0, a feasible solution on the maximal boundary (not obtained by M3).

Operators: $\{x\}$, where $\bar{x}'$ = the point on the maximal boundary given by M3*.

Problem statement for M3°, M3-operator method

Given: the condition of M3, plus
$\bar{x}$ is on a maximal boundary;
$x \in \{x\}$.

Find: $\bar{x}$ on the maximal boundary, such that exchange for x to increase z is not feasible;
z increased.

Additional assumption for efficiency:
any system of k equations, $\{g(\bar{x}) = 0\}$, can be solved for any set of k variables with the others fixed.

Problem space for M3°

Elements: $\{x\}$.

Initial element: x, given by M3 operator.

Operators: {Δx, with appropriate sign} and {$\Delta\bar{x}$, subsets of $\{x\}$}, where $\bar{x}' = \bar{x} + \Delta\bar{x}$.

Figure 10.15. M3: consistent exchange method.

two that seem to reflect some of the major boundaries to be crossed in getting from M1 to SM.

Figure 10.14 shows method M2, which adds the assumption that the objective function, f, is monotone in the feasible region. In the LP problem $\partial f/\partial x$ is constant, but this is not required to justify the main conclusion; namely, that if a given change in a varible is good, more change in the same direction is better. The effect of this is to create new operators and, through them, a new space. The basic decision is always to drive a varia-

ble to a boundary (in the direction of increasing z, of course). Thus the space for M2 becomes the boundary set of the original space for M1 (those points where at least one of the constraints, including the $x \geqslant 0$, attains zero). The operators in the space of M2 are full steps to a boundary (what are sometimes called macromoves in artificial intelligence). Now, finding the boundary is still a problem, although a more manageable one. Thus M2 has a second problem method, M2*, for this purpose. As described in Fig. 10.14 it can be a simple hill climber.

An additional strong assumption has been made in the procedure of M2, namely, that only changes in a single variable, x, will be considered. This reduces the number of operators, as well as making the operator submethod M2* simpler. It is not justified by the assumptions of the problem statement, however, and consequently M2 will terminate at suboptimal positions where no single variable can be changed to increase z without decreasing some other variables. (This is often called the maximal or the Pareto optimum set.) Rather than relax the operators to a wider class, the original method, M1, is held in reserve to move off the maximal set. (However, if done with small steps, this is extremely inefficient for the LP problem, since the system just jitters its way slowly up a bounding plane.)

The description of M2* gives an additional assumption: each equation $g(\bar{x})$ can be solved directly for any x, given that the values of the other x's are determined. This permits direct calculation of the extreme value of x on a boundary that is maximal. Slightly stronger conditions on the g's allow determination of the first constraint to become binding, without multiple evaluations.

Figure 10.15 shows method M3, which adds the assumption that the exchange rate (e) always has a consistent sign as one moves along the feasible region, in response to introducing a variable x (what we have called exchanging). Again, in the LP problem e is constant, but this is not required to justify the main conclusion: that in a maximal situation, if adjustments are made in other variables to allow a particlular variable to increase, and the gain from the exchange is positive, it will always be positive; hence the new variable should be exchanged for as much as possible, namely, until another boundary is reached. This assumption not only allows a better way of dealing with the maximal cul-de-sac than does M2, with its regression to M1, but also permits the problem space to be changed radically a second time.

The elements of the space now become the set of maximal points, thus a small subset of the space of M2. The operators remain the same; the individual variables. The application of an operator again cor-

responds to the solving of a subproblem, hence is accomplished by a submethod, M2*. The problem is as follows: given x (with a positive exchange rate), to advance it as far as possible. This means solving the constraints simultaneously for the variables, so as to remain on a boundary. As a change in the selected x is made, the current x moves off the maximal boundary by violating either the constraints or maximality. Adjustments must be made in the other x's to restore these conditions. What the new clause in the problem statement provides is not a way of making the adjustments, but a guarantee that if a change is once found that does increase z (after adjustment) it should be pushed to the limit.

We have not described M3*, other than to indicate the available operators. At its most general (i.e., assuming no other information), it requires a two-stage process, one to discover a good direction and the other to push it. The latter is again a two-stage process, one to change the selected $\bar{x}$ and the other to make the adjustments. We have included an additional assumption, similar to the one for M2*, that a direct way exists of solving systems of contraints for some variables in terms of others. This clearly can make an immense difference in the total efficiency of problem solving but does not alter the basic structuring of the task.

M3 is already a recognizable facsimile of SM. The space has been cut to all subsets of the variables, although the final contraction to subsets of m variables has not occurred. (It is implicit in the problem statement of M3, with some mild conditions on the g's, but has not been brought out.) The operators of M3 and SM are the same. More precisely, they are isomorphic—the process of applying an operator is quite different in the two methods. There are still some steps to go. The kinds of methods that are possible for the operator need explication. They are represented in M3* only by the assumption that systems of equations can be solved. But the schemes in SM use special properties of linear systems. Similarly, we have not explored direct calculation of the exchange rates, with the subsequent replacement of comparison in the main method by comparison in the operator, to avoid expensive computation.

We have not carried this example through in complete detail, nor have we established very many points on the path from a crude hill climber to SM. The two points determined are clearly appropriate ones and capture some of the important features of the method. They are not unexpected points, of course, since linear programming is well understood. The viewpoint underlying the analysis is essentially combinatorial, and such aspects have been thoroughly explored (e.g., see [23]). If these intermediate problems have any peculiar flavor, it is that they become established where the search spaces change, and these need not always

correspond to nice mathematical properties, abstractly considered. Thus convexity is not posited and its implications explored; rather a change of search space is posited and the problem statement that admits it sought.

A single ancestral lineage should not be expected. Just as theorems can have many proofs, so methods can have many decompositions of their information. In fact, in one respect at least the line represented by M2 and M3 does violence to SM. It never recognizes the shift of problem into a set of equality constraints with the consequent change in dimensionality. Thus, the g's and the x's are handled separately, whereas it is a very distinct feature of the simplex procedure that it handles them uniformly. One could easily construct another line starting from SM, which would preserve this feature. (It would face a problem of making the transition to M1.)

The examples selected—linear programming and matrix inversion—are certainly ones that seem most amenable to the kind of analysis we have proposed. If we considered methods, say, for determining inventory levels, the story might be different. Nevertheless, perhaps the case for continuity between weak and strong methods has been made plausible.

5. HUMAN BEHAVIOR IN ILL-STRUCTURED PROBLEMS

In the two issues discussed so far—the existence of weak methods, and the continuity between weak and strong methods—we have not seemed to be dealing directly with ill-structured problems. To re-evoke the concern of Reitman, the problem statements that we have exhibited seem quite precise. (Indeed, we took pains to make them so and in a more technical exposition would have completely formalized them.) According to our hypotheses the world is always formalized, seen from the viewpoint of the methods available, which require quite definite properties to operate. A human problem solver, however, would not feel that a problem was well structured just because he was using a method on it. Our second hypothesis identifies this feeling with the low power of the applicable methods.

The concern just expressed is still well taken. If we examine some problem solvers who are working on "really ill-structured" problems, what will we find? They will necessarily be human, since as noted earlier, men are the gatekeepers of this residual class of problems. Thus we cannot observe their problem-solving processes directly but must infer them from their behavior.

To have something definite in mind consider the following problem solvers and tasks:

A financial adviser: what investments should be made in a new account?
A foreman: is a given subordinate well adjusted to his work?
A marketing executive: which of two competitors will dominate a given market to which his firm is considering entry?

None of these problems is as ill structured as the proverbial injunctions to "know thyself" (asked of every man) and to "publish or perish" (asked of the academician). Still they are perhaps more typical of management problems than these two almost completely open-ended problems. They do have the feature of most concern to Reitman; namely, neither the criteria for whether a solution is acceptable nor the data base upon which to feed are particularly well defined.

The framework we have been using says that below the surface we should discover a set of methods operating. Our two hypotheses assert, first, that we should find general but weak methods; and, second, that we should not find methods that deal with the unstructured aspects (however they come to be defined) through any mechanism other than being general enough to apply to a situation with little definite information.

Our first implication would seem to be upset if we discover that the human being has available very strong methods that are applicable to these ill-structured problems. This is a rather difficult proposition to test, since, without the methods themselves to scrutinize, we have very little basis for judging the nature of problem solving. Powerful methods imply good solutions, but if only men solve the problem, comparative quality is hard to judge.

The three tasks in our list have the virture that comparisons have been made between the solutions obtained by human effort and those obtained by some mechanical procedure. For the second two the actual tasks are close nonmanagement analogs of the tasks listed. However, they all have the property (implicit in our list) that the problem solver is a man who by profession is concerned with solving the stated type of problem. This condition is important in discussing real management problems, since the capabilities of a novice (e.g., a college student used as a subject in an experiment) may differ considerably from those of the professional. In particular, the novice may differ in the direction of using only very general reasoning abilities (since he is inexperienced), whereas the professional may have special methods.

In all of the cases the result is the same. Rather simple mechanical procedures seem to do as well as the professional problem solver or even better; certainly they do not do worse.

The first task was investigated by Clarkson [1] in one of the early simulation studies. He actually constructed a program to simulate a trust investment officer in a bank. Thus the program and the human being attain the same level of solution. The program itself consists of a series of elementary evaluations as a data base, plus a recognition structure (called a discrimination net) to make contact between the specific situation and the evaluation; there are also several generate-and-tests. Thus the program does not have any special mechanisms for dealing with ill-structuredness. Indeed it deals with the task in a highly structured way, though with a rather large data base of information. The key point is that the human being, who can still be hypothesized to have special methods for ill-structured situations (since his internal structure is unknown), does not show evidence of this capability through superior performance.

The second task in its nonmanagement form is that of clinical judgment. It has been an active, substantial—and somewhat heated—concern in clinical psychology ever since the forties. In its original form, as reviewed by Meehl [12], it concerned the use of statistical techniques versus the judgments of clinicians. With the development of the computer it has broadened to any programmed procedure. Many studies have been done to confront the two types of judgment in an environment sufficiently controlled and understood to reveal whether one or the other was better. The results are almost uniformly that the programmed procedures perform no worse (and often better) than the human judgment of the professional clinician, even when the clinician is allowed access to a larger "data base" in the form of his direct impressions of the patient. Needless to say, specific objections, both methodological and substantive, have been raised about various studies, so the conclusion is not quite as clear-cut as stated. Nevertheless, it is a fair assertion that no positive evidence of the existence of strong methods of unknown nature has emerged.[7]

The third task is really an analog of an analog. Harris [7], in order to investigate the clinical versus statistical prediction problem just discussed, made use of an analog situation, which is an equally good analog to the marketing problem in our list. He tested whether formulas for predicting the outcome of college football games are better than human judgment. To get the best human judgments (i.e., professional) he made use of coaches of rival teams. Although there is a problem of bias, these coaches clearly have a wealth of information of as professional a nature

[7] A recent volume [10] contains some recent papers in this area, which provide access to the literature.

as the marketing manager has about the market for his goods. On the program side, Harris used some formulas whose predictions are published each week in the newspapers during the football season. An unfortunate aspect of the study is that these formulas are proprietary, although enough information is given about them to make the study meaningful. The result is the same: the coaches do slightly worse than the formulas.

Having found no evidence for strong methods that deal with unstructured problems, we might feel that our two hypotheses, are somewhat more strongly confirmed. However, unless the weak methods used by human beings bear some relationship to the ones we have enumerated, we should take little comfort. For our hypotheses take on substantial meaning only when the weak methods become explicit. There is less solid evidence on what methods people use than on the general absence of strong methods. Most studies simply compare performance, and do not attempt to characterize the methods used by the human problem solver. Likewise, many of the psychological studies on problems solving, although positive to our argument [14], employ artificial tasks that are not sufficiently ill structured to aid us here. The study by Clarkson just reviewed is an exception, since he did investigate closely the behavior of his investment officer. The evidence that this study provides is positive.

Numerous studies in the management science literature might be winnowed either to support or refute assertions about the methods used. Work in the behavioral theory of the firm [2], for instance, provides a picture of the processes used in organizational decision making that is highly compatible with our list of weak methods—searching for alternatives, changing levels of aspiration, etc. However, the characterizations are sufficiently abstract that a substantial issue remains whether they can be converted into methods that really do the decision making. Such descriptions abstract from task content. Now the methods that we have described also abstract from task content. But we know that these can be specialized to solve the problems they claim to solve. In empirical studies we do not know what other methods might have to be added to handle the actual detail of the management decision.

Only rarely are studies performed, such as Clarkson's, in which the problem is ill structured, but the analysis is carried out in detail. Reitman has studied the composition of a fugue by a professional composer [18], which is certainly ill structured enough. Some of the methods we have described, such as means-ends analysis, do show up there. Reitman's characterization is still sufficiently incomplete, however, that no real evidence is provided on our question.

6. DIFFICULTIES

We have explored three areas in which some positive evidence can be adduced for the two hypotheses. We have an explicit set of weak methods; there is some chance that continuity can be established between the weak and the strong methods; and there is some evidence that human beings do not have strong methods of unknown nature for dealing with ill-structured problems. Now it is time to consider some difficulties with our hypotheses. There are several.

6.1. The Many Parts of Problem Solving

At the beginning of this essay we noted that methods were only a part of problem solving, but nevertheless persisted in ignoring all the other parts. Let us now list some of them:

Recognition	Information acquisition
Evaluation	Executive construction
Representation	Method construction
Method identification	Representation construction

A single concern applies to all of these items. Do the aspects of problem solving that permit a problem solver to deal with ill-structured problems reside in one (or more) of these parts, rather than in the methods? If so, the discussions of this essay are largely beside the point.

This shift could be due simply to the power (or generality) of a problem solver not being localized in the methods rather than to anything specific to ill-structuredness. The first two items on the list illustrate this possibility. Some problems are solved directly by recognition; for example, who is it that has just appeared before my eyes? In many problems we seem to get nowhere until we suddenly "just recognize" the essential connection or form of the solution. Gestalt psychology has made this phenomenon of sudden restructuring central to its theory of problem solving. If it were true, our two hypotheses would certainly not be valid. Likewise for the second item, our methods say more about the organization of tests than about the tests themselves. Perhaps most of the power resides in sophisticated evaluations. This would work strongly against our hypotheses. In both examples it is possible, of course, that hypotheses of similar nature to ours apply. In the case of evaluations, for example, it might be that ill-structured problems could be handled only because the problem solver always had available some dis-

tinctions that applied to every situation, even though with less and less relevance.

The third item on the list, representation of problems, also raises a question of the locus of power (rather than of specific mechanisms related to ill-structured problems). At a global level we talk of the representation of a problem in a mathematical model, presumably a translation from its representation in some other global form, such as natural language. These changes of the basic representational system are clearly of great importance to problem solving. It seems, however, that most problems, both well structured and ill structured, are solved without such shifts. Thus the discovery of particularly apt or powerful global representations does not lie at the heart of the handling of ill-structured problems.

More to the point might be the possibility that only special representations can hold ill-structured problems. Natural language or visual imagery might be candidates. To handle ill-structured problems is to be able to work in such a representation. There is no direct evidence to support this, except the general observations that human beings have (all) such representations, and that we do not have good descriptions of them.

More narrowly, we often talk about a change in representation of a problem, even when both representations are expressed in the same language or imagery. Thus we said that Fig. 10.12 contained two representations of the LP problem, the original and the one for the simplex method. Such transformations of a problem occur frequently. For example, to discuss the application of heuristic search to inverting matrices we had to recast the problem as one of getting from the matrix A to I, rather than of getting from the initial data (A, I, $AX = I$) to X. Only after this step was the application of the method possible. A suspicion arises that changes of representation at this level—symbolic manipulation into equivalent but more useful form—might constitute a substantial part of problem solving. Whether such manipulations play any special role in handling ill-structured problems is harder to see. In any event, current research in artificial intelligence attempts to incorporate this type of problem solving simply as manipulations in another, more symbolic problem space. The spaces used by theorem provers, such as LT, are relevant to handling such changes.

Method identification, the next item, concerns how a problem statement of a method comes to be identified with a new problem, so that each of the terms in the problem statement has its appropriate referent in the problem as originally given. Clearly, some process performs this identification, and we know from casual experience that it often requires an

exercise of intellect. How difficult it is for the LP novice to "see" a new problem as an LP problem, and how easy for an old hand!

Conceivably this identification process could play a critical role in dealing with ill-structured problems. Much of the structuring of a problem takes place in creating the identification. Now it might be that methods still play the role assigned to them by our hypotheses, but even so it is not possible to instruct a computer to handle ill-structured problems, because it cannot handle the identification properly. Faced with an appropriate environment, given the method and told that it was the applicable one, the computer still could not proceed to solve the problem. Thus, though our hypotheses would be correct, the attempt to give them substance by describing methods would be misplaced and futile.

Little information exists about the processes of identification in situations relevant to this issue. When the situation is already formalized, matching is clearly appropriate. But we are concerned precisely with identification from a unformalized environment to the problem statement of a method. No substantial programs exist that perform such a task. Pattern recognition programs, although clearly designed to work in "natural" environments, have never been explored in an appropriately integrated situation. Perhaps the first significant clues will come out of the work, mentioned at the beginning of this chapter and still in its early stages, on how a machine can use a hand and eye in coordination. Although the problems that such a device faces seem far removed from management science problems, all the essentials of method identification are there in embryo. (Given that one has a method for picking up blocks, how does one identify how to apply this to the real world, seen through a moving television eye?)

An additional speculation is possible. The problem of identification is to find a mapping of the elements in the original representation (say, external) into the new representation (dictated by the problem statement of the method to be applied). Hence there are methods for the solution to this, just as for any other problem. These methods will be like those we have exhibited. (Note, however, that pattern recognition methods would be included.) The construction of functions in the induction method may provide some clues about how this mapping might be found. As long as the ultimate set of contacts with the external representation (represented in these identification methods as generates and tests) were rather elementary, such a reduction would indeed answer the issue raised and leave our hypotheses relevant.

An important aspect of problem solving is the acquisition of new information, the next item on the list. This occurs at almost every step,

of course, but most of the time it is directed at a highly specific goal; for instance, in method identification, which is a major occasion for assimilating information, acquisition is directed by the problem statement. In contrast, we are concerned here with the acquisition of information to be used at some later time in unforeseen ways. The process of education provides numerous examples of such accumulation.

For an ill-structured problem one general strategy is to gather additional information, without asking much about its relevance until obtained and examined. Clearly, in the viewpoint adopted here, a problem may change from ill structured to well structured under such a strategy, if information is picked up that makes a strong method applicable.

The difficulty posed for our hypotheses by information acquisition is not in assimilating it to our picture of methods. It is plausible to assume that there are methods for acquisition and even that some of them might be familiar; for example, browsing through a scientific journal as generate-and-test. The difficulty is that information acquisition could easily play a central role in handling ill-structured problems but that this depends on the specific content of its methods. If so, then without an explicit description of these methods our hypotheses cannot claim to be relevant. These methods might not formalize easily, so that ill-structured problems would remain solely the domain of human problem solvers. The schemes whereby information is stored away yet seems available almost instantly —as in the recognition of faces or odd relevant facts—are possibly aspects of acquisition methods that may be hard to explicate.

The last three items on the list name things that can be constructed by a problem solver and that affect his subsequent problem-solving behavior. Executive construction occurs because the gross shape of a particular task may have to be reflected in the top-level structure of the procedure that solves it. The induction method, with the three separate induction tasks mentioned, provides an example. Each requires a separate executive structure, and we could not give a single unified procedure to handle them all. Yet each uses the same fundamental method. Relative to our hypotheses, construction seems only to provide additional loci for problem-solving power. This item could become important if it were shown that solutions are not obtained to ill-structured problems without some construction activity.

The extended discussion of the parts of the problem-solving process other than methods, and the ways in which they might either refute or nullify our two hypotheses, stems from a conviction that the major weakness of these hypotheses is the substantial incompleteness of our

knowledge about problem solving. They have been created in response to partial evidence, and it seems unlikely that they will emerge unscathed as some of these other parts become better known.

6.2. Measures of Informational Demands

Throughout the chapter we have talked as if adding information to a problem statement leads to a decrease in generality and an increase in power. Figure 10.3 is the baldest form of this assertion. At the most general level it seems plausible enough. Here one crudely identifies the number of conditions in the problem statement with the size of the space being searched: as it gets smaller, so the problem solver must grow more powerful. At a finer level of analysis, however, this assertion seems often violated, and in significant ways; for example, a linear programming problem is changed into an integer programming problem by the addition of the constraint that the variables $\{x\}$ range over the positive integers rather than the positive reals. But this makes the problem harder, not easier. Of course, it may be that existing methods of integer programming are simply inefficient compared to what they could be. This position seems tenuous, at best. It is preferable, I think, to take as a major difficulty with these hypotheses that they are built on foundations of sand.

6.3. Vague Information

It is a major deficiency of these hypotheses (and of this chapter) that they do not come to grips directly with the nature of vague information. Typically, an ill-structured problem is full of vague information. This might almost be taken as a definition of such a problem, except that the term vague is itself vague.

All extant ideas for dealing with vagueness have one concept in common: they locate the vagueness in the referent of a quite definite (hence un-vague) expression. To have a probability is to have an indefinite event, but a quite definite probability. To have a subset is to have a quite definite expression (the name or description of the subset) which is used to refer to an indefinite, or vague, element. Finally, the constructs of this chapter are similarly definite. The problem solver has a definite problem statement, and all the vagueness exists in the indefinite set of problems that can be identified with the problem statement.[8]

[8] Reitman's proposals, although we have not described them here, have the same definite character [17]. So also does the proposal by Zadeh for "fuzzy" sets [24].

The difficulty with this picture is that, when a human problem solver has a problem he calls ill structured, he does not seem to have definite expressions which refer to his vague information. Rather he has nothing definite at all. As an external observer we might form a definite expression describing the range (or probability distribution) of information that the subject has, but this "meta" expression is not what the subject has that is this information.

It seems to me that the notion of vague information is at the core of the feeling that ill-structured problems are essentially different from well-structured ones. Definite processes must deal with definite things, say, definite expressions. Vague information is not definite in any way. This chapter implies a position on vague information; namely, that there are quite definite expressions in the problem solver (his problem statement). This is a far cry from a theory that explains the different varieties of vague information that a problem solver has. Without such explanations the question of what is an ill-structured problem will remain only half answered.

7. CONCLUSION

The items just discussed—other aspects of problem solving, the measurement of power and generality, and the concept of vagueness—do not exhaust the difficulties or deficiencies of the proposed hypotheses. But they are enough to indicate their highly tentative nature. Almost surely the two hypotheses will be substantially modified and qualified (probably even compromised) with additional knowledge. Even so, there are excellent reasons for putting them forth in bold form.

The general nature of problems and of methods is no longer a quasi-philosophic enterprise, carried on in the relaxed interstices between the development of particular mathematical models and theorems. The development of the computer has initiated the study of information processing, and these highly general schema that we call methods and problem-solving strategies are part of its proper object of study. The nature of generality in problem solving and of ill-structuredness in problems is also part of computer science, and little is known about either. The assertion of some definite hypotheses in crystallized form has the virtue of focusing on these topics as worthy of serious, technical concern.

These two hypotheses (to the extent that they hold true) also have some general implications for the proper study of management science. They say that the field need not be viewed as a collection of isolated mathematical gems, whose application is an art and which is largely

excluded from the domain of "nonquantifiable aspects" of management.[9] Proper to management science is the creation of methods general enough to apply to the ill-structured problems of management—taking them on their own terms and dealing with them in all their vagueness—and not demanding more in the way of data than the situations provide. To be sure, these methods will also be weak but not necessarily weaker than is inherent in the ill-structuring of the task.

That management science should deal with the full range of management problems is by no means a new conclusion. In this respect these two hypotheses only reinforce some existing strands of research and application. They do, however, put special emphasis on the extent to which the hard mathematical core of management science should be involved in ill-structured problems. They say such involvement is possible.

BIBLIOGRAPHY

1. G. P. E. Clarkson, *Portfolio Selection: a Simulation of Trust Investment*, Prentice-Hall, Englewood Cliffs, N.J., 1962.
2. R. M. Cyert and J. G. March, *A Behavioral Theory of the Firm*, Prentice-Hall, Englewood Cliffs, N.J., 1963.
3. J. Ellul, *The Technological Society*, Knopf, New York, 1964.
4. T. G. Evans, "A Heuristic Program to Solve Geometric Analogy Problems," *Proc. Spring Joint Computer Conference*, Vol. 25, 1964, pp. 327–338.
5. E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, 1963. (Reprints many of the basic papers.)
6. R. W. Floyd, "Assigning Meanings to Programs," *Proc. Am. Math. Soc., Symposium on Applied Mathematics*, Vol. 19, 1967, pp. 19–32.
7. J. Harris, "Judgmental versus Mathematical Prediction: an Investigation by Analogy of the Clinical versus Statistical Controversy," *Behav. Sci.*, 8, No. 4, 324–335 (Oct. 1963).
8. E. S. Johnson, "An Information-Processing Model of One Kind of Problem Solving," *Psychol. Monog.*, Whole No. 581, 1964.
9. T. Kilburn, R. L. Grimsdale, and F. H. Summer, "Experiments in Machine Learning and Thinking," *Proc. International Conference on Information Processing*, UNESCO, Paris, 1959.
10. B. Kleinmuntz (ed.), *Formal Representation of Human Judgment*, Wiley, New York, 1968.
11. A. A. Kuehn and M. J. Hamburger, "A Heuristic Program for Locating Warehouses," *Mgmt. Sci.*, 9, No. 4, 643–666 (July 1963).

[9] One need only note the extensive calls to arms issued to the industrial operations researcher to consider the total decision context, and not merely that which he can quantify and put into his model, to realize the firm grip of this image.

12. P. E. Meehl, *Clinical vs. Statistical Prediction,* University of Minnesota Press, Minneapolis, Minn., 1954.
13. A. Newell and G. Ernst, "The Search for Generality," in *Proc. IFIP Congress 65* (E. W. Kalenich, ed.), Spartan Books, New York, 1965, pp. 17–24.
14. A. Newell and H. A. Simon, "Programs as Theories of Higher Mental Processes," in *Computers in Biomedical Research* (R. W. Stacey and B. Waxman, eds.), Vol. 2, Academic Press, New York, 1965, pp. 141–172.
15. A. Newell, J. C. Shaw, and H. A. Simon, "Empirical Explorations of the Logic Theory Machine: a Case Study in Heuristic," *Proc. Western Joint Computer Conference,* Feb. 1957, pp. 218–230. Reprinted in Ref. [5].
16. A. Newell, J. C. Shaw, and H. A. Simon, "Elements of a Theory of Human Problem Solving," *Psychol. Rev.,* 65, No. 3, 151–166 (May 1958).
17. W. R. Reitman, "Heuristic Decision Procedures, Open Constraints, and the Structure of Ill-Defined Problems," in *Human Judgments and Optimability* (M. W. Shelly and G. L. Bryan, eds.), Wiley, New York, 1964, pp. 282–315.
18. W. R. Reitman, *Cognition and Thought,* Wiley, New York, 1965.
19. A. L. Samuel, "Some Studies in Machine Learning, Using the Game of Checkers," *IBM J. Res. and Devel.* 3, 221–229 (July 1959). Reprinted in Ref. [5].
20. O. Selfridge, "Pattern Recognition and Modern Computers," and G. P. Dinneen, "Programming Pattern Recognition," *Proc. Western Joint Computer Conference,* 1955, pp. 91–93, 94–100.
21. H. A. Simon and A. Newell, "Heuristic Problem Solving: the Next Advance in Operations Research," *Opns. Res.,* 6, No. 1, 1–10 (Jan.–Feb. 1958).
22. H. A. Simon and K. Kotovsky, "Human Acquisition of Concepts for Sequential Patterns," *Psychol. Rev.,* 70, 534–546 (1963).
23. A. W. Tucker, "Combinatorial Algebra of Matrix Games and Linear Programs," in *Applied Combinatorial Mathematics* (E. F. Beckenbach, ed.), Wiley, New York, 1964, pp. 320–347.
24. L. A. Zadeh, "Fuzzy Sets," *Inform. and Control,* 8, 338–353 (1965).

ERRATA

Page 368 Figure 10.1, second line under the figure should read "is not under the control of the inputting process."

Page 369 Paragraph 1.1, line 9 should read "a_{ij}, b_i, c_j, $i = 1, \ldots, m; j = 1, \ldots, n$"

Page 371 Third paragraph in the formula should read "Procedure: compute $x = -b/2a \pm 1/2a \sqrt{b^2 - 4ac}$."

Page 386 Third paragraph, fifth line should read "applied to elements in the problem space produce new elements. (Operators need not"

Page 387 Line 8 in formula should read "$q_n(q_{n-1} \cdots q_1(x_0) \ldots) = x_d$"

Page 389 Line 10 change (m,x) to (m,t)

Page 389 Line 4 from the bottom should read "Apply; $\xrightarrow{(m,t)}$ match $\xrightarrow{+}$ construct $\xrightarrow{x'}$

Page 395 Paragraph 4.1., line 5 change "applicablilty" to "applicability"

Page 400 Line 3 from the bottom change "varible" to "variable"

Page 401 Bottom line replace (;) with (:)

Appendix D

Nils J. Nilsson. "Artificial Intelligence," pp 778-801, in Volume 4, *Information Processing 74*, Proceedings of IFIP Congress 74, organized by the International Federation for Information Processing, Stockholm, Sweden, August 5-10, 1974, Jack L. Rosenfeld, editor, copyrighted 1974. Reprinted by permission of North-Holland Publishing Company, Amsterdam, The Netherlands.

Appendix D

ARTIFICIAL INTELLIGENCE

Nils J. NILSON

Artificial Intelligence Center, Stanford Research Institute
Menlo Park, California 94025, USA

(INVITED PAPER)

This paper is a survey of Artificial Intelligence (AI). It divides the field into four core topics (embodying the base for a science of intelligence) and eight applications topics (in which research has been contributing to core ideas). The paper discusses the history, the major landmarks, and some of the controversies in each of these twelve topics. Each topic is represented by a chart citing the major references. These references are contained in an extensive bibliography. The paper concludes with a discussion of some of the criticisms of AI and with some predictions about the course of future research.

1. INTRODUCTION

Can we ever hope to understand the nature of intelligence in the same sense that we understand, say, the nature of flight? Will our understanding of intelligence ever be sufficient to help us build working models--machines that think and perceive--in the same way that our understanding of aerodynamics helps us build airplanes? Intelligence seems so varied. We see it when a chemist discovers the structure of a complex molecule, when a computer plays chess, when a mathematician finds a proof, and even when a child walks home from school. Are there basic mechanisms or processes that are common to all of these activities and to all others commonly thought to require intelligence?

The field of Artificial Intelligence (AI) has as its main tenet that there are indeed common processes that underlie thinking and perceiving, and furthermore that these processes can be understood and studied scientifically. The processes themselves do not depend on whether the subject being thought about or perceived is chemistry, chess, mathematics, or childhood navigation. In addition, it is completely unimportant to the theory of AI who is doing the thinking or perceiving--man or computer. This is an implementational detail.

These are the emerging beliefs of a group of computer scientists claiming to be founding a new science of intelligence. While attempting to discover and understand the basic mechanisms of intelligence, these researchers have produced working models in the form of computer programs capable of some rather impressive feats: playing competent chess, engaging in limited dialogs with humans in English, proving reasonably difficult mathematical theorems in set theory, analysis, and topology, guessing (correctly) the structure of complex organic molecules from mass-spectrogram data, assembling mechanical equipment with a robot hand, and proving the correctness of small computer programs.

Whether the activities of these workers constitute a new scientific field or not, at the very least AI is a major campaign to produce some truly remarkable computer abilities. Like going to the moon or creating life, it is one of man's grandest enterprises. As with all grand enterprises, it will have profound influences on man's way of life and on the way in which he views himself. In this paper, I will try to describe the AI campaign, how it seems to be organized into subcampaigns, who is doing what, some of the current internal controversies, and the main achievements. There is the usual word of caution: I've made some rather large simplifications in attempting to stand aside from the field and look at it with perspective. Not all workers would necessarily agree with what follows.

Before beginning we must discuss an important characteristic of AI as a field, namely, that it does not long retain within it any of its successful applications. Computer aides to mathematicians, such as differential equation solvers, that originated (at least partly) from AI research, ultimately become part of applied mathematics. A system, named DENDRAL, that hypothesizes chemical structures of organic molecules based on mass-spectrogram data is slowly escaping its AI birthplace and will likely become one of the standard tools of chemists. This phenomenon is well-recognized by AI researchers and has led one of them to state that AI is known as the "no-win" field. It exports all of its winning ideas.

On reflection, this is not surprising. When a field takes as its subject matter all of thinking, and then when particular brands of that thinking are applied to chemistry, mathematics, physics, or whatever, these applications become parts of chemistry, mathematics, physics, etc. When people think about chemistry, we call it part of chemistry--not an application of psychology. The more successful AI becomes, the more its applications will become part of the application area.

Destined apparently to lack an applied branch, is there a central core or basic science of AI that will continue to grow and contribute needed ideas to applications in other areas? I think the answer is yes. Just what form these central ideas will ultimately take is difficult to discern now. Will AI be something like biology--diverse but still united by the common structure of DNA? What will be the DNA of AI?

Or will the science of AI be more like the whole of science itself--united by little more than some vague general principles such as the scientific method? It is probably too early to tell. The

present central ideas seem more specific than does the scientific method but less concrete than DNA.

2. WHAT IS HAPPENING IN AI?

2.1 The structure of the field

As a tactic in attempting to discover the basic principles of intelligence, AI researchers have set themselves the preliminary goal of building computer programs that can perform various intellectual tasks that humans can perform. There are major projects currently under way whose goals are to understand natural language (both written and spoken), play master chess, prove non-trivial mathematical theorems, write computer programs, and so forth. These projects serve two purposes. First, they provide the appropriate settings in which the basic mechanisms of intelligence can be discovered and clarified. Second, they provide non-trivial opportunities for the application and testing of such mechanisms that are already known. I am calling these projects the first-level applications of AI.

I have grouped these first-level applications (somewhat arbitrarily) into eight topics shown spread along the periphery of Figure 1. These are the eight that I think have contributed the most to our basic understanding of intelligence. Each has strong ties to other (non-AI) fields, as well as to each other; the major external ties are indicated by arrows in Figure 1.

Basic mechanisms of intelligence and implementational techniques that are common to several applications, I call core topics. It seems to me that there are four major parts to this central core:

- Techniques for modeling and representation of knowledge.
- Techniques for common sense reasoning, deduction, and problem solving.
- Techniques for heuristic search.
- AI systems and languages.

These four parts are shown at the center of Figure 1. Again, we have indicated ties to other fields by arrows. It must be stressed that most AI research takes place in the first-level applications areas even though the primary goal may be to contribute to the more abstract core topics.

If an application is particularly successful, it might be noticed by specialists in the application area and developed by them as a useful and economically viable product. Such applications we might call second-level applications to distinguish them from the first-level applications projects undertaken by the AI researchers themselves. Thus, when AI researchers work on a project to develop a prototype system to understand speech, I call it a first-level application. If General Motors were to develop and install in their assembly plants a system to interpret television images of automobile parts on a conveyor belt, I would call it a second-level application. (We should humbly note that perhaps several second-level applications will emerge without benefit of obvious AI parentage. In fact, these may contribute mightily to AI science itself.)

Thus, even though I agree that AI is a field that cannot retain its applications, it is the second-level applications that it lacks. These belong to the applications areas themselves. Until all of the principles of intelligence are uncovered, AI researchers will continue to search for them in various first-level applications areas.

Figure 1, then, divides work in AI into twelve major topics. I have attempted to show the major papers, projects, and results in each of these topics in Charts 1 through 12, each containing references to an extensive bibliography at the end of this paper. These charts help organize the literature as well as indicate something about the structure of work in the field. By arrows linking boxes within the charts we attempt to indicate how work has built on (or has been provoked by) previous work. The items in the bibliography are coded to indicate the subheading to which they belong. I think that the charts (taken as a whole) fairly represent the important work even though there may be many differences of opinion among workers about some of the entries* (and especially about how work has built on previous work).

Obviously, a short paper cannot be exhaustive. But in this section I will summarize what is going on in AI research by discussing the major accomplishments and status of research in each of the twelve subheadings.

2.2 The core topics

Fundamentally, AI is the science of knowledge--how to represent knowledge and how to obtain and use knowledge. Our core topics deal with these fundamentals. The four topics are highly interdependent, and the reader should be warned that it is probably wrong to attempt to think of them separately even though we are forced to write about them separately.

2.2.1 Common-sense reasoning, deduction, and problem-solving (Chart 1)

By reasoning, etc., we mean the major processes involved in using knowledge: Using it to make inferences and predictions, to make plans, to answer questions, and to obtain additional knowledge. As a core topic, we are concerned mainly with reasoning about everyday, common domains (hence, common sense) because such reasoning is fundamental, and we want also to avoid the possible trap of developing techniques applicable only to some specialized domain. Nevertheless, contributions to our ideas about the use of knowledge have come from all of the applications areas.

There have been three major themes evident in this core topic. We might label these puzzle-solving, question-answering, and common-sense reasoning.

Puzzle-solving. Early work on reasoning concentrated on writing computer programs that could solve simple puzzles (tower of Hanoi, missionaries and cannibals, logic problems, etc.). The Logic Theorist and GPS (see Chart 1) are typical examples. From this work certain problem-solving concepts were developed and clarified in an uncluttered atmosphere. Among these were the concepts of heuristic search, problem spaces and states, operators (that transformed one problem state into another), goal and subgoal states, means-ends analysis, and reasoning backwards. The fact

*In particular, some might reasonably claim machine vision (or more generally, perception) and language understanding to be core topics.

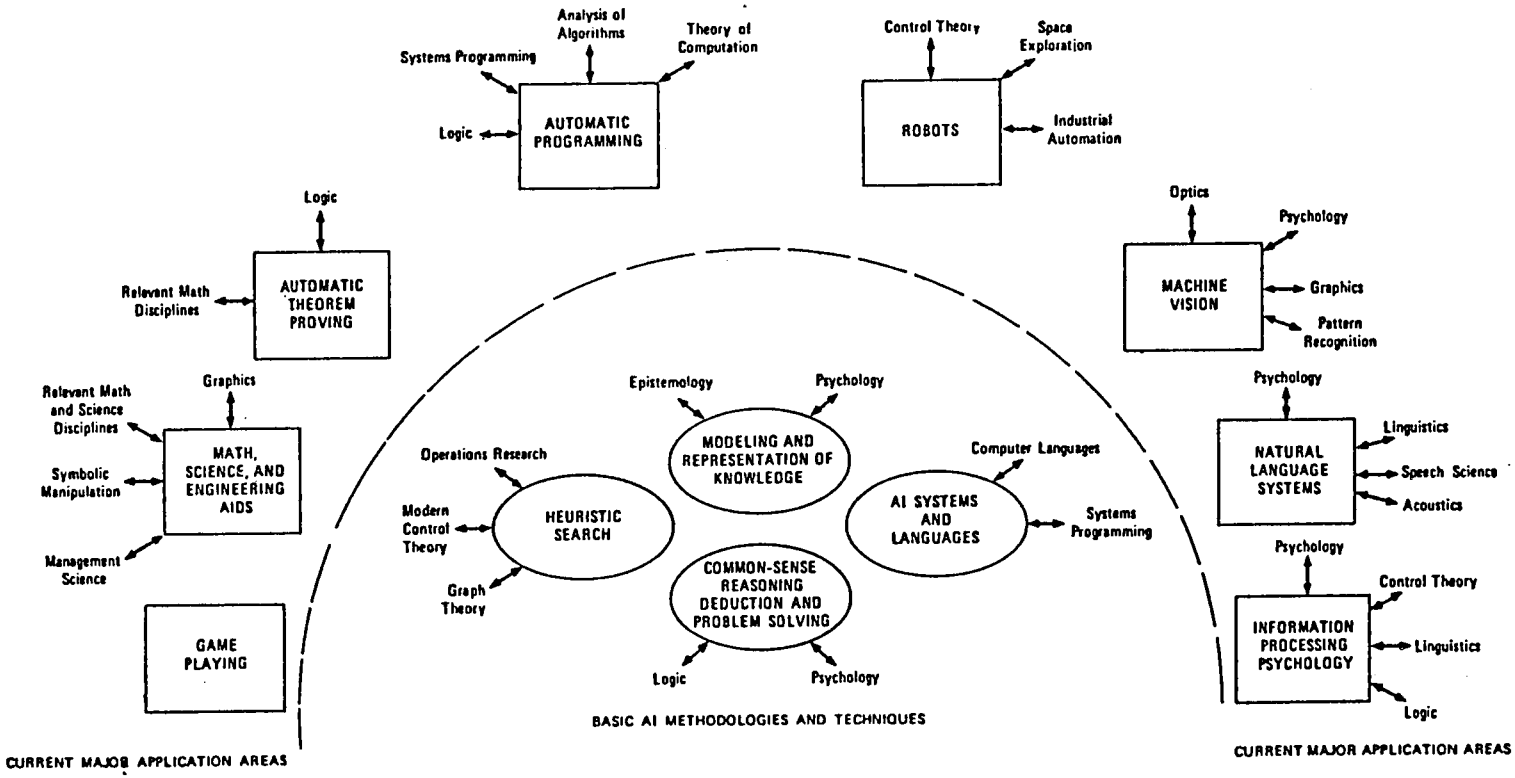


FIGURE 1 MAJOR SUB-PARTS OF AI SHOWING TIES TO OTHER FIELDS

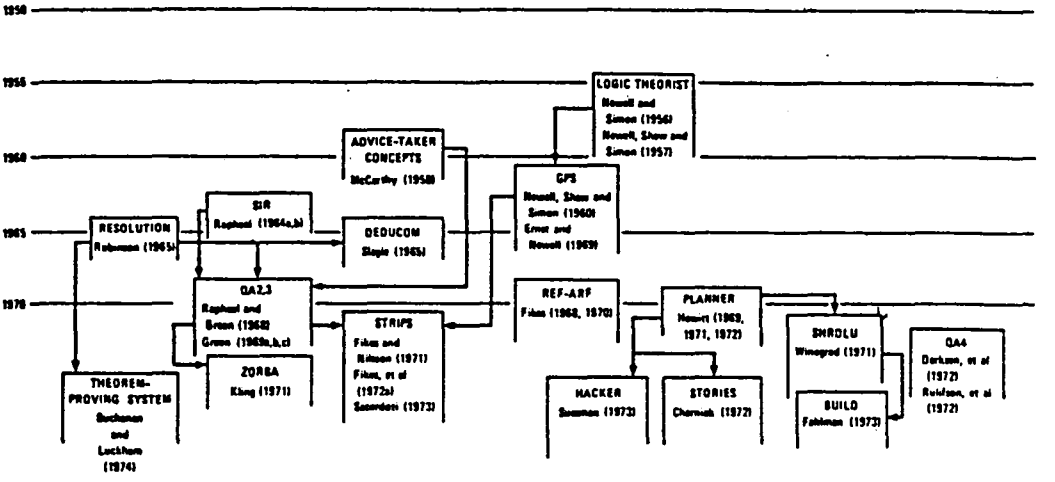


CHART 1: COMMON-SENSE REASONING, DEDUCTION AND PROBLEM SOLVING

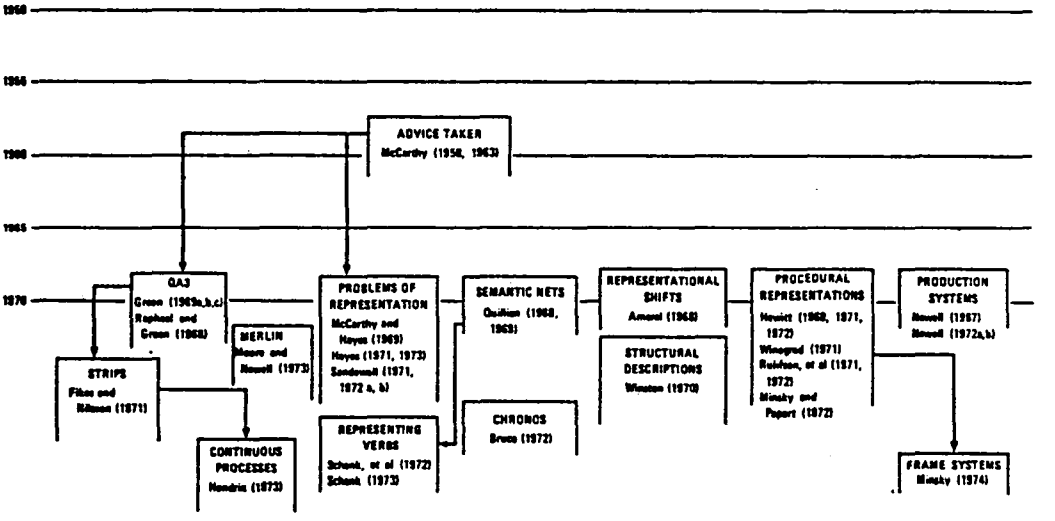


CHART 2: MODELING AND REPRESENTATION OF KNOWLEDGE

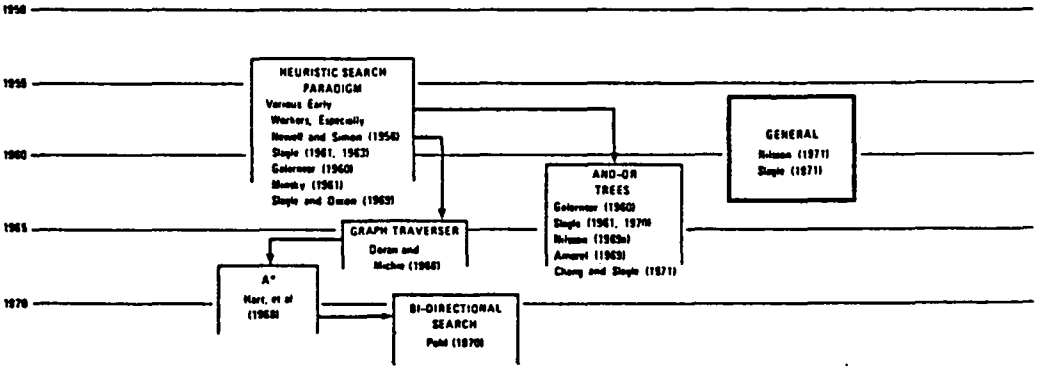


CHART 3: HEURISTIC SEARCH

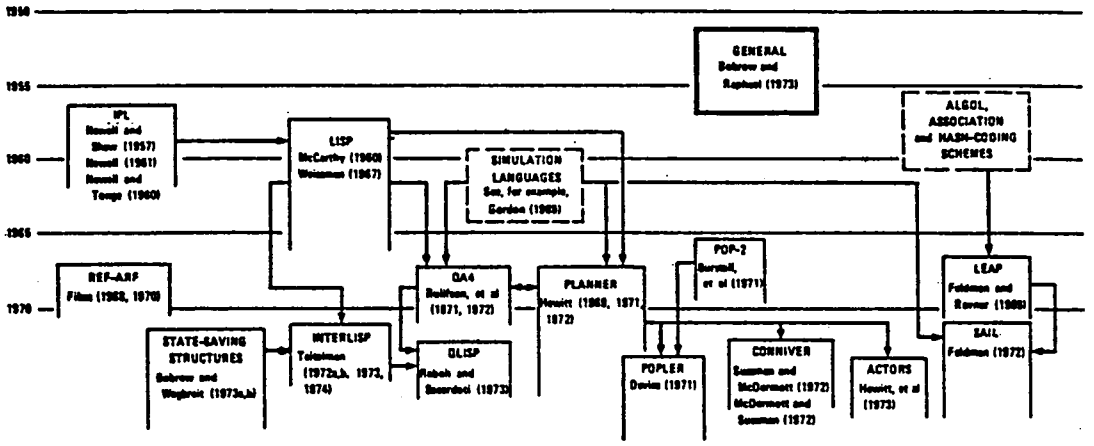


CHART 4: AI SYSTEMS AND LANGUAGES

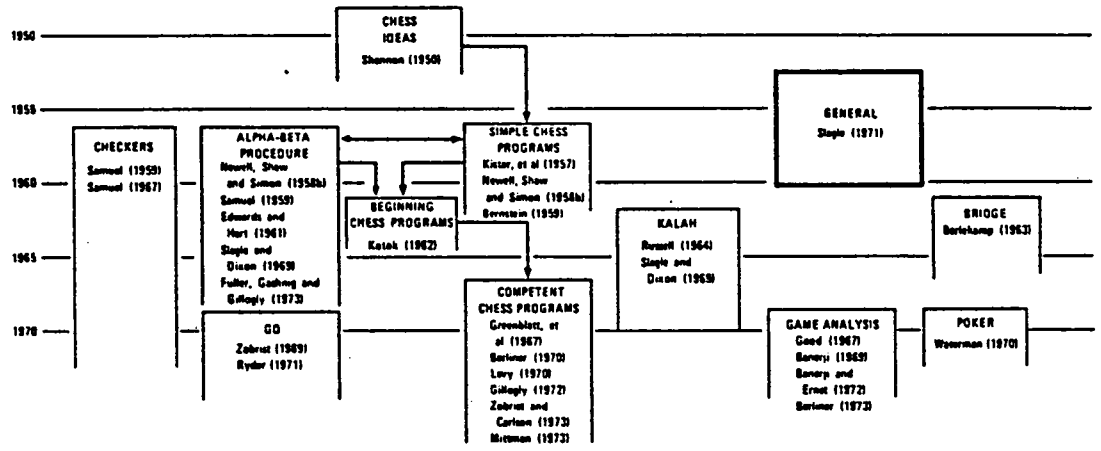


CHART 5: GAME PLAYING

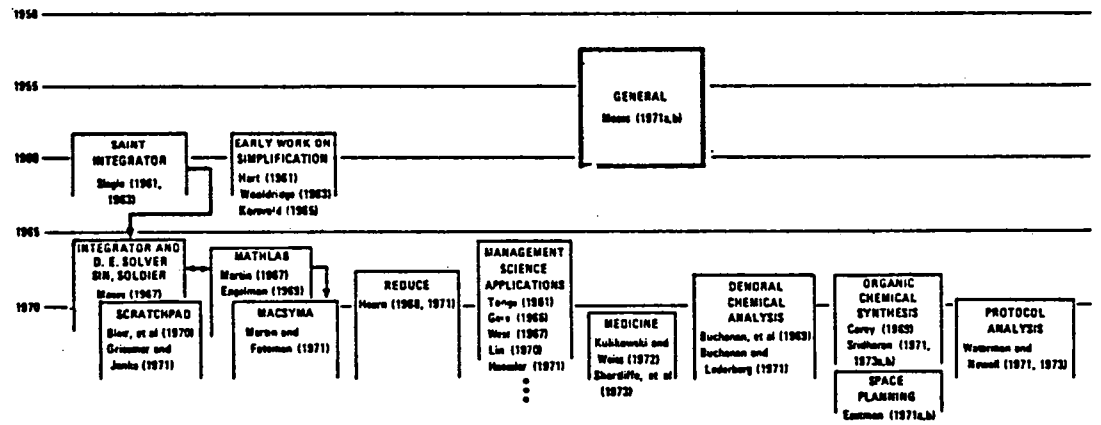


CHART 6: MATH, SCIENCE AND ENGINEERING AIDS

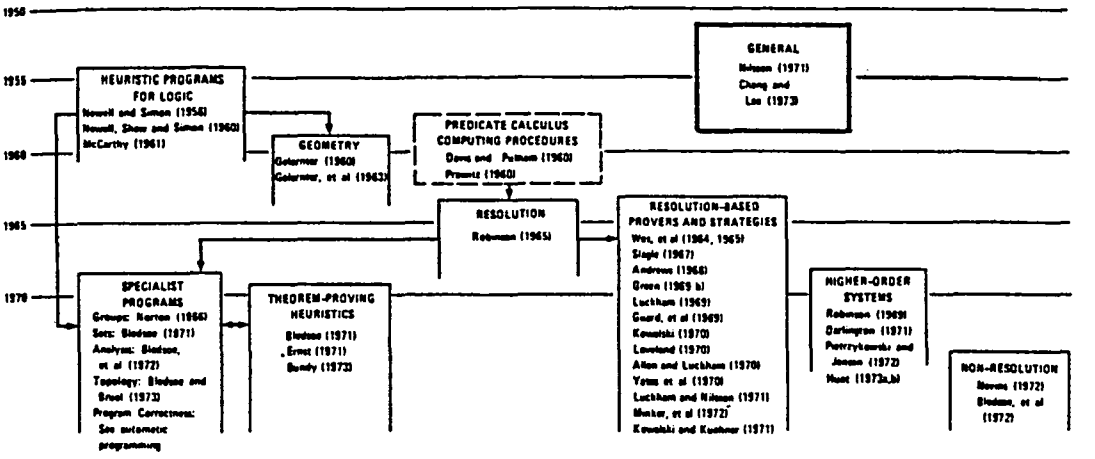


CHART 7: AUTOMATIC THEOREM PROVING

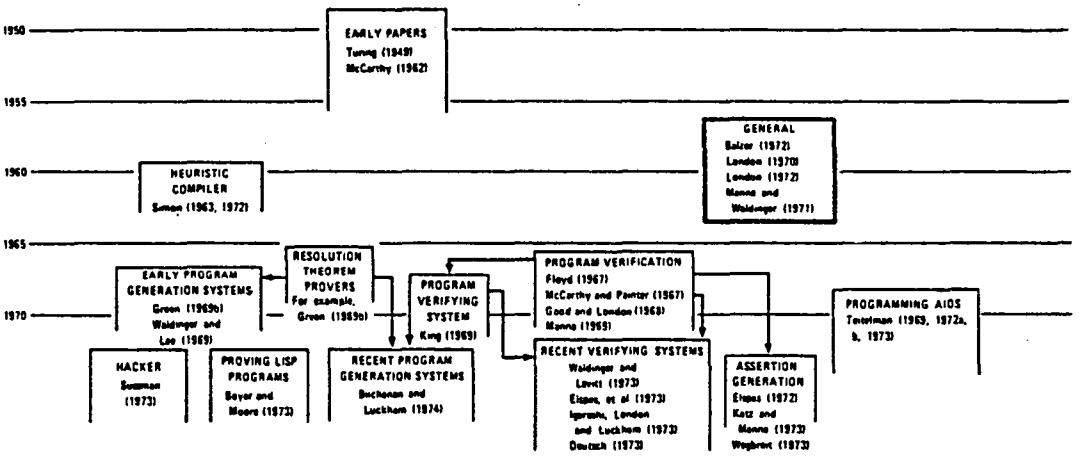


CHART 8: AUTOMATIC PROGRAMMING (Including Verification)

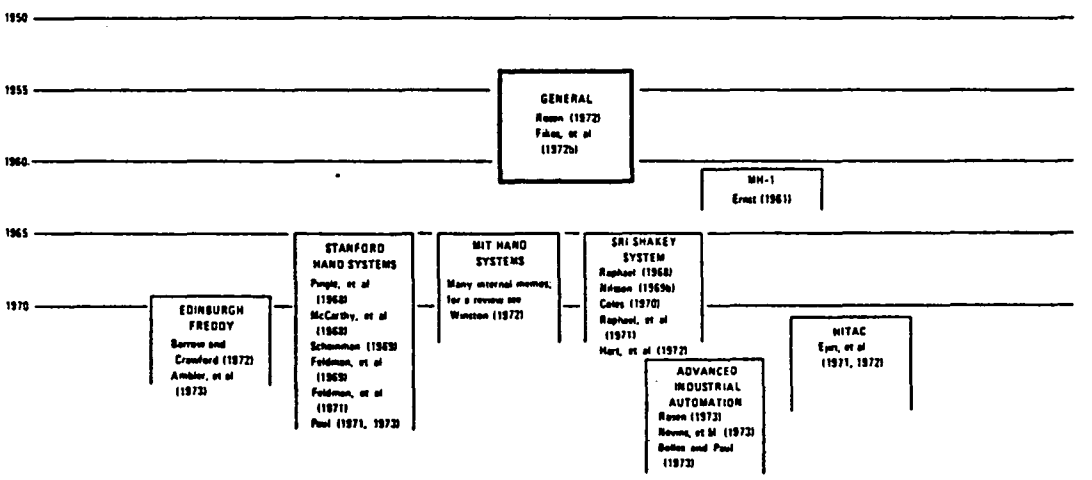


CHART 9: ROBOTS

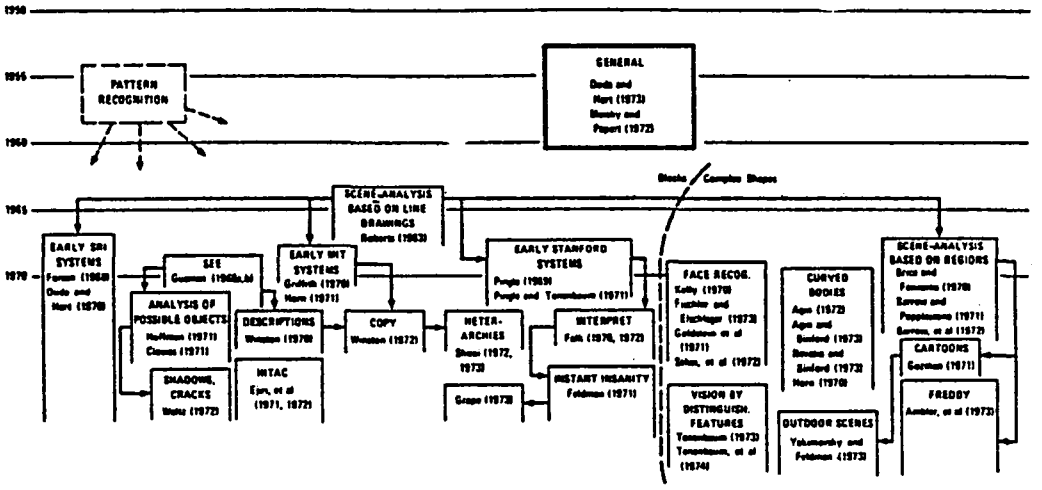


CHART 10: MACHINE VISION

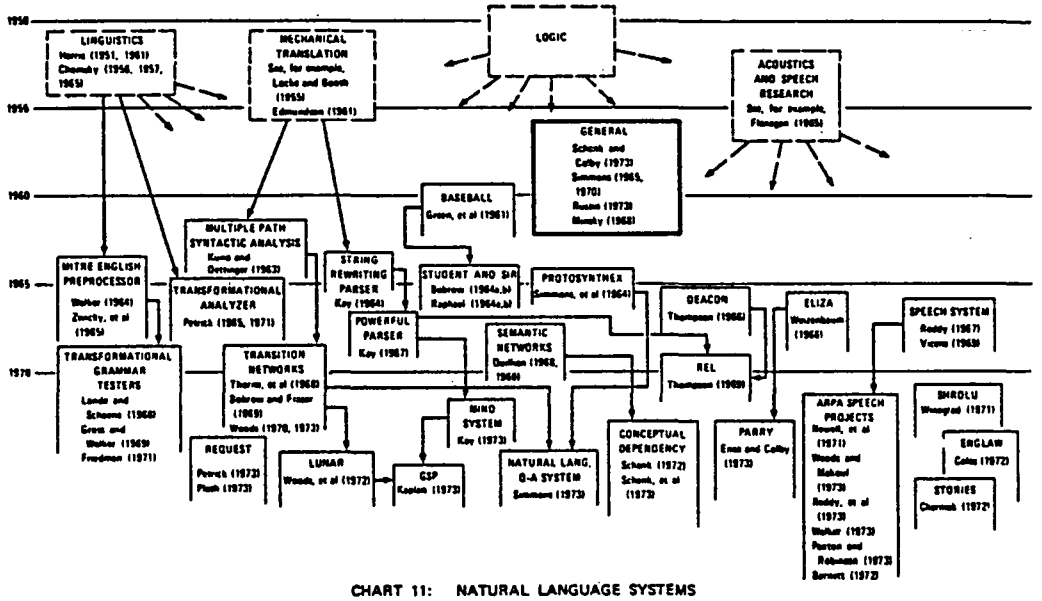


CHART 11: NATURAL LANGUAGE SYSTEMS

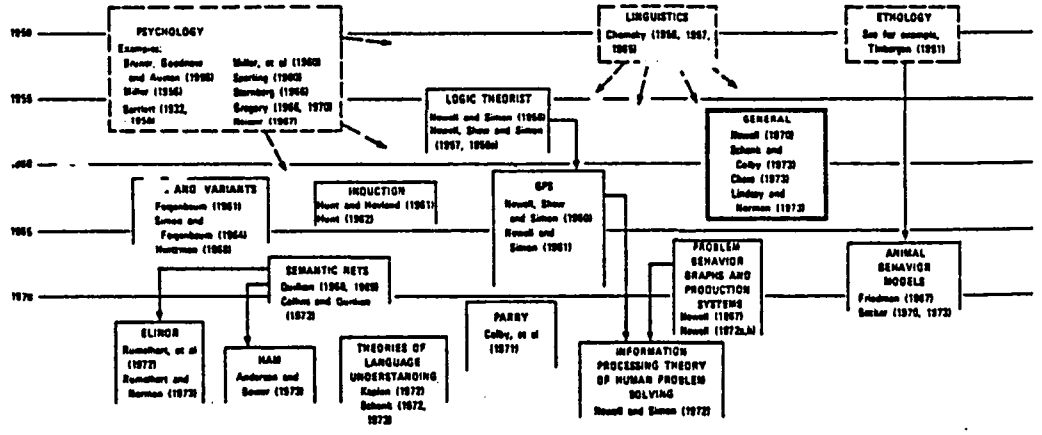


CHART 12: INFORMATION PROCESSING PSYCHOLOGY

that these useful ideas seem so familiar in AI research today testifies to the success of this early work. But the very cleanness of puzzles allowed researchers to avoid facing what has turned out to be the key problem, namely dealing with knowledge, huge amounts of knowledge, diverse, cluttered and interrelated.

Question-answering. As one step toward facing the problem of dealing with knowledge, several researchers concentrated on building inferential question-answering systems. (See, in particular, the references listed under SIR, QA2, and QA3 in Chart 1.) Such systems should be able to store a large number of facts and should be able to respond to reasonable questions whose answers could be deduced from these facts. These systems required mechanisms for logical inference and led AI researchers into a romance with logic in general and with Robinson's resolution principle in particular. (See Chart 7.) This line of research clarified our concepts of applying inference techniques to common-sense knowledge and led to various useful schemes for associative retrieval of stored data. We also learned that for large question-answering systems the question of when to use inference methods was more important than the nature of the inference mechanism itself. Thus, we learned that we would need large amounts of secondary knowledge about how and when to use the primary knowledge of the domain.

Common-sense reasoning. In 1958, McCarthy proposed an ADVICE-TAKER that would be able to accept knowledge and use it to deduce answers to questions and to figure out simple plans for courses of action. One might ask such a system, for example, how to get to Timbuktu (a favorite example of McCarthy's). If the system knew about airline schedules, airports, how to get to airports, and other common (but immensely diverse) knowledge, it might answer thus: (1) go to your travel agent and find out about flights to Timbuktu, (2) using this information, select a flight and make a reservation, (3) drive to the airport at the appropriate time, (4) park your car, and (5) get on the appropriate airplane. Each of these steps, of course, could be expanded in detail.

Problems of this sort are clearly not as clean as puzzles; they demand the use of large amounts of knowledge; yet they have in common with puzzles the feature of planning a course of action to accomplish a goal.

Robotics research (see Chart 9) has probably contributed the most to our knowledge of how to generate plans based on large amounts of common-sense knowledge. Researchers at MIT, using an arm in a domain of simple blocks (called the BLOCKS world) and at SRI using a mobile robot in a domain of corridors and rooms, have developed various reasoning systems that can generate plans of action for a robot. Of these, we might mention in particular STRIPS, SHRDLU, and HACKER (see Chart 1).

There has been a lot of useful internal controversy about how to build reasoning systems and about the best directions for research. For a while, there was hope in some quarters that some universal system (based, for example, like QA3 on Robinson's resolution principle) could be used for all of the tasks we have mentioned so far: puzzle-solving, question-answering, and common-sense reasoning. First attempts to build such universal systems were unsuccessful in the incorporation of the necessary domain-specific knowledge and techniques and, as far as I know, there are at present no serious advocates of a simple universal system.

At the opposite extreme of this controversy, however, are the proponents of what I would call ad hocism. To them, following any systematic approach is anathema. Each task should simply be programmed on its own using whatever tricks might be needed. There is no doubt that this kind of opportunism is healthy for a growing field still in search of its general principles. Still, the following point must be made against rampant ad hocism: One part of developing a science is to discover those concepts that are important. We must try to produce intelligent behavior out of systems limited to various combinations of trial concepts. Our failures tell us whure our present concepts are weak and give us hints about new ones that might be needed. If our trial concepts are always allowed the crutch of ad hocism, we do not learn enough about where the concepts are weak.

Another controversy concerns how much knowledge we ought to give our reasoning programs. At one extreme are researchers who insist that the program should be given only some basic premises from which it must derive any intermediate knowledge it needs to arrive at an answer. At the other (and impossible) extreme, programs would be provided explicitly with answers to all problems. There are some who feel that derivation of answers ultimately will play such a large role in intelligent systems that we may as well concentrate now on derivation techniques. To force derivation, they tend to work with knowledge-impoverished systems.

The consensus just now emerging from this controversy is that, because of combinatoric problems, an intelligent system probably will be able to make only reasonably direct derivations at any stage. Thus, to deal with a large domain, such a system must begin with a large skeletal network of basic knowledge about the domain and knowledge about how to use its knowledge. Any excursion from the known (explicitly represented) knowledge into the unknown (derived) can thus be well-guided (i.e., practical) even though the "volume" of the unknown part itself can be extremely large. It is senseless to insist that, to answer a single question, an intelligent system must repeat the tedious trial and error evolution of a large part of our cultural and scientific knowledge to say nothing of possibly having to repeat much of biological evolution itself. Even the "let's derive all" school would agree. What members of this school and some others did not realize was just how much knowledge would finally be needed by intelligent systems. Given this realization, the only possible course is to build "knowledge-based" programs.*

2.2.2 Modeling and representation of knowledge (Chart 2)

Our ideas about how to represent knowledge have come from several of the applications areas. (Quite

*Minsky (1974) guesses that a knowledge-based system reasoning about visual images (a system such as might be possessed by a typical human) "might need a few millions, but not billions, of structural units, interconnections, pointers."

obviously, every AI program uses some representational scheme. We cite in Chart 2 just a few of the important contributions.) Researchers in machine vision and perception and in natural language understanding were perhaps the first to realize how much knowledge would be needed by high performance programs. These two applications areas have thus probably contributed the most to our repertoire of representational techniques.

The systems mentioned in Chart 2 cover some of the major suggestions. For example:

Green (1969a,b,c): Statements in the first order predicate calculus.
Quillian (1968): Concept nodes in a graph structure linked by various relationships.
Schank et al. (1972): Canonical concept structures having "slots" for case information.
Hewitt (1969,71) and Winograd (1971): Pattern-invoked procedures plus assertions.
Rulifson et al. (1971): Pattern-invoked procedures plus special list structures such as n-tuples, bags and sets with property lists all organized in a discrimination net.
Newell (1967): Sets of productions organized as Markov tables.
Minsky (1974): Hierarchically organized structures called "frame systems." These have "free variables" (analogous to Schank's slots) that can be matched against constants occurring in the data to be analyzed.

For a period there was some controversy over whether knowledge should be represented assertionally or procedurally. (As an extreme case, a spiral, say, can be represented assertionally by a list of the points in the plane through which it passes, or it can be represented procedurally by a program that draws it.) Something of a cult was made of the "procedural embedding" of knowledge, but this controversy seems to be settling down now to an acceptance of the value of a combination of assertional and procedural knowledge.

Another concern, having antecedents in logic, is how to represent certain "modal" concepts involving time, necessity, possibility, and so forth. McCarthy & Hayes (1969) have analyzed some of the difficulties in formalizing these concepts; meanwhile, Hendrix (1973) and Bruce (1972) have developed systems that begin to deal with some of them.

McCarthy and Hayes (1969) also discuss two fundamental problems concerning representation and reasoning. One is called the frame problem, and it concerns certain difficulties of model maintenance. If we have a representation of the world at a certain instant (based on observations and a priori knowledge), how should we represent and use "laws of physics" to update the model so that it represents the world (reasonably accurately) at some future instant? If a robot removes a book from a shelf, can we assume that a door across the room remains open without having to derive this fact or observe it again? There are several ways of dealing with this problem, e.g., Green (1969), Fikes and Nilsson (1971), Sandewall (1972), and Hewitt (1969). These are nicely discussed by Hayes (1973).

Another problem is the qualification problem. If a system uses its representation to "prove," say, that a certain plan will achieve a desired goal (the goal of being at the airport), how are we to deal with certain difficulties arising when new information is received prior to executing the plan. Suppose, for example, someone tells us that our automobile is out of gasoline so that now our plan (that called for driving to the airport) will not work. We had proved that it would, and now new information has rendered the proof invalid even though all of the information on which the original proof was based is still present. Hayes (1973) discusses this violation of the "extension property" and shows the close connection between the qualification problem and the frame problem. System builders [e.g., Hewitt (1969) and Rulifson et al. (1972)] have invented certain constructs that apparently get around these difficulties, although in a way that is somewhat unsatisfactory to logicians.

We are still quite a way, it seems, from having a sound theoretical basis for knowledge representation. It is my view that the necessity of developing large and complex reasoning systems will produce the new concepts out of which the needed theories will be constructed.

2.2.3 Heuristic search (Chart 3)

One of the first results of early AI research was the development of a point of view toward problem-solving sometimes called "the heuristic search paradigm." There are two closely related versions of this paradigm. In one, a "problem" is transformed into the canonical problem of finding a path through a "space" of problem states from the initial state to a goal (i.e., solution) state. In the other, a problem is "reduced" to various subproblems that are also reduced in turn (and so on) until the ultimately resulting subproblems have trivial or known solutions. Each version is merely a slightly different way of thinking about basically the same problem-solving process. In each, the process involves generating alternative paths toward solutions, setting up certain key milestone states (or subproblems), and managing search resources wisely to find acceptable solutions.

The word "heuristic" is used because these techniques emphasize the use of special knowledge from the problem domain that "aids in discovering a solution" by drastically reducing the amount of search that would otherwise have to be employed. Often this knowledge takes the form of "rules-of-thumb" that help to limit or direct the search. Sometimes there are constraining relations that can be employed to limit the search needed. [A good example of the use of constraints is the work of Waltz (1972).]

I have already referred to some of the heuristic search paradigm ideas (subgoals, reasoning backwards, and so on) as being basic to common-sense reasoning, deduction, and problem solving (Chart 1). Here (in Chart 3), we want to cite mainly those aspects of heuristic search dealing with the search process itself. Once a problem is represented as a search problem, how can a solution be found efficiently?

The searching occurs in one of two graph structures, ordinary graphs (or trees), and AND-OR graphs (or trees), depending on whether the problem is viewed as one of finding a path to a goal state or one of reducing problems to subproblems, respectively. The search techniques that have been developed (by workers in AI, control theory, and operations

research) are now commonly used in many AI programs and in many of their applications. Most of these techniques make use of heuristically-based evaluation functions that rank-order the unexplored nodes in the graph and thus indicate where search can most efficiently proceed. Furthermore, there are some theorems [Hart et al. (1968)] stating conditions under which these search methods are guaranteed to find optimal paths. The problem of efficiently searching a graph has essentially been solved and thus no longer occupies AI researchers. This one core area, at least, seems to be well under control.

2.2.4 AI systems and languages (Chart 4)

The programming languages developed and used by AI researchers are included among the core topics because they embody the most useful of the core ideas already discussed. Early AI researchers saw the need for programs that could store, access, and manipulate lists of symbolic information. The means for achieving these and other operations were built into various list processing languages, primarily IPL-V and LISP.

After some years of research using these languages, it became apparent that AI systems had a common, recurring need for operations such as search, expression-retrieval, and pattern-matching. The next step was to build these operations into the languages themselves. Thus, in the late 1960s, another generation of AI languages emerged, languages such as QA4 and PLANNER.

Edward Feigenbaum once characterized progress in AI research as progress along the "what-to-how" spectrum of computer languages. At the "how" end of this spectrum are the machine languages used by programmers who must give the most detailed instructions to the computer. As one progresses toward the "what" end, the programmer leaves more and more of the details of how operations are to be carried out to the language and can be more and more concerned only with what is to be done. AI languages are now moderately far along toward the "what" end, and the proper goal of AI research (according to this view) is to create languages even closer to the "what" end. It may well be that, ultimately, the field of AI will in large part be concerned with the development of superpowerful computing languages. In this light, the best way to measure AI progress is to look at the AI languages.

We do not have space here to trace the development of AI languages nor to describe the special features that they make available to AI researchers. Fortunately, there is an excellent tutorial paper by Bobrow and Raphael (1973) that gives a very clear account of the new languages.

Currently, a large part of AI research is being conducted by experimenting with systems written in the new languages. The languages provide especially powerful mechanisms for representing the extensive knowledge needed by present programs. Furthermore, this knowledge can now be easily added incrementally as the program evolves under the tutelage of human experts in the domain. Winograd's (1971) natural language understanding system and Waldinger and Levitt's (1974) system for proving assertions about programs are good examples of how the power of these languages is being used.

It would not be unreasonable to expect that current and future experimentation will lead to the crystallization of additional concepts [such as, perhaps, Minsky's (1974) Frame Systems] that will be incorporated in a new round of AI languages, possibly in the late 1970s.

2.3 First-level applications topics

2.3.1 Game playing (Chart 5)

Programs have been written that can play several games that humans find difficult. As the most famous example, we might mention the chess playing program, MAC-HACK, of Greenblatt et al. (1967). A version of this program achieved a United States Chess Federation rating of 1720 in one tournament. Samuel's programs for checkers have beaten experts in the game. Several other programs are mentioned in the chart.

Levy (1970) described a program written by Atkins, Slate, and Gorland at Northwestern University and said that he thought it was stronger than Greenblatt's. He estimated its rating at about 1750, which would make it, he claims, the 500th best player in Britain.

Computer chess tournaments are now held routinely. Results of these and other news about computer chess have been rather extensively reported in the SIGART Newsletter since 1972.

Most game playing programs still use rather straightforward tree-searching ideas and are weak in their use of high-level strategic concepts. It is generally agreed that advances in the use of strategy and in end-game play are necessary before chess programs can become substantially better, and they must become substantially better before they can beat human champions. (World Champion Bobby Fischer is rated at about 2810.) Levy (1970) is rather pessimistic about the rate of future progress in chess and has made a £750 bet with Professors McCarthy, Papert, and Michie that a program cannot beat him in a match by August 1978. (Levy's rating in 1970 was 2380.)

2.3.2 Math, science, and engineering aids (Chart 6)

The chart lists just a few examples of AI techniques that have been applied in systems that help human professionals. The early AI work on symbolic integration, together with the work on algebraic simplification, contributed to a number of systems for symbolic mathematical computations. Moses (1971b) presents a good review. Systems presently exist that can solve symbolically an equation like $y^{2x} - 3y^x + 2 = 0$ (for x), and that can integrate symbolically an expression like $\int (x + e^x)^2 \, dx$. Such systems are quite usefully employed in physics research, for example, in which expressions arise having hundreds of terms.

Another quite successful application is the DENDRAL program that hypothesizes chemical structures from a combination of mass spectrogram and nuclear magnetic resonance data. The system is presented with this data from a sample of a known chemical compound (that is, its chemical formula is known). It uses several levels of knowledge about chemical structures and how they break up in mass spectroscopy to infer the structure of the compound. It can deal with

a large number of organic compounds including complex amines and estrogenic steroids. Its performance on the steroids often exceeds the best human performance.

The DENDRAL project typifies a style of AI system building that has been quite successfully applied to chemistry and some other domains. This design style involves intensive interaction between AI scientists and applications area scientists. The latter are queried in the minutest detail to extract from them rules and other knowledge that are operationally useful in the domain. These are then coded into the system by the AI scientists and tests are run to judge their effectiveness. The process is long and involves several iterations. The applications scientists are often confronted with apparent contradictions between how they say they make decisions and how they actually make decisions. Few of them have any really global or completely accurate theory of how they apply their knowledge. Furthermore, this knowledge is often informal and heuristic. As a result, the emerging system is a collection of "mini-theories" and special rules of only local effectiveness. To use this design strategy, the system must be one that can deal with many, and sometimes conflicting, mini-theories. It must also be a system to which new knowledge can gradually be added and old knowledge modified.

After several months or years of this sort of gradual shaping of the system, it comes to simulate the performance of the human experts whose knowledge it has gained. This general strategy is beginning to be employed extensively in AI applications. [For example, see also Shortliffe et al. (1973).]

2.3.3 Automatic theorem proving (Chart 7)

There are three major themes evident in attempts to get computer programs to prove theorems in mathematics and logic. First, early work by AI researchers produced heuristic programs that could prove simple theorems in propositional logic and high-school level theorems in plane geometry. These programs used (but mainly helped to refine) concepts like reasoning backwards, means-ends analysis, use of subgoals, and the use of a model to eliminate futile search paths. The fact that logicians had already developed powerful procedures that effectively eliminated propositional logic as a domain requiring heuristic problem-solving techniques does not detract from the value of this early work.

Logicians were also developing techniques for proving theorems in the first order predicate calculus. J. A. Robinson (1965) synthesized some of this work into a procedure for using a single rule of inference, resolution, that could easily be mechanized in computer programs. Building resolution-based provers quickly became a second theme in automatic theorem proving, while other approaches languished. Resolution had a great influence on other application areas as well (Charts 1 and 8). Performance of the resolution systems reached impressive, if not superhuman, levels. Programs were written that could prove reasonably complex, sometimes novel, theorems in certain domains of mathematics. The best performance, however, was achieved by man-machine systems in which a skilled human provided strategic guidance leaving the system to verify lemmas and to fill in short chains of deduction. [See especially Guard et al. (1969) and Allen and Luckham (1970). The latter system has been used to obtain proofs of new mathematical results announced without proof in the Notices of the American Mathematical Society.]

Various strategies were developed to improve the efficiency of the resolution provers. These strategies were mainly based on the form or syntax of the expressions to be proved and not on any special knowledge or semantics of the domain. In automatic theorem proving, just as in other applications areas, semantic knowledge was needed to improve performance beyond the plateau reached by the late 1960s.

The work of Bledsoe and his students is typical of the third and latest theme in automatic theorem proving. Although they emphasize the importance of man-machine systems, their programs themselves have become knowledge-based specialists in certain mathematical domains. The use of semantic knowledge in theorem-proving systems has also renewed interest in heuristics for subgoaling, and so forth. The programs of this group are capable of proving some rather impressive theorems, and it can be expected that the present man-machine systems will produce ever more competent and more completely automatic offspring.

2.3.4 Automatic programming (Chart 8)

Work in automatic programming has two closely interrelated goals. One is to be able to prove that a given program acts in a given way; the other is to synthesize a program that (provably) will act in a given way. The first might be called program verification and the second program generation. Work on one goal usually contributes to progress toward the other; hence, we combine them in our discussion.

Most of the work on program verification is based on a technique proposed by Floyd (1967). [See also Turing (1949).] This technique involves associating assertions with various points in the flow chart of a program and then proving these assertions. Originally, the assertions had to be provided by a human, but some recent work has been devoted to generating the assertions automatically. Once proposed, one can attempt to have the assertions proved either by a human or by a machine. The latter course involves a close link between this field and that of automatic theorem proving.

A recent system developed at the Stanford Research Institute [Elspas et al. (1973)] is typical of one in which the assertions are both produced [Elspas (1972)] and proved [Waldinger and Levitt (1973)] automatically. This system has been used to verify several programs including a real-number division algorithm and some sort programs. It has also proved theorems about a pattern matcher and a version of Robinson's (1965) unification algorithm. It is a good example of a modern AI program in that it makes effective use of a large amount of domain-specific knowledge.

The closely related work on program generation has succeeded in producing some simple programs. Typical of this work is the system of Buchanan and Luckham (1974). Broadly viewed, the problem of constructing a computer program includes the problem of constructing a plan, say, for a robot, and thus there are close links between work in automatic programming, robotics, and common-sense reasoning and deduction.

Sussman's (1973) HACKER is another system that writes simple programs for a limited domain (the BLOCKS world). Sussman's goal for HACKER is for it to simulate his own programming style. An important feature of HACKER is its strategy of attempting first to

write a simple "let's-hope-that-this-will-do" program, and then debugging it until it does succeed at its task. To employ this strategy, HACKER uses a great deal of knowledge about likely classes of program bugs and how to fix them.

Again, some of the most successful work has been in connection with man-machine systems. We include in this category certain aids to human programmers such as those found in the INTERLISP system [Teitelman (1972a, b, 1973)]. In fact, any techniques that help make the production of programs more efficient might be called part of automatic programming. Balzer (1972) provides a good summary of this broad view of the field.

2.3.5 Robots (Chart 9)

Every now and then, man gathers up whatever technology happens to be around and attempts to build robots. During the late 1960s, research on robots provided a central focus for integrating much of the AI technology. To build an intelligent robot is to build a model of man. Such a robot should have general reasoning ability, locomotive and manipulative skills, perceptual (especially visual) abilities, and facility with natural language. Thus, robot research is closely linked with several other applications areas. In fact, most of the research on machine vision (Chart 10) was, and is, being performed in connection with robot projects.

Our problem-solving and representational techniques are probably already adequate to allow useful general purpose robot applications; however, such robots would be perceptually impoverished until we develop much more powerful visual abilities. Robotics is a particularly good domain in which to pursue the necessary vision research.

The robot research of the late 1960s produced systems capable of forming and then intelligently executing plans of action based on an internal model of the world. The Edinburgh, Stanford, HITAC, and MIT systems consisted of manipulator arms and TV cameras or other visual input devices. These became capable of building structures out of simple blocks. In one case (Stanford), the system could assemble an automobile water pump. The Stanford Research Institute system consisted of a mobile cart and TV camera (but no arm). It could form and execute plans for navigating through a simple environment of rooms, doorways, and large blocks, and its visual system could recognize and locate doorways, floor-wall boundaries, and the large blocks. The system had sophisticated techniques to allow it to recover from errors and unforeseen circumstances, and it could store (learn) generalized versions of the plans it produced for future use.

Since practical applications of general purpose robot systems seem more remote than they do in other applications areas, the increasingly pragmatic research climate of the early 1970s has seen a lessening of activity in general robotics research. In the meantime, various projects with the practical goal of advancing industrial automation have begun to apply some of the already-developed manipulative and visual skills to factory assembly and inspection problems. It seems reasonable to predict that man's historic fascination with robots, coupled with a new round of advances in vision and reasoning abilities, will lead to a resurgence of interest in general robot systems, perhaps during the late 1970s.

2.3.6 Machine vision (Chart 10)

The ability to interpret visual images of the world is adequate enough even in some insects to guide many complex behavior patterns. Yet the analysis of everyday visual scenes by machine still remains a largely unconquered challenge to AI researchers. Early work concentrated almost exclusively on designing systems that could classify two-dimensional images into a small number of categories—alphanumeric character recognition, for example. In fact, much of the AI work during the 1950s was concerned with pattern recognition. Researchers, such as Frank Rosenblatt and Oliver Selfridge, were influential in shaping this early period. Pattern classification (or recognition) continues as a separate active research interest, but since about 1965, AI interest in vision has centered on the more difficult problem of interpreting and describing complex three-dimensional scenes. Both aspects, classification and description, are thoroughly and clearly treated in an excellent textbook by Duda and Hart (1973).

Much of the scene analysis work can be traced to Robert's (1963) influential thesis. It established a trend of analyzing scenes composed of prismatic solids (the so-called "blocks world"). Working with these (sometimes complex) scenes composed of simple objects helped to establish a wide range of techniques for converting raw video images into symbolic descriptions based on concepts such as lines, regions, and simple shapes. The MIT "COPY" system, for example, can use a visual input device to look at a scene consisting of a structure of blocks. The system can analyze the scene to form a representation of how the blocks are arranged. This representation can then later be used (with the robot arm system) to reproduce this exact block structure from disarranged blocks.

Some successful excursions outside the blocks world have been made. (See the entries to the right of the dashed line in Chart 10). Indeed, many researchers contend that continuing to work with blocks has actually hindered research progress in machine vision because it has allowed workers to avoid facing certain key problems associated with domain semantics, distinguishing features of complex objects, and new representational schemes. In any case, working with more complex scenes is now well established. The spirit of recent work is well described in a note by Tenenbaum (1973). Again, knowledge about the domain is crucial to scene interpretation!

2.3.7 Natural language systems (Chart 11)

Processing written and spoken forms of natural languages has been the subject of so much research that the task of surveying progress in this field alone should humble anyone. How, then, are we to say anything very helpful in just a few sentences. The interested reader should see the GENERAL volumes cited in Chart 11.

Perhaps a description of the power of some representative systems will suffice. Winograd (1971)

implemented a system that carried on "blocks-world" written dialogs like the following:

Man (typing into a terminal): Which cube is sitting on the table?
System (generating text on a display): The large green one which supports the red pyramid.
M: Is there a large block behind a pyramid?
S: Yes, three of them: A large red one, a large green cube, and the blue one.
M: Put a small one onto the green cube which supports a pyramid.
S: OK
M: Put the littlest pyramid on top of it.
S: OK

The system demonstrates its understanding of the last two commands by having a simulated robot arm carry out appropriate actions in a simulated blocks world.

The work of Schank (1972) typifies a rather successful trend in natural language understanding. Many of the recent systems, in one way or another, attempt to match a section of input text or utterance against semantically likely stored structures (that are more or less complex.) These structures are themselves schemas or scenario families having variables that are bound to constants in the input during matching. The instantiated scenarios serve as a sort of deep structure that represent the meaning of the utterance. [See also Minsky (1974).]

The goals of a coordinated scientific effort to produce systems to understand limited utterances of continuous speech are clearly outlined in a plan by Newell et al. (1971). If the goals are met, by 1976 a prototype system should be able (in the context of a limited domain of discourse) to understand (in a few times real time) an American (whose dialect is not extremely regional) speaking (in a "natural" manner) ordinary (although perhaps somewhat simple) English sentences constructed from a 1000-word vocabulary. These projects bring together workers in acoustics and speech research as well as in AI. The projects seem to be more or less on schedule and will probably achieve creditable performance by 1976. (In the spirit of the vagueness of the phrase "a few times real time," the projects ought to achieve the 1976 goals at least sometime in the late 1970s.)

In my opinion, the work in natural language understanding is extremely important both for its obvious applications and for its future potential contributions to the core topics of AI. It is the prime example of a field in which reasonable performance could not be achieved by knowledge-impoverished systems. We now know that understanders need large amounts of knowledge; the challenge is to attempt to build some really large systems that have the adequate knowledge and to learn, by our mistakes, the organizational principles needed to keep these large systems from becoming unwieldy.

2.3.8 Information processing psychology (Chart 12)

Computer science in general and AI in particular have had a tremendous impact on psychology. They provide the concepts and the very vocabulary out of which to construct the most useful theories of human behavior. In my opinion the reason that, say, prior to 1955, there were, in fact, no adequate theories of human behavior, perception, and cognition is because the concepts out of which to construct these theories had not yet been formulated. Before we have the concepts (and they are now gradually accumulating) it is as impossible to understand human thought as it was impossible to understand navigation, say, before we had the concept of sonar. Man understands the world by constructing models, and his models are often based on concepts drawn from his technological inventions. We may not understand man immediately after building the first robot, but we certainly won't understand him before! (We note in passing that knowledge about the structure and function of the neuron—or any other basic component of the brain—is irrelevant to the kind of understanding of intelligence that we are seeking. So long as these components can perform some very simple logical operations, then it doesn't really matter whether they are neurons, relays, vacuum-tubes, transistors, or whatever.)

An excellent short account of the relationship between AI and psychology has been written by Newell (1970). While he, perhaps prudently, adopts a somewhat less extreme position than mine about the dependence of psychology on AI, he nevertheless shows how thoroughly information processing ideas have penetrated psychological theory.

Most of the information-processing-based psychology to date has been devoted to explaining either memory (e.g., EPAM and HAM in Chart 12), perception [e.g., Sternberg (1966)], or problem solving [e.g., Newell and Simon (1972)]. Probably the most complete attempt at understanding human problem-solving ability is the last-mentioned work of Newell and Simon. This volume proposes an information processing theory of problem-solving based on the results of many years of research in psychology and AI.

Animal behavior, while long the special interest of experimental psychologists, has had little information-processing-based theoretical attention. Some models inspired by ethologists have been proposed by Friedman (1967). I think that the production system model advanced to explain certain human problem solving behavior by Newell (1967) and colleagues might be a starting point for an extensive theory of animal behavior. Newell, himself, notes that these production systems can be viewed as generalizations of stimulus-response systems. [Incidentally, the entire repertoire of what was called "intermediate-level actions" of the Stanford Research Institute robot system (Raphael et al. 1971) was independently programmed in almost exactly this production formalism. Production systems have been used in other AI programs as well.] Newell and Simon (1972, p. 803) have also stated that they "have a strong premonition that the actual organization of human problem solving programs closely resembles the production system organization ...." It would seem profitable then to attempt to trace the evolutionary development of this hypothesized production system organization down through some of the higher animals at least.

3. CONCLUSIONS

In summary, we see that the AI campaign is being waged on several different fronts, and that the victories, as well as the setbacks, contribute to a growing common core of ideas that aspires to be a science of Intelligence. Against this background,

it is worth mentioning some of the popular criticisms of AI:

(1) AI hasn't really done anything yet. There are a few "toy" programs that play middling chess and solve simple puzzles like "missionaries and cannibals," but the actual accomplishments of AI measured against its promises are disappointing. [See, for example, Dreyfus (1965, 1972).] [My comment about this kind of criticism is that its authors haven't really looked at AI research past about 1960.]
(2) Not only has AI not achieved anything, but its goals are actually impossible. Thus, AI is something like alchemy. It is impossible in principle to program into computers such necessities of intelligence as "fringe consciousness" and "perspicuous grouping." [Again, see Dreyfus (1965, 1972).] [This kind of criticism is actually rather brave in view of the fate of many previous impossibility predictions. This attack simply looks like a poor bet to me.]
(3) The subject matter of AI, namely intelligence, is too broad. It's like claiming science is a field. [This criticism may have some merit.]
(4) Everything happening in AI could just as well happen in other parts of computer science, control engineering, and psychology. There is really no need for this AI "bridge" between already established disciplines. [See Lighthill (1973).] [This kind of criticism caused quite a stir in Great Britain recently. I think I have shown that the so-called bridge has quite a bit of internal structure and is contributing a heavy traffic of ideas into its termini.]
(5) AI is impossible because it is attempting to reduce (to understanding) something fundamentally "irreducible." Furthermore, this very attempt is profane; there are certain awesome mysteries in life that best remain mysterious. [See Roszak (1972).] [My prejudice about this view is that, at best, it is, of course, nonsense. A blind refusal even to attempt to understand is patently dangerous. By all means, let us not foreclose a "rhapsodic understanding" of these mysteries, but let us also really understand them.]
(6) AI is too dangerous, so it probably ought to be abandoned--or at least severely limited. [See Weizenbaum (1972).] [My view is that the potential danger of AI, along with all other dangers that man presents to himself, will survive at least until we have a science that really understands human emotions. Understanding these emotions, no less than understanding intelligence and perception, will be an ultimate consequence of AI research. Not to understand them is to be at their mercy forever, anyway.]

The one criticism having any weight at all, I think, is that AI may be too broad and diverse to remain a cohesive field. So far, it has stayed together reasonably well. Whether it begins to fractionate into separate exotic applications areas of computer science depends largely, I think, on whether these applications continue to contribute core ideas of great generality.

What is the status of these core ideas today? There are two extreme views. I have heard John McCarthy say (perhaps only provocatively to students) that really intelligent programs are a long way off and that when we finally achieve them they will be based on ideas that aren't around yet. Their builders will look back at AI in 1974 as being a period of pre-history of the field.

On the other hand, what if we already have most of the ideas that we are going to get, ideas like millions of coordinated mini-theories, procedural embedding of knowledge, associative retrieval, and scenario frames. Suppose that we have now only to devote the large effort required to build really huge intelligent systems based on these ideas. To my knowledge, no one advocates this alternative view, but consider this: Whatever the nature of an intelligent system, it will be exceedingly complex. Its performance will derive in large part from its complexity. We will not be sure that AI is ready to build a large, intelligent system until after we have done so. The elegance of the basic ideas and the new and powerful languages alone will not be sufficient indication of our maturity. At some time, we will have to put together exceedingly complex systems. The time at which it is appropriate to try will always be a guess.

My guess is that we still have a good deal of work to do on the problem of how to obtain, represent, coordinate, and use the extensive knowledge we now know is required. But these ideas will not come to those who merely think about the problem. They will come to those who both think and experiment with much larger systems than we have built so far.

Another problem, of a more practical type, concerns knowledge acquisition. Today, the knowledge in a program must be put in "by hand" by the programmer although there are beginning attempts at getting programs to acquire knowledge through on-line interaction with skilled humans. To build really large, knowledgeable systems, we will have to "educate" existing programs rather than attempt the almost impossible feat of giving birth to already competent ones. [Some researchers (e.g., Papert, 1972) expect that at least some of the principles we discover for educating programs will have an impact, perhaps revolutionary, on how we educate people.]

In this connection, we have already mentioned that several successful AI systems use a combination of man and machine to achieve high performance levels. I expect this research strategy to continue and to provide the setting in which the human expert(s) can gradually transfer skills to the machine. [Woods and Makhoul (1973) consciously apply a strategy such as this and call it "incremental simulation."]

I have not yet mentioned in this paper the subject of learning. It is because I have come to agree with John McCarthy that we cannot have a program learn a fact before we know how to tell it that fact and before the program knows how to use that fact. We have been busy with telling and using facts. Learning them is still in the future, although some isolated successes have, in fact, occurred. [See especially, Samuel (1959, 1967), Winston (1970), Fikes et al. (1972a), and Sussman (1973).]

Continuing our discussion of the likely future of AI, we note that the increasingly pragmatic attitude of those who have been sponsoring AI research will have a great effect on the course of this research. There may even be a temporary reduction of effort by AI researchers in the core topics and the first-level applications areas in favor of increased support of engineers and scientists building second-level applications. The results of these second-level efforts may, in fact, be rather spectacular. I have in mind such things as automated factories, automatic robots

C-3

for factories and warehouses, medical diagnosis systems, systems that will automate a large amount of office work, legal aids, teaching aids, interactive software production systems, and so on. [Firschein et al. (1973), make some predictions about when these and other intelligent systems may come.]

The short range result of this increased pragmatism may tend to fractionate the field. In the long run, though, if there really are many more core ideas to be discovered, these technological efforts will stimulate their discovery, provided that a sufficient level of basic investigation continues.

In closing, I have one final prediction. As AI successes grow, so will the criticisms of AI, especially from those who are certain that intelligence cannot be mechanized. These critics, having been forced out of various mystical trenches in the past, will be especially vigorous in their defense of what little ground remains to them. The ensuing debates will have the crucially important side effect of getting us all to consider how we want to use and control our new intellectual powers. I hope that society assesses these powers accurately and is not lulled by certain otherwise well-meaning humanists into believing that Artificial Intelligence is not real.

ACKNOWLEDGMENTS

I am grateful for the comments and criticisms of the following people: Woodrow Bledsoe, Stephen Coles, Edward Feigenbaum, Jerome Feldman, Richard Fikes, Cordell Green, Peter Hart, Michael Kassler, John McCarthy, Allen Newell, Charles Rosen, Earl Sacerdoti, Jay Tenenbaum, Richard Waldinger, and Donald Walker.

BIBLIOGRAPHY

Each entry has a code symbol or symbols associated with one or more of the twelve subheadings of AI that we have discussed in the paper. These symbols are: DED = Common-Sense Reasoning, Deduction, and Problem Solving; REP = Modeling and Representation of Knowledge; SEARCH = Heuristic Search; SYS = AI Systems and Languages; GAME = Game Playing; AIDS = Math, Science, and Engineering Aids; TP = Automatic Theorem Proving; PROG = Automatic Programming; ROB = Robots; VIS = Machine Vision; LANG = Natural Language Systems; PSYC = Information Processing Psychology. A prefix "-G" after a symbol means that the reference contains a general discussion or survey.

The code symbol "GEN" identifies the reference as being general to the whole field of AI. These general references are:

Collins and Michie (1968)
Dale and Michie (1968)
Dreyfus (1965, 1972)
Feigenbaum (1963, 1969)
Feigenbaum and Feldman (1963)
Firschein, et al. (1973)
Hunt (1974)
Jackson (1974)
Lighthill (1973)
Meltzer and Michie (1969, 1970, 1971, 1972)
Michie (1968)
Minsky (1961, 1965, 1968)
Newell (1973)
Papert (1968)
Papert (1972)
Roszak (1972)
Simon (1969)
Slagle (1971)
Solomonoff (1966)
Turing (1950)
Weizenbaum (1972)

REFERENCES

Agin, G. (1972), "Representation and Description of Curved Objects," Ph.D. thesis, Elec. Eng. Dept., Stanford Univ., Stanford, CA, July 1972. Available as Stanford Artificial Intelligence Laboratory Memo AIM-173, Oct. 1972. (VIS)

Agin, Gerald J. and Binford, Thomas O. (1973), "Computer Description of Curved Objects," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (VIS)

Allen, John and Luckham, David (1970), "An Interactive Theorem-Proving Program," Machine Intelligence, Vol. 5, 321-336, 1970. (TP)

Amarel, S. (1968), "On Representations of Problems of Reasoning About Actions," Machine Intelligence, Vol. 3, D. Michie, ed., 131-170, Edinburgh Univ. Press, Edinburgh, 1968. (REP)

Amarel, S. (1969), "On the Representation of Problems and Goal Directed Procedures for Computers," Comm. Am. Soc. Cybernetics, Vol. 1, No. 2, 1969. (SEARCH)

Ambler, A. P. et al. (1973), "A Versatile Computer-Controlled Assembly System," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (ROB, VIS)

Anderson, J. R. and Bower, G. H. (1973), Human Associative Memory, V. H. Winston and Sons, Washington, D.C., 1973. (PSYC)

Andrews, P. B. (1968), "Resolution with Merging," J. ACM, Vol. 15, 367-381, 1968. (TP)

Balzer, R. M. (1972), "Automatic Programming," Institute Technical Memo, Univ. of So. Calif./ Information Sciences Institute, Sept. 1972. (PROG-G)

Banerji, R. B. (1969), Theory of Problem Solving, American Elsevier Publishing Company, New York, 1969. (GAME)

Banerji, R. B. and Ernst, G. W. (1972), "Strategy Construction Using Homomorphisms Between Games," Artificial Intelligence, Vol. 3, No. 4, 223-249, Winter 1972. (GAME)

Barnett, Jeffrey (1972), "A Vocal Data Management System," 1972 Conf. on Speech Communication and Processing, Newton, MA, 24-26 April 1972, U.S. Air Force Cambridge Research Laboratories, Bedford, MA, 22 Feb. 1972, 340-343 (AFCRL-72-0120, Special Reports Number 131). (LANG)

Barrow, H. and Popplestone, R. (1971), "Relational Descriptions in Picture Processing," Machine Intelligence, Vol. 6, B. Meltzer and D. Michie, eds., 377-396, Edinburgh Univ. Press, 1971. (VIS)

Barrow, H. G. and Crawford, G. F. (1972), "The Mark 1.5 Edinburgh Robot Facility," Machine Intelligence, Vol. 7, B. Meltzer and D. Michie, eds., 465-480, American Elsevier, 1972. (ROB)

Barrow, H. G., Ambler, A. P., and Burstall, R. M. (1972), "Some Techniques for Recognizing Structures in Pictures," Frontiers of Pattern Recognition, S. Watanabe, ed., 1-29, Academic Press, New York, 1972. (VIS)

Bartlett, F. C. (1958), Thinking, Basic Books, New York. (PSYC)

Bartlett, F. C. (1932), Remembering, Cambridge Univ. Press, Cambridge. (PSYC)

Becker, J. D. (1970), "An Information-Processing Model of Intermediate-Level Cognition," Memo No. 119, Stanford Artificial Intelligence Project, Computer Science Dept., Stanford Univ., Stanford, CA. Also Report No. 2335, Bolt, Beranek, and Newman, Inc., Cambridge, MA. (PSYC)

Becker, J. D. (1973), "A Model for the Encoding of Experiential Information," Computer Models of Thought and Language, Schank and Colby, eds., W. H. Freeman and Co., San Francisco, 1973. (PSYC)

Berlekamp, E. (1963), "Program for Double-Dummy Bridge Problems, A Strategy for Mechanical Game Playing," J. ACM, Vol. 10, No. 3, 357-364, July 1963. (GAME)

Berliner, Hans (1970), "Experiences Gained in Constructing and Testing a Chess Program," Proc. IEEE Systems Science and Cybernetics Conf., 216-223, Pittsburgh, PA, Oct. 1970. (GAME)

Berliner, Hans J. (1973), "Some Necessary Conditions for a Master Chess Program," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (GAME)

Bernstein, A. et al. (1959), "A Chess Playing Program for the IBM 704," Proc. Western Joint Comp. Conf. AIEE, 157-159, Mar. 1959. (GAME)

Blair, F. W., Griesmer, J. H., and Jenks, R. D. (1970), "An Interactive Facility for Symbolic Mathematics," Proc. Intl. Comp. Symposium, 394-419, Bonn, Germany, 1970. (AIDS)

Bledsoe, W. W. (1971), "Splitting and Reduction Heuristics in Automatic Theorem Proving," Artificial Intelligence, Vol. 2, No. 1, 55-77, Spring 1971. (TP)

Bledsoe, W. W., Boyer, R. S., and Henneman, W. H. (1972), "Computer Proofs of Limit Theorems," Artificial Intelligence, Vol. 3, 27-60, 1972. (TP)

Bledsoe, W. W. and Bruel, P. (1973), "A Man-Machine Theorem Proving System," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, 1973. (TP)

Bobrow, D. G. (1964a), "Natural Language Input for a Computer Problem-Solving System," Doctoral dissertation, MIT, Sept. 1964. Reprinted in Semantic Information Processing, M. Minsky, ed., MIT Press, Cambridge, MA, 1968. (LANG)

Bobrow, D. G. (1964b), "A Question-Answering System for High-School Algebra Word Problems," Proc. AFIPS Fall Joint Comp. Conf., 591-614, 1964. (LANG)

Bobrow, D. G. and Fraser, J. B. (1969), "An Augmented State Transition Network Analysis Procedure," Proc. Intl. Joint Conf. on Artificial Intelligence, 557-567, Washington, D.C., 1969. (LANG)

Bobrow, D. G. and Raphael, B. (1973), "New Programming Languages for AI Research," SRI Artificial Intelligence Center Tech. Note 82, Stanford Research Institute, Menlo Park, CA, Aug. 1973. To appear in Computer Surveys, 1974. (SYS-G)

Bobrow, D. G. and Wegbreit, B. (1973a), "A Model for Control Structures for Artificial Intelligence Programming Languages," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (SYS)

Bobrow, D. G. and Wegbreit, B. (1973b), "A Model and Stack Implementation of Multiple Environments," CACM, Vol. 16, No. 10, Oct. 1973. (SYS)

Bolles, R. and Paul, R. (1973), "The Use of Sensory Feedback in a Programmable Assembly System," Memo CS-396, Comp. Sci. Dept., Stanford Univ., Artificial Intelligence Lab. Memo AIM-220, Oct. 1973. (ROB)

Boyer, Robert S. and Moore, J Strother (1973), "Proving Theorems About LISP Functions," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (PROG)

Brice, C. R. and Fennema, C. L. (1970), "Scene Analysis Using Regions," Artificial Intelligence, Vol. 1, No. 3, 205-226. (VIS)

Bruce, Bertram (1972), "A Model for Temporal References and Its Application in a Question Answering Program," Artificial Intelligence, Vol. 3, 1-26, 1972. (REP)

Bruner, J. S., Goodnow, J. J., and Austin, G. A. (1956), A Study of Thinking, Wiley, New York. (PSYC)

Buchanan, B. G., Sutherland, G., and Feigenbaum, E. (1969), "Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry," Machine Intelligence, Vol. 4, B. Meltzer and D. Michie, eds., 209-254, American Elsevier Publishing Company, New York, 1969. (AIDS)

Buchanan, B. G. and Lederberg, J. (1971), "The Heuristic DENDRAL Program for Explaining Empirical Data," Proc. IFIP Congress, Vol. 71, Ljubljana, Yugoslavia, 1971. Also Stanford Univ. AIM 141. (AIDS)

Buchanan, J. R. and Luckham, D. C. (1974), "On Automating the Construction of Programs," Stanford Artificial Intelligence Lab. Memo, forthcoming 1974. (DED, PROG)

Bundy, Alan (1973), "Doing Arithmetic with Diagrams," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford CA, Aug. 1973. (TP)

Burstall, R. M., Collins, J. S., and Popplestone, R. J. (1971), Programming in POP2, 279-282, Edinburgh Univ. Press, Edinburgh, 1971. (SYS)

Carbonell, J. R. (1971), "AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction," IEEE Trans. on Man-Machine Systems, Vol. MMS-11, No. 4, 190-202, Dec. 1970. (AIDS)

Chang, C. L. and Slagle, J. R. (1971), "An Admissible and Optimal Algorithm for Searching and/or Graphs," Artificial Intelligence, Vol. 2, 117-128, 1971. (SEARCH)

Chang, C. L. and Lee R. C. (1973), Symbolic Logic and Mechanical Theorem Proving, Academic Press, 1973. (TP-G)

Charniak, E. C. (1972), "Toward a Model of Children's Story Comprehension," AI TR-266, MIT, Cambridge, MA. (LANG, DED)

Chase, W. G. (1973), Visual Information Processing, Academic Press, 1973. (PSYC-G)

Chomsky, N. (1956), "Three Models for the Description of Language," IRE Trans. on Info. Theory, Vol. IT-2(3), 113-124, 1956. (PSYC, LANG)

Chomsky, N. (1957), Syntactic Structures, Mouton, The Hague, 1957. (PSYC, LANG)

Chomsky, N. (1965), Aspects of the Theory of Syntax, MIT Press, Cambridge, MA, 1965. (LANG)

Clowes, M. G. (1971), "On Seeing Things," Artificial Intelligence, Vol. 2, No. 1, 79-116, 1971. (VIS)

Colby, K. M. and Enea, H. (1967), "Heuristic Methods for Computer Understanding of Natural Language in Context Restricted On-Line Dialogues," Mathematical Biosciences, Vol. 1, 1-25, 1967. (LANG)

Colby, K. M., Weber, S, and Hilf, F. D. (1971), "Artificial Paranoia," Artificial Intelligence, Vol. 2, No. 1, 1-25, Spring 1971. (PSYC)

Coles, L. S. (1970), "An Experiment in Robot Tool Using," Stanford Research Institute Tech. Note No. 41, Stanford Research Institute, Menlo Park, CA, Oct. 1970. (ROB)

Coles, L. S. (1972), "Techniques for Information Retrieval Using an Inferential Question-Answering System with Natural-Language Input," SRI Artificial Intelligence Center Tech. Note 74, Stanford Research Institute, Menlo Park, CA, Nov. 1972. (LANG)

Collins, A. M. and Quillian, M. R. (1972), "Retrieval Time from Semantic Memory," J. Verbal Learning and Verbal Behavior, Vol. 8, 240-247, 1969. (PSYC)

Collins, N. and Michie, D., eds. (1968), Machine Intelligence, Vol. 1, American Elsevier Publishing Company, New York, 1967. (GEN)

Corey, E. J. (1969), "Computer-Assisted Design of Complex Organic Synthesis," Science, 10 Oct. 1969. (AIDS)

Dale, E. and Michie, D., eds. (1968), Machine Intelligence, Vol. 2, American Elsevier Publishing Company, New York, 1968. (GEN)

Darlington, J. L. (1971), "A Partial Mechanization of Second-Order Logic," Machine Intelligence, Vol. 6, 91-100, B. Meltzer and D. Michie, eds., Edinburgh Univ. Press, Edinburgh. (TP)

Davies, D. J. M. (1971), "POPLER: A POP-2 Planner," Memo MIP-R-89, School of Artificial Intelligence, Univ. of Edinburgh. (SYS)

Davis, M. and Putnam, H. (1960), "A Computing Procedure for Quantification Theory," J. ACM, Vol. 7, 201-215, 1960. (TP)

Derksen, J., Rulifson, J. F., and Waldinger, R. J. (1972), "The QA4 Language Applied to Robot Planning," AFIPS Conf. Proc., Vol. 41, Part II, 1181-1187, Fall Joint Comp. Conf., 1972. (DED)

Deutsch, L. P. (1973), "An Interactive Program Verifier," Ph.D. thesis, Dept. of Computer Science, Univ. of California, Berkeley, 1973. (PROG)

Doran, J. and Michie, D. (1966), "Experiments with the Graph Traverser Program," Proc. Roy. Soc. A, Vol. 294, 235-259, 1966. (SEARCH)

Dreyfus, H. (1965), "Alchemy and Artificial Intelligence," RAND Corporation Paper P3244 (AD 625 719), Dec. 1965. (GEN)

Dreyfus, H. L. (1972), What Computers Can't Do, Harper and Row, 1972. (GEN)

Duda, R. and Hart, P. (1970), "Experiments in Scene Analysis," Proc. 1st Natl. Symposium on Industrial Robots, IIT Research Institute, Chicago, IL, Apr. 1970. (VIS)

Duda, R. and Hart, P. (1973), Pattern Classification and Scene Analysis, John Wiley & Sons, New York, 1973. (VIS-G)

Eastman, C. M. (1971a), "GSP: A System for Computer Assisted Space Planning," Proc. 8th Annual Design Automation Workshop, Atlantic City, NJ. (AIDS)

Eastman, C. M. (1971b), "Heuristic Algorithms for Automated Space Planning," Proc. 2d Intl. Joint Conf. on Artificial Intelligence, Imperial College, London, 1971. (AIDS)

Edmundson, H. P., ed. (1961), Proc. of the Natl. Symposium on Machine Translation, Prentice-Hall, Englewood Cliffs, NJ, 1961. (LANG)

Edwards, D. and Hart, T. (1961), "The Alpha-Beta Heuristic," MIT Artificial Intelligence Memo No. 30 (revised), 28 Oct. 1963. Originally printed as "The Tree Prune (TP) Algorithm," 4 Dec. 1961. (GAME)

Ejiri, M. et al. (1971), "An Intelligent Robot with Cognition and Decision-Making Ability," Proc. of 2d Intl. Joint Conf. on Artificial Intelligence, Imperial College, London, 350-358, 1971. (ROB, VIS)

Ejiri, E. et al. (1972), "A Prototype Intelligent Robot that Assembles Objects from Plan Drawings," IEEE Trans. Comp., 161-170, Feb. 1972. (ROB, VIS)

Elcock, E. W. et al. (1971), "ABSET, A Programming Language Based on Sets: Motivation and Examples," Machine Intelligence, Vol. 6, Edinburgh Univ. Press, Edinburgh, 1971. (SYS)

Elspas, B. (1972), "The Semiautomatic Generation of Inductive Assertions for Program Correctness Proofs," Report No. 55, Seminar, Des Instituts fur Theorie der Automaten und Schaltnetzwerke, Gesellschaft fur Mathematik und Datenverarbeitung, Bonn, 21 Aug. 1972. (PROG)

Elspas, B. et al. (1973), "Design of an Interactive System for Verification of Computer Programs," SRI Report, Project 1891, Stanford Research Institute, Menlo Park, CA, July 1973. (PROG)

Enea, H. and Colby, K. M. (1973), "Idiolectric Language-Analysis for Understanding Doctor-Patient Dialogues," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, 278-284, Stanford Univ., Stanford, CA, 1973. (LANG)

Engelman, C. (1969), "MATHLAB 68," Information Processing 68, A. J. H. Morrell, ed., 462-467, North-Holland Publishing Company, Amsterdam, 1969. (AIDS)

Ernst, G. W. and Newell, Allen (1969), GPS: A Case Study in Generality and Problem Solving, Academic Press, New York, 1969. (DED)

Ernst, G. W. (1971), "The Utility of Independent Subgoals in Theorem Proving," Information and Control, Apr. 1971. (TP)

Ernst, H. (1961), "MH-1, A Computer-Operated Mechanical Hand," D.Sc. dissertation, Dept. of Elec. Eng., MIT, Cambridge, MA. (ROB)

Fahlman, S. (1973), "A Planning System for Robot Construction Tasks," Report AI TR-283, Artificial Intelligence Lab., MIT, Cambridge, MA, May 1973. (DED)

Falk, G. (1970), "Computer Interpretation of Imperfect Line Data as a Three-Dimensional Scene," Ph.D. thesis, Stanford Univ., Comp. Sci. Dept., 1970. Available as CS-180 and AIM-132. (VIS)

Falk, G. (1972), "Interpretation of Imperfect Line Data as a Three Dimensional Scene," Artificial Intelligence, Vol. 3, No. 2, 101-144, 1972. (VIS)

Feigenbaum, E. A. (1961), "The Simulation of Verbal Learning Behavior," Proc. Western Joint Comp. Conf., 121-132, 1961. Also in Computers and Thought, E. A. Feigenbaum and J. Feldman, eds., 297-309, McGraw-Hill Book Company, New York, 1963. (PSYC)

Feigenbaum, E. (1963), "Artificial Intelligence Research," IEEE Trans. Info. Theory, Vol. IT-9, No. 4, 248-261, Oct. 1963. (GEN)

Feigenbaum, E. (1969), "Artifical Intelligence: Themes in the Second Decade," Information Processing 68, Vol. 2, A.J.H. Morrell, ed., 1008-1022, North-Holland Publishing Company, Amsterdam, 1969. Also printed as Stanford Univ. Artificial Intelligence Project Memo No. 67, 15 Aug. 1968. (GEN)

Feigenbaum, E. and Feldman, J., eds. (1963), Computers and Thought, McGraw-Hill Book Company, New York, 1963. (GEN)

Feldman, J. A. et al. (1969), "The Stanford Hand-Eye Project," Proc. 1st Intl. Joint Conf. on Artificial Intelligence, 521-526, Washington, D.C., 1969. (ROB)

Feldman, J. A. et al. (1971), "The Use of Vision and Manipulation to Solve the Instant Insanity Puzzle," Proc. 2d Intl. Joint Conf. on Artificial Intelligence, London, 1971. (ROB, VIS)

Feldman, J. A. et al. (1972), "Recent Developments in SAIL--An ALGOL-Based Language for Artificial Intelligence," 1972 FJCC Proc., 5-7 Dec. 1972, Anaheim, CA. (SYS)

Feldman, J. A. and Rovner, P. D. (1969), "An ALGOL-Based Associative Language," Comm. ACM, 434-449, Aug. 1969. (SYS)

Fikes, R. E. (1968), "A Heuristic Program for Solving Problems Stated as Nondeterministic Procedures," Ph.D. thesis, Carnegie-Mellon Univ., 1968. (DED, SYS)

Fikes, R. E. (1970), "REF-ARF: A System for Solving Problems Stated as Procedures," Artificial Intelligence, Vol. 1(1), 1970. (DED, SYS)

Fikes, R. E. and Nilsson, N. J. (1971), "STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving," Artificial Intelligence, Vol. 2, 189-208, 1971. (REP, DED)

Fikes, R. E., Hart, P. E., and Nilsson, N. J. (1972a), "Learning and Executing Generalized Robot Plans," Artificial Intelligence, Vol. 3, 251-288, 1972. (DED)

Fikes, R. E., Hart, P. E., and Nilsson, N. J. (1972b), "Some New Directions in Robot Problem Solving," B. Meltzer and D. Michie, eds., Machine Intelligence, Vol. 7, Edinburgh Univ. Press, Edinburgh, 1972. (ROB-G)

Firschein, Oscar et al. (1973), "Forecasting and Assessing the Impact of Artificial Intelligence on Society," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (GEN)

Fischler, Martin A. and Elschlager, Robert A. (1973), "The Representation and Matching of Pictorial Structures," IEEE Trans. on Computers, Vol. C-22, No. 1, 67-92, Jan. 1973. (VIS)

Flanagan, J. L. (1965), Speech Analysis, Synthesis and Perception, Academic Press, New York. (LANG)

Floyd, R. W. (1967), "Assigning Meanings to Programs," Proc. of a Symposium in Applied Mathematics, Vol. 19, J. T. Schwartz, ed., Am. Math. Soc., 19-32, 1967. (PROG)

Forsen, G. (1968), "Processing Visual Data with an Automaton Eye," Pictorial Pattern Recognition, 471-502, Thompson Book Co., Washington, D.C., 1968. (VIS)

Friedman, Joyce (1971), A Computer Model of Transformational Grammar, 166, American Elsevier Publishing Company, New York, 1971. (LANG)

Friedman, L. (1967), "Instinctive Behavior and Its Computer Synthesis," Behavioral Science, Vol. 12, No. 2, Mar. 1967. (PSYC)

Fuller, S., Gaschnig, J., and Gillogly, J. (1973), "Analysis of the Alpha-Beta Pruning Algorithm," Carnegie-Mellon Univ., Dept. of Comp. Sci. Report, Pittsburgh, PA, July 1973. (GAME)

Gelernter, H. (1960), "Realization of a Geometry Theorem-Proving Machine," Proc. 1959 Intl. Conf. on Info. Proc., 273-282, UNESCO, Paris. Also in Computers and Thought, E. A. Feigenbaum and J. Feldman, eds., 134-152, McGraw-Hill Book Company, New York, 1963. (TP, SEARCH)

Gelernter, H., Hansen, J. R., and Loveland, D. W. (1963), "Empirical Explorations of the Geometry Theorem Machine," Computers and Thought, E. A. Feigenbaum and J. Feldman, eds., 153-163, McGraw-Hill Book Company, New York, 1963. (TP)

Gere, W. S. (1966), "Heuristics in Job Shop Scheduling," Management Science, Vol. 13, 167-190. (AIDS)

Gillogly, J. (1972), "The Technology Chess Program," Artificial Intelligence, Vol. 3, No. 3, 145-163, Fall 1972. (GAME)

Goldstein, A. Jay, Harmon, Leon D., and Lesk, Ann B. (1971), "Identification of Human Faces," Proc. IEEE, Vol. 59, No. 5, 748-760, May 1971. (VIS)

Good, I. J. (1967), "A Five-Year Plan for Automatic Chess," Machine Intelligence, Vol. 2, E. Dale and D. Michie, eds., 89-118, Edinburgh Univ. Press, Edinburgh, 1967. (GAME)

Good, D. I. and London, R. L. (1968), "Interval Arithmetic for the Burroughs B5500: Four ALGOL Procedures and Proofs of Their Correctness," Comp. Sci. Tech. Report No. 26, Univ. of Wisconsin, 1968. (PROG)

Gordon, G. (1969), System Simulation, Prentice-Hall, Englewood Cliffs, NJ, 1969. (SYS)

Grape, G. (1973), "Model Based (Intermediate Level) Computer Vision," Ph.D. thesis, Comp. Sci. Dept., Stanford Univ., Stanford, CA, 1973. (VIS)

Greenblatt, R. et al. (1967), "The Greenblatt Chess Program," Proc. AFIPS Fall Joint Comp. Conf., 801-810, 1967. (GAME)

Green, B. F. et al. (1961), "Baseball: An Automatic Question Answerer," Proc. Western Joint Comp. Conf., 219-224. (LANG)

Green, C. (1969a), "Theorem-Proving by Resolution as a Basis for Question-Answering Systems," Machine Intelligence, Vol. 4, B. Meltzer and D. Michie, eds., 183-205, American Elsevier Publishing Company, New York, 1969. (REP, DED)

Green, C. (1969b), "The Application of Theorem-Proving to Question-Answering Systems," Doctoral dissertation, Elec. Eng. Dept., Stanford Univ., Stanford, CA, June 1969. Also printed as Stanford Artificial Intelligence Project Memo AI-96, June 1969. (REP, DED, PROG, TP)

Green, C. (1969c), "Application of Theorem-Proving to Problem Solving," Proc. Intl. Joint Conf. Artificial Intelligence, Donald E. Walker and Lewis M. Norton, eds., Washington, D.C., May 1969. (REP, DED)

Gregory, R. L. (1966), Eye and Brain, McGraw-Hill Book Company, New York. (PSYC)

Gregory, R. L. (1970), The Intelligent Eye, Weidenfeld and Nicolson, London, 1970. (PSYC)

Griesmer, J. H. and Jenks, R. D. (1971), "SCRATCHPAD/1--An Interactive Facility for Symbolic Mathematics," Proc. ACM 2d Symposium on Symbolic and Algebraic Manipulation, S. R. Petrick, ed., Los Angeles, CA, 23-25 Mar. 1971. (AIDS)

Griffith, A. K. (1970), "Computer Recognition of Prismatic Solids," MAC Tech. Report 73, Project MAC, MIT, Cambridge, MA. (VIS)

Griffith, A. K. (1973), "Mathematical Models for Automatic Line Detection," J. ACM, 62-80, 1973. (VIS)

Gross, Louis N. and Walker, Donald E. (1969), "On-Line Computer Aids for Research in Linguistics," Information Processing, Vol. 68, A. J. H. Morrell, ed., North-Holland Publishing Company, Amsterdam, 1969. (LANG)

Guard, J. R. et al. (1969), "Semi-Automated Mathematics," J. ACM, Vol. 16, 49-62, 1969. (TP)

Guzman, A. (1968a), "Decomposition of a Visual Scene Into Three-Dimensional Bodies," Proc. AFIPS 1968 Fall Joint Comp. Conf., Vol. 33, 291-304, Thompson Book Co., Washington, D.C. (VIS)

Guzman, A. (1968b), "Computer Recognition of Three-Dimensional Objects in a Visual Scene," MAC Tech. Report 59, thesis, Project MAC, MIT, Cambridge, MA, 1968. (VIS)

Guzman, A. (1971), "Analysis of Curved Line Drawings Using Context and Global Information," Machine Intelligence, Vol. 6, B. Meltzer and D. Michie, eds., 325-376, Edinburgh Univ. Press, Edinburgh. (VIS)

Haessler, R. W. (1971), "A Heuristic Programming Solution to a Nonlinear Cutting Stock Problem," Management Science, Vol. 17B, 793-802. (AIDS)

Harris, Z. (1951), Structural Linguistics, Univ. of Chicago Press, Chicago, 1951. (LANG)

Harris, Z. (1961), String Analysis of Sentence Structure, Mouton, The Hague, 1961. (LANG)

Hart, P. E. et al. (1972), "Artificial Intelligence--Research and Applications," Annual Tech. Report to ARPA, Contract DAHC04-72-C-0008, Stanford Research Institute, Menlo Park, CA, Dec. 1972. (ROB)

Hart, P., Nilsson, N., and Raphael, B. (1968), "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Trans. Sys. Sci. Cybernetics, Vol. 4, No. 2, 100-107, 1968. (SEARCH)

Hart, T. (1961), "SIMPLIFY," Memo 27, Artificial Intelligence Group, Project MAC, MIT, Cambridge, MA, 1961.. (AIDS)

Hayes, P. (1971), "A Logic of Action," Machine Intelligence, Vol. 6, B. Meltzer and D. Michie, eds., 495-520, American Elsevier Publishing Company, 1971. (REP)

Hayes, P. F. (1973), "The Frame Problem and Related Problems in Artificial Intelligence," Artificial and Human Thinking, A. Elithorn and D. Jones, eds., Elsevier Scientific Publishing Co., New York, 1973. (REP)

Hearn, A. C. (1968), "REDUCE, A User-Oriented Interactive System for Algebraic Simplification," Interactive Systems for Experimental Applied Mathematics, 79-90, M. Klerer and J. Reinfelds, eds., Academic Press, New York and London, 1968. (AIDS)

Hearn, Anthony C. (1971), "REDUCE 2: A System and Language for Algebraic Manipulation," Proc. ACM 2d Symposium on Symbolic and Algebraic Manipulation, S. R. Petrick, ed., 23-25 Mar. 1971, Los Angeles, CA. (AIDS)

Hendrix, G. (1973), "Modeling Simultaneous Actions and Continuous Processes," Artificial Intelligence, Vol. 4, 145-180, 1973. (REP)

Hendrix, Gary G., Thompson, Craig W., and Slocum, Jonathan (1973), "Language Processing via Canonical Verbs and Semantic Models," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (LANG)

Hewitt, C. (1969), "PLANNER: A Language for Proving Theorems in Robots," 1st Intl. Joint Conf. on Artificial Intelligence, Washington, D.C., 1969. (REP, SYS, DED)

Hewitt, C. (1971), "Procedural Embedding of Knowledge in PLANNER," Proc. 2d Intl. Joint Conf. on Artificial Intelligence, British Computer Society, London, England, 167-182, 1971. (REP, SYS, DED)

Hewitt, C. (1972), "Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot," Ph.D. thesis, Dept. of Math., MIT, Cambridge, MA, 1972. (SYS, REP, DED)

Hewitt, C., Bishop, P., and Steiger, R. (1973), "A Universal Modular Actor Formalism for Artificial Intelligence," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (SYS)

Hintzman, D. L. (1968), "Explorations with a Discrimination Net Model for Paired-Associate Learning," J. Mathematical Psychology, Vol. 5, 123-162, 1968. (PSYC)

Horn, B. K. P. (1970), "Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View," MAC Tech. Report 79, Project MAC, MIT, Cambridge, MA, Nov. 1970. (VIS)

Horn, B. K. P. (1971), "The Binford-Horn Line Finder," Vision Flash, No. 16, Artificial Intelligence Laboratory, MIT, Cambridge, MA. Later issued as MIT Artificial Intelligence Lab Memo 285, Mar. 1973. (VIS)

Huet, G. P. (1973a), "A Unification Algorithm for Type Theory," IRIA Laboria, 1973. (TP)

Huet, G. P. (1973b), "A Mechanization of Type Theory," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (TP)

Huffman, D. A. (1971), "Impossible Objects as Nonsense Sentences," Machine Intelligence, Vol. 6, B. Meltzer and D. Michie, eds. 295-323, Edinburgh Univ. Press, Edinburgh, 1971. (VIS)

Hunt, E. B. (1962), Concept Formation, John Wiley & Sons, New York. (PSYC)

Hunt, E. B. (1974), Artificial Intelligence, Academic Press, 1974. (GEN)

Hunt, E. B. and Hovland, C. I. (1961), "Programming a Model of Human Concept Formation," Proc. Western Joint Comp. Conf., Vol. 19, 145-155. (PSYC)

Igarashi, W., London, R., and Luckham, D. (1973), "Automatic Verification of Programs I: A Logical Basis and Implementation," Stanford Univ. Artificial Intelligence Lab. Memo, No. 200, Stanford Univ., Stanford, CA, May 1973. (PROG)

Jackson, P. C. (1974), Introduction to Artificial Intelligence, Mason and Lipscomb, New York, 1974. (GEN)

Kaplan, R. M. (1972), "Augmented Transition Networks as Psychological Models of Sentence Comprehension," Artificial Intelligence, Vol. 3, 77-100, 1972. (PSYC)

Kaplan, R. M. (1973), "A General Syntactic Processor," Natural Language Processing, R. Rustin, ed., 293-241, Algorithmic Press, New York, 1973. (LANG)

Katz, S. M. and Manna, Z. (1973), "Heuristic Approach to Program Verification," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, 500-512, Stanford Univ., Stanford, CA, Aug. 1973. (PROG)

Kay, Martin (1964), "A General Procedure for Rewriting Strings," presented at the 1964 Annual Meeting, Association for Machine Translation and Computational Linguistics, Indiana Univ., Bloomington. (LANG)

Kay, Martin (1967), "Experiments with a Powerful Parser," RM-5452-PR, RAND Corporation, Santa Monica, CA, 1967. (LANG)

Kay, Martin (1973), "The Mind System," Natural Language Processing, R. Rustin, ed., 155-187, Courant Computer Science Symposium 8, 20-21 Dec., Algorithmics Press, Inc., New York, 1973. (LANG)

Kelly, M. (1970), "Visual Identification of People by Computer," Memo AI-130, Comp. Sci. Dept., Stanford Univ., Stanford, CA, July 1970. (VIS)

King, J. (1969), "A Program Verifier," Doctoral dissertation, Comp. Sci. Dept., Carnegie-Mellon Univ., Pittsburgh, PA, 1969. (PROG)

Kister, J. et al. (1957), "Experiments in Chess," J. ACM, Vol. 4, 174-177, Apr. 1957. (GAME)

Kling, R. E. (1971), "A Paradigm for Reasoning by Analogy," Artificial Intelligence, Vol. 2, No. 2, 147-178, Fall 1971. (DED)

Koffman, Elliot B. and Blount, Sumner E. (1973), "Artificial Intelligence and Automatic Programming in CAI," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (AIDS)

Korsvold, K. (1965), "An On-Line Algebraic Simplification Program," Artificial Intelligence Project Memo No. 37, Stanford Univ., Stanford, CA, Nov. 1965. (AIDS)

Kotok, A. (1962), "A Chess Playing Program for the IBM 7090," Bachelor's thesis, MIT, 1962. (GAME)

Kowalski, R. (1970), "Search Strategies for Theorem Proving," Machine Intelligence, Vol. 5, B. Meltzer and D. Michie, eds., 181-200, American Elsevier Publishing Company, New York, 1970. (TP)

Kowalski, R. and Kuehner, D. (1971), "Linear Resolution with Selection Function," Artificial Intelligence, Vol. 2, 227-260, 1971. (TP)

Kulikowski, C. A. and Weiss, S. (1972), "The Medical Consultant Program--Glaucoma," Tech. Report TR-5, Computers in Biomedicine, Dept. of Comp. Sci., Rutgers Univ., New Brunswick, NJ, July 1972. (AIDS)

Kuno, Susumu and Oettinger, Anthony G. (1963), "Multiple-Path Syntactic Analyzer," Information Processing 1962, 306-312, North-Holland Publishing Company, Amsterdam, 1963. (LANG)

Levy, D. N. L. (1970), "Computer Chess--A Case Study," Machine Intelligence, Vol. 6, B. Meltzer and D. Michie, eds., 151-163, Edinburgh Univ. Press, 1970. (GAME)

Lighthill, J. (1973), "Artificial Intelligence: A General Survey," Artificial Intelligence: A Paper Symposium, Science Research Council Pamphlet, Science Research Council, State House, High Holburn, London, Apr. 1973. (GEN)

Lin, Shen (1970), "Heuristic Techniques for Solving Large Combinatorial Problems on a Computer," Theoretical Approaches to Non-Numerical Problem-Solving, R. Banerji and M. Mesarovic, eds., 410-418, Springer-Verlag, New York, 1970. (AIDS)

Lindsay, P. H. and Norman, D. A. (1972), Human Information Processing: An Introduction to Psychology, Academic Press, 1972. (PSYC-G)

Locke, William N. and Booth, A. Donald, eds. (1955), Machine Translation of Languages: Fourteen Essays, John Wiley & Sons, New York, 1955. (LANG)

Londe, Dave L. and Schoene, William J. (1968), "TGT: Transformational Grammar Tester," Proc. AFIPS Conf., Vol. 32, 1968 Spring Joint Comp. Conf., 385-393, Thompson Book Co., Washington, D.C. (LANG)

London, R. (1970), "Bibliography on Proving the Correctness of Computer Programs," Machine Intelligence, Vol. 5, B. Meltzer and D. Michie, eds., 569-580, American Elsevier Publishing Company, New York, 1970. (PROG-G)

London, R. L. (1972), "The Current State of Proving Programs Correct," Proc. ACM 25th Annual Conf., 39-46, 1972. (PROG-G)

Loveland, D. W. (1970), "A Linear Format for Resolution," Symposium on Automatic Demonstration, M. Laudet, ed., Springer-Verlag, New York, 1970. (TP)

Luckham, D. (1969), "Refinement Theorems in Resolution Theory," Stanford Artificial Intelligence Project Memo AI-81, 24 Mar. 1969. Also in Proc. IRIA 1968 Symp. Autom. Demonstration, Lecture Notes on Mathematics No. 125, Springer-Verlag, New York, 1970. (TP)

Luckham, D. and Nilsson, N. (1971), "Extracting Information from Resolution Proof Trees," Artificial Intelligence, 1971. (TP)

Manna, Z. (1969), "The Correctness of Programs," J. Computer Syst. Sci., Vol. 3, May 1969. (PROG)

Manna, Z. and Waldinger, R. J. (1971), "Toward Automatic Program Synthesis," Comm. ACM, Vol. 14, No. 3, 151-165, Mar. 1971. (PROG-G)

Martin, W. A. (1967), "Symbolic Mathematical Laboratory," MAC Tech. Report 36 (thesis), Project MAC, MIT, Cambridge, MA, Jan. 1967. (AIDS)

Martin, W. A. and Fateman, R. J. (1971), "The MACSYMA System," Proc. ACM 2d Symposium on Symbolic and Algebraic Manipulation, S. R. Petrick, ed., Los Angeles, CA, 23-25 Mar. 1971. (AIDS)

McCarthy, J. (1958), "Programs with Common Sense," in "Mechanization of Thought Processes," Vol. 1, 77-84, Proc. Symp., Nat. Phys. Lab., London, 24-27, Nov. 1958. Reprinted in Semantic Information Processing, M. Minsky, ed., 403-410, MIT Press, Cambridge, MA, 1968. (REP, DED)

McCarthy, J. (1960), "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part 1," Comm. ACM, Vol. 3, No. 4, 184-195, Apr. 1960. (SYS)

McCarthy, John (1961), "Computer Programs for Checking Mathematical Proofs," Proc. Amer. Math. Soc. on Recursive Function Theory, New York, Apr. 1961. (TP)

McCarthy, J. (1962), "Towards a Mathematical Science of Computation," Proc. IFIP Congr., Vol. 62, North-Holland Publishing Company, Amsterdam, 1962. (PROG)

McCarthy, J. (1963), "Situations, Actions and Causal Laws," Memo. No. 2, Stanford Univ. Artificial Intelligence Project, 1963. Reprinted in Semantic Information Processing, M. Minsky, ed., 410-418, MIT Press, Cambridge, MA, 1968. (REP)

McCarthy, J. and Painter, J. A. (1967), "Correctness of a Compiler for Arithmetic Expressions," in Proc. Symp. Applied Mathematics, Vol. 19, Math. Aspects of Computer Science, J. T. Schwarcz, ed., 33-41, American Mathematical Society, Providence, RI, 1967. (PROG)

McCarthy, J. et al. (1968), "A Computer with Hands, Eyes, and Ears," Proc. 1968 Fall Joint Comp. Conf., Vol. 33, 329-338, Thompson Book Company, Washington, D.C. (ROB)

McCarthy, J. and Hayes, P. (1969), "Some Philosophical Problems from the Standpoint of Artificial Intelligence," Machine Intelligence, Vol. 4, American Elsevier Publishing Company, New York, 1969. (REP)

McDermott, D. V. and Sussman, G. J. (1972), "The CONNIVER Reference Manual," MIT, Artificial Intelligence Lab., Memo No. 259, May 1972. (SYS)

Meltzer, B. and Michie, D., eds. (1969), Machine Intelligence, Vol. 4, American Elsevier Publishing Company, New York, 1969. (GEN)

Meltzer, B. and Michie, D., eds. (1970), Machine Intelligence, Vol. 5, American Elsevier Publishing Company, New York, 1970. (GEN)

Meltzer, B. and Michie, D., eds. (1971), Machine Intelligence, Vol. 6, American Elsevier Publishing Company, New York, 1971. (GEN)

Meltzer, B. and Michie, D., eds. (1972), Machine Intelligence, Vol. 7, American Elsevier Publishing Company, New York, 1972. (GEN)

Michie, D., ed. (1968), Machine Intelligence, Vol. 3, American Elsevier Publishing Company, Princeton, NJ, 1968. (GEN)

Miller, G. A. (1956), "The Magical Number Seven, Plus or Minus Two," Psychological Review, Vol. 63, 81-97. (PSYC)

Miller, G. A., Galanter, E., and Pribram, K. H. (1960), Plans and the Structure of Behavior, Holt, Rinehart & Winston, New York. (PSYC)

Minker, J., Fishman, D. H., and McSkimin, J. R. (1972), "The Maryland Refutation Proof Procedure System (MRPPS)," TR-208, Comp. Sci. Center, Univ. of Maryland, College Park, MD, 1972. (TP)

Minsky, M. (1961), "Steps Toward Artificial Intelligence," Proc. IRE, Vol. 49, 8-30, Jan. 1961. (SEARCH, GEN)

Minsky, M. L. (1965), "Matter, Mind, and Models," IFIP, 1965. (GEN)

Minsky, M., ed. (1968), Semantic Information Processing, MIT Press, Cambridge, MA. (GEN, LANG-G)

Minsky, M. (1974), "Frame-Systems: A Framework for Representation of Knowledge," forthcoming 1974. (REP)

Minsky, M. and Papert, S. (1972), Progress Report, AI Memo 252, Artificial Intelligence Laboratory, Cambridge, MA. (REP, VIS-G)

Mittman, B. (1973), "Can a Computer Beat Bobby Fischer?," Datamation, 84-87, June 1973. (GAME)

Moore, J. and Newell, A. (1973), "How Can Merlin Understand?," Dept. of Com. Sci. Report, Carnegie-Mellon Univ., Nov. 15, 1973. (REP)

Moses, J. (1967), "Symbolic Integration," MAC Tech. Report 47, Project MAC, MIT, Cambridge, MA, Dec. 1967. (AIDS)

Moses, Joel (1971a), "Symbolic Integration: The Stormy Decade," Proc. ACM 2d Symposium on Symbolic and Algebraic Manipulation, S. R. Petrick, ed., Los Angeles, CA, 23-25 Mar. 1971. (AIDS-G)

Moses, Joel (1971b), "Algebraic Simplification: A Guide for the Perplexed," Proc. ACM 2d Symposium on Symbolic and Algebraic Manipulation, S. R. Petrick, ed., Los Angeles, CA, 23-25 Mar. 1971. (AIDS-G)

Neisser, U. (1967), Cognitive Psychology, Appleton-Century-Crofts, New York. (PSYC)

Nevatia, Ramakant and Binford, Thomas O. (1973), "Structured Descriptions of Complex Objects," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, 641-645, Stanford Univ., Stanford, CA, Aug. 1973. (VIS)

Nevins, A. J. (1972), "A Human Oriented Logic for Automatic Theorem Proving," Tech. Report 268, MIT Artificial Intelligence Laboratory, MIT, Cambridge, MA, Oct. 1972. (TP)

Nevins, J. L., Whitney, D. E., and Simunovic, S. N. (1973), "Report on Advanced Automation," No. R-764, prepared for National Science Foundation, Grant No. GK-34094 and GI-39432X, The Charles Stark Draper Lab., Inc., Cambridge, MA, Nov. 1973. (ROB)

Newell, A., ed. (1961), Information Processing Language V Manual, Prentice-Hall, Englewood Cliffs, NJ. (SYS)

Newell, A. (1967), Studies in Problem Solving: Subject 3 on the Cryptarithmetic Task: Donald + Gerald = Robert, Carnegie-Mellon Univ., Pittsburgh, PA. (PSYC, REP)

Newell, A. (1970), "Remarks on the Relationship Between Artificial Intelligence and Cognitive Psychology," Theoretical Approaches to Non-Numerical Problem Solving, R. B. Banerji and M. D. Mesarovic, eds., Springer-Verlag, Berlin. (PSYC-G)

Newell, A. (1972a), "Production Systems: Models of Control Structures," Visual Information Processing, Wm. Chase, ed., Academic Press, New York, 1972. (REP, PSYC)

Newell, A. (1972b), "A Theoretical Exploration of Mechanisms for Coding the Stimulus," Coding Processes in Human Memory, A. Melter and E. Martin, eds., V. H. Winston, Washington, D.C., 1972. (REP, PSYC)

Newell, A. (1973), "Artificial Intelligence and the Concept of Mind," Computer Models of Thought and Language, Roger Schank and Kenneth Colby, eds., W. H. Freeman & Co., San Francisco, 1973. (GEN)

Newell, A. et al. (1971), Speech Understanding Systems, Advanced Research Projects Agency Report, 1971. Also published by North Holland Publishing Company, Amsterdam, The Netherlands, 1973. (LANG)

Newell, A. and Shaw, J. C. (1957), "Programming the Logic Theory Machine," Proc. West. Joint Comp. Conf., 230-240, 1957. (SYS)

Newell, A., Shaw, J. C., and Simon, Herbert (1957), "Empirical Explorations with the Logic Theory Machine: A Case Study in Heuristics," Proc. Western Joint Comp. Conf. 1957, 218-239. Also in Computers and Thought, E. A. Feigenbaum and J. Feldman, eds., 109-133, McGraw-Hill Book Company, New York, 1963. (DED, PSYC)

Newell, A., Shaw, J. C., and Simon, H. A. (1958a), "Elements of a Theory of Human Problem Solving," Psychological Review, Vol. 65, 151-166, 1958. (PSYC)

Newell, A., Shaw, J., and Simon, H. (1958b), "Chess Playing Programs and the Problem of Complexity," IBM J. Res. Develop., Vol. 2, 320-335, Oct. 1958. Reprinted in Computers and Thought, Feigenbaum and Feldman, eds., 39-70, McGraw-Hill Book Company, New York, 1963. (GAME)

Newell, A., Shaw, J. C., and Simon, H. A. (1960), "Report on a General Problem-Solving Program for a Computer," Information Processing: Proc. Intl. Conf. Information Processing, 256-264, UNESCO, Paris. Also printed in Computers and Automation, July 1959. (DED, PSYC, TP)

Newell, A. and Simon, H. A. (1956), "The Logic Theory Machine: A Complex Information Processing System," IRE Trans. on Information Theory, Vol. IT-2, No. 3, 61-79. (SEARCH, DED, PSYC, TP)

Newell, A. and Simon, H. (1961), "GPS, A Program that Simulates Human Thought," Lernende Automaten, H. Billing, ed., 109-124, R. Oldenbourg, Munich. Reprinted in Computers and Thought, Feigenbaum and Feldman, eds., McGraw-Hill Book Company, New York, 1963. (PSYC)

Newell, A. and Simon, Herbert A. (1972), Human Problem Solving, Prentice-Hall, Englewood Cliffs, NJ, 1972. (PSYC)

Newell, A. and Tonge, F. M. (1960), "An Introduction to Information Processing Language V," Comm. ACM, Vol. 3, 205-211. (SYS)

Nilsson, N. J. (1969a), "Searching Problem-Solving and Game-Playing Trees for Minimal Cost Solutions," Information Processing 68, Vol. 2, A. J. H. Morrel, ed., 1556-1562, North-Holland Publishing Company, Amsterdam, 1969. (SEARCH)

Nilsson, N. J. (1969b), "A Mobile Automaton: An Application of Artificial Intelligence Techniques," Proc. IJCAI, 509-515, May 1969. (ROB)

Nilsson, N. J. (1971), Problem-Solving Methods in Artificial Intelligence, McGraw-Hill Book Company, New York, 1971. (SEARCH-G, TP-G)

Norton, L. (1966), "ADEPT--A Heuristic Program for Proving Theorems of Group Theory," MAC Tech. Report 33, thesis, Project MAC, MIT, Cambridge, MA, 1966. (TP)

Papert, S. (1968), "The Artificial Intelligence of Hubert L. Dreyfus, A Budget of Fallacies," MIT Artificial Intelligence Memo No. 54, Jan. 1968. (GEN)

Papert, Seymour A. (1972), "Teaching Children Thinking," Programmed Learning and Educational Technology, Vol. 9, No. 5, Sept. 1972. (GEN)

Paul, R. C. (1971), "Trajectory Control of a Computer Arm," Proc. 2d Intl. Joint Conf. on Artificial Intelligence, London, England, Sept. 1971. (ROB)

Paul, R. C. (1973), "Modeling, Trajectory Calculation and Servoing of a Computer Controlled Arm," Ph.D. dissertation, Comp. Sci. Dept., Stanford Univ., Stanford, CA, 1973. (ROB)

Paxton, William H. and Robinson, Ann E. (1973), "A Parser for a Speech Understanding System," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (LANG)

Petrick, S. R. (1965), "A Recognition Procedure for Transformational Grammars," Ph.D. thesis, MIT, Cambridge, MA. (LANG)

Petrick, S. R. (1971), "Transformational Analysis," Natural Language Processing, Courant Computer Science Symposium 8, 20-21 Dec. 1971, R. Rustin, ed., 27-41, Algorithmics Press, New York. (LANG)

Petrick, S. R. (1973), "Semantic Interpretation in the Request System," Proc. Intl. Conf. on Computational Linguistics, Pisa, Italy, 1973. (LANG)

Pietrzykowski, T. and Jensen, D. (1972), "A Complete Mechanization of Omega-Order Type Theory," Proc. ACM Natl. Conf., Vol. 1, 82-92, 1972. (TP)

Pingle, K. K. (1969), "Visual Perception by a Computer," Automatic Interpretation and Classification of Images, A. Grasselli, ed., 277-284, Academic Press, New York, London, 1969. (VIS)

Pingle, K. K., Singer, J., and Wichman, W. (1968), "Computer Control of a Mechanical Arm Through Visual Input," Proc. IFIP Cong. 1968, Vol. 2. (ROB)

Pingle, K. K. and Tenenbaum, J. M. (1971), "An Accommodating Edge Follower," IJCAI-2, 1971. (VIS)

Plath, Warren, J. (1973), "Transformational Grammar and Transformational Parsing in the Request System," Proc. Intl. Joint Conf. on Computational Linguistics, Pisa, Italy, 1973. (LANG)

Pohl, I. (1970), "Heuristic Search Viewed as Path Finding in a Graph," Artificial Intelligence, Vol. I, 193-204, 1970. (SEARCH)

Prawitz, D. (1960), "An Improved Proof Procedure," Theoria, Vol. 26, 102-139, 1960. (TP)

Quillian, M. R. (1968), "Semantic Memory," Semantic Information Processing, MIT Press, Cambridge, MA. (REP, PSYC, LANG)

Quillian, Ross (1969), "The Teachable Language Comprehender: A Simulation Program and Theory of Language," Comm. ACM, Vol. 12, 459-476, 1969. (REP, PSYC, LANG)

Raphael, B. (1964a), "A Computer Program Which 'Understands,'" Proc. AFIPS Fall Joint Comp. Conf., 577-589, 1964. (DED, LANG)

Raphael, B. (1964b), "SIR: A Computer Program for Semantic Information Retrieval," MIT, June 1964. Reprinted in Semantic Information Processing, M. Minsky, ed., MIT Press, Cambridge, MA, 1968. (DED, LANG)

Raphael, B. (1968), "Programming A Robot," Proc. IFIP Cong. 68, H135-H140, Edinburgh, 1968. (ROB)

Raphael, B. et al. (1971), "Research and Applications--Artificial Intelligence," Stanford Research Institute Final Report, Contract NASW-2164, National Acronautics and Space Administration, Dec. 1971. (ROB)

Raphael, B. and Green, C. (1968), "The Use of Theorem Proving Techniques in Question Answering Systems," J. ACM, 169, 1968. (DED, REP)

Reboh, R. and Sacerdoti, E. (1973), "A Preliminary QLISP Manual," SRI Artificial Intelligence Center Tech. Note 81, Stanford Research Institute, Menlo Park, CA, Aug. 1973. (SYS)

Reddy, D. R. (1967), "Computer Recognition of Connected Speech," J. ASA, Vol. 42, 329-347. (LANG)

Reddy, D. R. et al. (1973), "The Hearsay Speech Understanding System: An Example of the Recognition Process," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (LANG)

Roberts, L. G. (1963), "Machine Perception of Three-Dimensional Solids," MIT Lincoln Lab., Lexington, MA, Tech. Report No. 315, May 1963. Also in Optical and Electro-Optical Information Processing, J. T. Tippett et al., eds., MIT Press, Cambridge, MA, 1965. (VIS)

Robinson, J. A. (1965), "A Machine-Oriented Logic Based on the Resolution Principle," J. ACM, Vol. 12, No. 1, 23-41, Jan. 1965. (DED, TP)

Robinson, J. A. (1969), "A Note on Mechanizing Higher-Order Logic," Machine Intelligence, Vol. 5, B. Meltzer and D. Michie, eds., 123-133, Edinburgh Univ. Press, Edinburgh, 1969. (TP)

Rosen, C. A. (1972); "Robots, Productivity, and Quality," Proc. ACM Natl. Conf. 1972, Boston, MA, Aug. 14-16, 1972. (ROB-G)

Rosen, C. (1973), "Exploratory Research in Advanced Automation," SRI Report to National Science Foundation, Grant GI-38200X, Stanford Research Institute, Menlo Park, CA, Dec. 1973. (ROB)

Roszak, T. (1972), Where the Wasteland Ends: Politics and Transcendence in Post-Industrial Society, Doubleday, 1972. (GEN)

Rulifson, J. F., Waldinger, R. J., and Derksen, J. A. (1971), "A Language for Writing Problem-Solving Programs," Proc. IFIP, TA-2, 111-115, 1971. (REP, SYS)

Rulifson, J. F., Derksen, J. A., and Waldinger, R. J. (1972), "QA4: A Procedural Calculus for Intuitive Reasoning," Tech. Note 73, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, CA, 1972. (REP, SYS, DED)

Rumelhart, D. E., Lindsay, P. H., and Norman, D. A. (1972), "A Process Model for Long-Term Memory," Organization and Memory, E. Tulving and W. Donaldson, eds., Academic Press, New York, 1972. (PSYC)

Rumelhart, David E. and Norman, Donald A. (1973), "Active Semantic Networks as a Model of Human Memory," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (PSYC)

Russell, R. (1964), "KALAH--The Game and the Program," Stanford Univ. Artificial Intelligence Project Memo No. 22, 3 Sept. 1964. (GAME)

Rustin, R., ed. (1973), Natural Language Processing, Algorithmic Press, New York, 1973. (LANG-G)

Ryder, J. L. (1971), "Heuristic Analysis of Large Trees as Generated in the Game of GO," AI Memo 155, Artificial Intelligence Project, Stanford Univ., Stanford, CA, 1971. (GAME)

Sacerdoti, Earl D. (1973), "Planning in a Hierarchy of Abstraction Spaces," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. To appear in Artificial Intelligence, 1974. (DED)

Sakai, T., Nagao, M., and Knode, T. (1972), "Computer Analysis and Classification of Photographs of Human Faces," First USA-Japan Comp. Conf. Proc., Oct. 1972. (VIS)

Samuel, A. (1959), "Some Studies in Machine Learning Using the Game of CHECKERS," IBM J. Research Develp., Vol. 3, 211-229, 1959. Reprinted in Computers and Thought, E. Feigenbaum and J. Feldman, eds., 71-105, McGraw-Hill Book Company, New York, 1963. (GAME)

Samuel, A. L. (1967), "Some Studies in Machine Learning Using the Game of CHECKERS II--Recent Progress," IBM J. Res. Dev., Vol. 11, 601-617, 1967. (GAME)

Sandewall, E. J. (1971), "Representing Natural Language Information in Predicate Calculus," Machine Intelligence, Vol. 6, B. Meltzer and D. Michie, eds., 255-277, Edinburgh Univ. Press, Edinburgh. (REP)

Sandewall, E. J. (1972a), "Formal Methods in the Design of Question-Answering System," J. Artificial Intelligence, 237, 1972. (REP)

Sandewall, E. J. (1972b), "PCF-2, A First-Order Calculus for Expressing Conceptual Information," Computer Science Report, Comp. Sci. Dept., Uppsala Univ., Uppsala, Sweden. (REP)

Schank, R. C. (1972), "Conceptual Dependency: A Theory of Natural Language Understanding," Cognitive Psychology, Vol. 3, 552-631, 1972. (PSYCH, LANG)

Schank, R. C. (1973), "The Fourteen Primitive Actions and Their Inferences," Stanford AIM-183 Comp. Sci. Dept., Stanford Univ., Stanford, CA, 1973. (REP, LANG, PSYC)

Schank, R. C. et al. (1972), "Primitive Concepts Underlying Verbs of Thought," Stanford AIM-162, Comp. Sci. Dept., Stanford Univ., Stanford, CA, 1972. (REP, LANG, PSYC)

Schank, R. C. and Colby, K. (1973), Computer Models of Thought and Language, W. H. Freeman & Co., San Francisco, 1973. (LANG-G, PSYC-G)

Schank, R. C., Goldman, Neil, Rieger, Charles J., and Riesbeck, Chris (1973), "MARGIE: Memory, Analysis, Response Generation, and Inference on English," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (LANG)

Schank, R. C., Tesler, L., and Weber, S. (1970), "SPINOZA II: Conceptual Case-Based Natural Language Analysis," Memo AIM-109, Stanford Artificial Intelligence Project, Stanford Univ., Stanford, CA, 1970. (LANG)

Scheinman, V. D. (1969), "Design of a Computer-Controlled Manipulator," thesis, Dept. of M.E., Stanford Univ., Stanford, CA. Available as Stanford AIM-92, June 1969. (ROB)

Shannon, C. (1950), "Programming a Digital Computer for Playing Chess," Philosophy Magazine, Vol. 41, 356-375, Mar. 1950. Reprinted in The World of Mathematics, Vol. 4, J. R. Newman, ed., Simon and Schuster, New York, 1954. (GAME)

Shirai, Y. (1972), "A Heterarchical Program for Recognition of Polyhedra," Memo No. 263, Artificial Intelligence Laboratory, MIT, Cambridge, MA. (VIS)

Shirai, Y. (1973), "A Context Sensitive Line Finder for Recognition of Polyhedra," Artificial Intelligence, Vol. 4, No. 2, 95-119, Summer 1973. (VIS)

Shortliffe, E. H. et al. (1973), "An Artificial Intelligence Program to Advise Physicians Regarding Antimicrobial Therapy," Computers and Biomedical Research, Vol. 6, 544-560, 1973. (AIDS)

Simmons, R. F. (1965), "Answering English Questions by Computer: A Survey," Comm. ACM, Vol. 8, 53-70, 1965. (LANG-G)

Simmons, R. F. (1969), "Natural Language Question-Answering Systems: 1969," Comm. ACM, Vol. 13, 15-30, 1970. (LANG-G)

Simmons, Robert F. (1973), "Semantic Networks: Their Computation and Use for Understanding English Sentences," Computer Models of Thought and Language, K. M. Colby and Roger Schank, eds., W. H. Freeman and Company, San Francisco, CA, 1973. (LANG)

Simmons, Robert F., Klein, S., and McConlogue, D. (1964), "Indexing and Dependency Logic for Answering English Questions," American Documentation, Vol. 15, No. 3, 196-204, 1964. (LANG)

Simon, H. A. (1963), "Experiments with a Heuristic Compiler," J. ACM, Vol. 10, No. 4, Oct. 1963. (PROG)

Simon, H. A. (1969), The Sciences of the Artificial, MIT Press, Cambridge, MA. (GEN)

Simon, H. A. (1972), "The Heuristic Compiler," Representation and Meaning, H. A. Simon and L. Siklossy, eds., Prentice-Hall, Englewood Cliffs, NJ, 1972. (PROG)

Simon, H. A. and Feigenbaum, E. A. (1964), "An Information-Processing Theory of Some Effects of Similarity, Familiarization, and Meaningfulness in Verbal Learning," J. Verbal Learning and Verbal Behavior, Vol. 3, 385-396. (PSYC)

Slagle, J. (1961), "A Computer Program for Solving Problems in Freshman Calculus (SAINT)," Lincoln Laboratory Report 5G-001, May 1961. (SEARCH, AIDS)

Slagle, J. R. (1963), "A Heuristic Program that Solves Symbolic Integration Problems in Freshman Calculus," J. ACM, Vol. 10, No. 4, 507-520, Oct. 1963. Also in Computers and Thought, E. Feigenbaum and J. Feldman, eds., 191-203, McGraw-Hill Book Company, New York, 1963. (AIDS, SEARCH)

Slagle, J. (1965), "Experiments with a Deductive Question-Answering Program," Comm. ACM, Vol. 8, 792-798, Dec. 1965. (DED)

Slagle, J. R. (1967), "Automatic Theorem Proving with Renameable and Semantic Resolution," J. ACM, Vol. 14, 687-697, 1967. (TP)

Slagle, J. R. (1971), Artificial Intelligence: The Heuristic Programming Approach, McGraw-Hill Book Company, New York, 1971. (GEN, GAME-G, SEARCH-G)

Slagle, J. R. (1970), "Heuristic Search Programs," Theoretical Approaches to Non-Numerical Problem Solving, R. Banerji and M. I. Mesarovic, eds., 246-273, Springer-Verlag, New York, 1970. (SEARCH)

Slagle, J. R. and Dixon, J. (1969), "Experiments with Some Programs that Search Game Trees," J. ACM, Vol. 16, No. 2, 189-207, Apr. 1969. (GAME, SEARCH)

Solomonoff, R. (1966), "Some Recent Work in Artificial Intelligence," Proc. IEEE, Vol. 54, No. 112, Dec. 1966. (GEN)

Sperling, G. (1960), "The Information Available in Brief Visual Presentations," Psychological Monographs, Vol. 74. (PSYC)

Sridharan, N. S. (1971), "An Application of Artificial Intelligence to Organic Chemical Synthesis," thesis, State Univ. of New York at Stony Brook, New York, July 1971. (AIDS)

Sridharan, N. S. et al. (1973a), "A Heuristic Program to Discover Syntheses for Complex Organic Molecules," Stanford Artificial Intelligence Memo 205, Stanford Univ., Stanford, CA, June 1973. (AIDS)

Sridharan, N. S. (1973b), "Search Strategies for the Task of Organic Chemical Synthesis," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (AIDS)

Sternberg, S. (1966), "High Speed Scanning in Human Memory," Science, Vol. 153, 652-654. (PSYC)

Sussman, G. J. (1973), "A Computational Model of Skill Acquisition," Tech. Note AI TR-297, Artificial Intelligence Laboratory, MIT, Cambridge, MA, Aug. 1973. (DED, PROG)

Sussman, G. J. and McDermott, D. V. (1972), "From PLANNER to CONNIVER--A Genetic Approach," Proc. AFIPS FJCC, Vol. 41, 1171-1180, 1972. (SYS)

Teitelman, W. (1969), "Toward a Programming Laboratory," Proc. 1st Intl. Joint Conf. on Artificial Intelligence, Washington, D.C., 1969. (PROG)

Teitelman, W. (1972a), "Do What I Mean," Computers and Automation, Apr. 1971. (PROG, SYS)

Teitelman, W. (1972b), "Automated Programming--The Programmer's Assistant," Proc. Fall Joint Comp. Conf., Dec. 1972. (PROG, SYS)

Teitelman, W. (1973), "CLISP--Conversational LISP," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (PROG, SYS)

Teitelman, W. (1974), INTERLISP Reference Manual, Xerox and Bolt, Beranek and Newman. Copies available from Xerox Palo Alto Research Center, Palo Alto, CA. (SYS)

Tenenbaum, J. M. (1973), "On Locating Objects by Their Distinguishing Features in Multisensory Images," SRI Artificial Intelligence Center Tech. Note 84, Stanford Research Institute, Menlo Park, CA, Sept. 1973. To appear in Computer Graphics and Image Processing, 1974. (VIS)

Tenenbaum, J. M. et al. (1974), "An Interactive Facility for Scene Analysis Research," SRI Artificial Intelligence Center Tech. Note. 87, Stanford Research Institute, Menlo Park, CA, 1974. (VIS)

Thompson, F. B. (1966), "English for the Computer," Proc. AFIPS 1966 Fall Joint Comp. Conf., Vol. 29, 349-356, Spartan Books, New York. (LANG)

Thompson, F. B. et al. (1969), "REL: A Rapidly Extensible Language System," Proc. 24th Natl. ACM Conf., 1969. (LANG)

Thorne, J., Bratley, P., and Dewar, H. (1968), "The Syntactic Analysis of English by Machine," Machine Intelligence, Vol. 3, D. Michie, ed., American Elsevier Publishing Company, New York, 1968. (LANG)

Tinbergen, N. (1951), The Study of Instinct, Clarendon Press, Oxford, 1951. (PSYC)

Tonge, F. (1961), A Heuristic Program for Assembly Line Balancing, Prentice Hall, 1961. (AIDS)

Turing, A. M. (1949), "Checking a Large Routine," Report of a Conference on High Speech Automatic Calculating-Machines, McLennon Laboratory, Univ. of Toronto, Canada. (PROG)

Turing, A. M. (1950), "Computing Machinery and Intelligence," Mind, Vol. 59, 433-460, Oct. 1950. Reprinted in Computers and Thought, 11-35, E. Feigenbaum and J. Feldman, eds., McGraw-Hill Book Company, New York, 1963. (GEN)

Vicens, P. (1969), "Aspects of Speech Recognition by Computer," Report CS-127, Ph.D. thesis, Comp. Sci. Dept., Stanford Univ., Stanford, CA. (LANG)

Waldinger, R. J. and Lee, R. C. T. (1969), "PROW: A Step Toward Automatic Program Writing," Proc. Intl. Joint Conf. on Artificial Intelligence, 241-252, 1969. (PROG)

Waldinger, R. J. and Levitt, K. N. (1973), "Reasoning About Programs," Tech. Note 86, SRI Artificial Intelligence Center, Oct. 1973, Stanford Research Institute, Menlo Park, CA. To Appear in Artificial Intelligence, 1974. (PROG)

Walker, Donald E., ed. (1964), English Preprocessor Manual, The Mitre Corporation, Bedford, MA, 1964 (SR-132). (LANG)

Walker, Donald E. (1973), "Speech Understanding, Computational Linguistics, and Artificial Intelligence," Tech. Note 85, SRI Artificial Intelligence Center, Stanford Research Institute, Menlo Park, CA, Aug. 1973. (LANG)

Waltz, D. G. (1972), "Generating Semantic Descriptions from Drawings of Scenes with Shadows," AI TR-271, Artificial Intelligence Laboratory, MIT, Aug. 1972. (VIS)

Waterman, D. A. (1970), "Generalization Learning Techniques for Automating the Learning of Heuristics," Artificial Intelligence, Vol. I, Nos. 1 and 2, Spring 1970. (GAME)

Waterman, D. A. and Newell, A. (1971), "Protocol Analysis as a Task for Artificial Intelligence," Artificial Intelligence, Vol. 2, Nos. 2 and 3, 285-318, 1971. (AIDS)

Waterman, D. A. and Newell, A. (1973), "PAS-II: An Interactive Task-Free Version of an Automatic Protocol Analysis System," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (AIDS)

Wegbreit, Ben (1973), "Heuristic Methods for Mechanically Deriving Inductive Assertions," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (PROG)

Weissman, C. (1967), LISP 1.5 Primer, Dickenson Press, 1967. (SYS)

Weizenbaum, J. (1966), "ELIZA--A Computer Program for the Study of Natural Language Communication Between Man and Machine," Comm. ACM, Vol. 9, 36-45, 1966. (LANG)

Weizenbaum, J. (1972), "On the Impact of the Computer on Society," Science, Vol. 176, No. 609, 1972. (GEN)

West, J. D. (1967), "A Heuristic Model for Scheduling Large Projects with Limited Resources," Management Science, Vol. 13B, 359-377. (AIDS)

Winograd, T. (1971), "Procedures as Representation for Data in a Computer Program for Understanding Natural Language," Tech. Report AI TR-17, MIT, Cambridge, MA, 1971. Published as Understanding Natural Language, Academic Press, New York, 1972. (REP, DED, LANG)

Winston, P. H. (1970), "Learning Structural Descriptions from Examples," Tech. Report AI TR-231, Artificial Intelligence Laboratory, MIT, Cambridge, MA, 1970. (REP, VIS)

Winston, P. H. (1972), "The MIT Robot," Machine Intelligence, Vol. 7, 431-463, B. Meltzer and D. Michie, eds., American Elsevier Publishing Company, 1972. (ROB, VIS)

Woods, W. A. (1970), "Transition Network Grammars for Natural Language Analysis," Comm. ACM, Vol. 13, 591-606, 1970. (LANG)

Woods, W. A. (1973), "An Experimental Parsing System for Transition Network Grammars," Natural Language Processing, R. Rustin, ed., 111-154, Algorithmics Press, New York, 1973. (LANG)

Woods, W. A., Kaplan, R. M., Nash-Webber, G. (1972), "The Lunar Science Natural Language Information System: Final Report," BBN Report No. 2378, Bolt, Beranek and Newman, Inc., Cambridge, MA, June 1972. (LANG)

Woods, W. A. and Makhoul, J. (1973), "Mechanical Inference Problems in Continuous Speech Understanding," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (LANG)

Wooldridge, D. (1963), "An Algebraic Simplify Program in LISP," Artificial Intelligence Project Memo No. 11, Stanford Univ., Stanford, CA, Dec. 1963. (AIDS)

Wos, L. T., Carson, D. G., and Robinson, G. A. (1964), "The Unit Preference Strategy in Theorem Proving," Proc. AFIPS, Fall 1964, Vol. 25, 615-621, Spartan Books, Washington, D.C. (TP)

Wos, L. T., Robinson, G., and Carson, D. (1965), "Efficiency and Completeness of the Set of Support Strategy in Theorem-Proving," J. ACM, Vol. 12, No. 4, 536-541, Oct. 1965. (TP)

Yakimovsky, Yoram and Feldman, Jerome A. (1973), "A Semantics-Based Decision Theory Region Analyzer," Adv. Papers 3d Intl. Conf. on Artificial Intelligence, Stanford Univ., Stanford, CA, Aug. 1973. (VIS)

Yates, R., Raphael, B., and Hart, T. (1970), "Resolution Graphs," Artificial Intelligence, Vol. 1, No. 4, 1970. (TP)

Zobrist, A. (1969), "A Model of Visual Organization for the Game of GO," Proc. AFIPS, Spring 1969, 103-112. (GAME)

Zobrist, A. and Carlson, F., Jr. (1973), "An Advice-Taking Chess Computer," Scientific American, June 1973. (GAME)

Zwicky, Arnold M. et al. (1965), "The Mitre Syntactic Analysis Procedure for Transformational Grammars," Proc. AFIPS, Fall 1965, Vol. 27, 317-326. (LANG)

Appendix E

Edward A. Feigenbaum. "The art of artificial intelligence – Themes and case studies of knowledge engineering," pp 227-240, in the *Proceedings of the National Computer Conference – 1978*, copyrighted 1978. Reproduced by permission of AFIPS Press.

Appendix E

The art of artificial intelligence—Themes and case studies of knowledge engineering

by EDWARD A. FEIGENBAUM
Stanford University
Stanford, California

INTRODUCTION—AN EXAMPLE

This paper will examine emerging themes of knowledge engineering, illustrate them with case studies drawn from the work of the Stanford Heuristic Programming Project, and discuss general issues of knowledge engineering art and practice.

Let me begin with an example new to our workbench: a system called PUFF, the early fruit of a collaboration between our project and a group at the Pacific Medical Center (PMC) in San Francisco.*

A physician refers a patient to PMC's pulmonary function testing lab for diagnosis of possible pulmonary function disorder. For one of the tests, the patient inhales and exhales a few times in a tube connected to an instrument/computer combination. The instrument acquires data on flow rates and volumes, the so-called flow-volume loop of the patient's lungs and airways. The computer measures certain parameters of the curve and presents them to the diagnostician (physician or PUFF) for interpretation. The diagnosis is made along these lines: normal or diseased; restricted lung disease or obstructive airways disease or a combination of both; the severity; the likely disease type(s) (e.g., emphysema, bronchitis, etc.); and other factors important for diagnosis.

PUFF is given not only the measured data but also certain items of information from the patient record, e.g., sex, age, number of pack-years of cigarette smoking. The task of the PUFF system is to infer a diagnosis and print it out in English in the normal medical summary form of the interpretation expected by the referring physician.

Everything PUFF knows about pulmonary function diagnosis is contained in (currently) 55 rules of the IF. . . THEN. . . form. No textbook of medicine currently records these rules. They constitute the partly-public, partly-private knowledge of an expert pulmonary physiologist at PMC, and were extracted and polished by project engineers working intensively with the expert over a period of time. Here is an example of a PUFF rule (the unexplained acronyms refer to various data measurements):

RULE 31

IF:
1) The severity of obstructive airways disease of the patient is greater than or equal to mild, and
2) The degree of diffusion defect of the patient is greater than or equal to mild, and
3) The tlc (body box) observed/predicted of the patient is greater than or equal to 110 and
4) The observed-predicted difference in rv/tlc of the patient is greater than or equal to 10

THEN:
1) There is strongly suggestive evidence (.9) that the subtype of obstructive airways disease is emphysema, and
2) It is definite (1.0) that "OAD, Diffusion Defect, elevated TLC, and elevated RV together indicate emphysema." is one of the findings.

One hundred cases, carefully chosen to span the variety of disease states with sufficient exemplary information for each, were used to extract the 55 rules. As the knowledge emerged, it was represented in rule form, added to the system and tested by running additional cases. The expert was sometimes surprised, sometimes frustrated, by the occasional gaps and inconsistencies in the knowledge, and the incorrect diagnoses that were logical consequences of the existing rule set. The interplay between knowledge engineer and expert gradually expanded the set of rules to remove most of these problems.

As cumulation of techniques in the art demands and allows, a new tool was not invented when an old one would do. The knowledge engineers pulled out of their toolkit a version of the MYCIN system (to be discussed later), with the rules about infectious diseases removed, and used it as the inference engine for the PUFF diagnoses. Thus PUFF, like MYCIN, is a relatively simple backward-chaining infer-

* Dr. J. Osborn, Dr. R. Fallat, John Kunz, Diane McClung.

ence system. It seeks a valid line-of-reasoning based on its rules and rooted in the instrument and patient data. With a little more work at fitting some existing tools together, PUFF will be able to explain this line-of-reasoning, just as MYCIN does.

As it is, PUFF only prints out the final interpretation, of which the following is an example:

PATIENT DATA:
The degree of dyspnea: MODERATELY-SEVERE
The severity of coughing: MILD
Sputum production MODERATELY-SEVERE
The number of pack-years of smoking: 48
referral diagnosis: BRONCHITIS
IVC/IVC-predicted: 80
RV/RV-predicted: 191
FVC/FVC-predicted: 87
TLC(body box)observed/predicted: 127
Predicted FEV1/FVC: 83
TLC(DLCO)observed/predicted: 83
FEV1/FVC ratio: 50
RV/TLC Observed-Predicted: 21
MMF/MMF-predicted: 19
the DLCO/DLCO-predicted: 48
The slope (F50_obs-F25_obs)/FVC_obs: 19

DEGREE OF OBSTRUCTIVE AIRWAYS DISEASE:

OAD degree by SLOPE: (MODERATELY-SEVERE 700)
OAD degree by MMF: (SEVERE 900)
OAD degree by FEV1: (MODERATELY-SEVERE 700)
FINAL OAD DEGREE: (MODERATELY-SEVERE 910) (SEVERE 900)
No conflict. Final degree:
(MODERATELY-SEVERE 910)

INTERPRETATION:

Obstruction is indicated by curvature of the flow-volume loop.
Forced Vital Capacity is normal and peak flow rates are reduced, suggesting airway obstruction.
Flow rate from 25-75 of expired volume is reduced, indicating severe airway obstruction.
OAD, Diffusion Defect, elevated TLC, and elevated RV together indicate emphysema.
OAD, Diffusion Defect, and elevated RV indicate emphysema.
Change in expired flow rates following bronchodilation shows that there is reversibility of airway obstruction.
The presence of a productive cough is an indication that the OAD is of the bronchitic type.
Elevated lung volumes indicate overinflation.
Air trapping is indicated by the elevated difference between observed and predicted RV/TLC ratios.
Improvement in airway resistance indicates some reversibility of airway
Airway obstruction is consistent with the patient's smoking history.
The airway obstruction accounts for the patient's dyspnea.
Although bronchodilators were not useful in this one case, prolonged use may prove to be beneficial to the patient.
The reduced diffusion capacity indicates airway obstruction of the mixed bronchitic and emphysematous types.
Low diffusing capacity indicates loss of alveolar capillary surface.
Obstructive Airways Disease of mixed types

150 cases not studied during the knowledge acquisition process were used for a test and validation of the rule set. PUFF inferred a diagnosis for each. PUFF-produced and expert-produced interpretations were coded for statistical analysis to discover the degree of agreement. Over various types of disease states, and for two conditions of match between human and computer diagnoses ("same degree of severity" and "within one degree of severity"), agreement ranged between approximately 90 percent and 100 percent.

The PUFF story is just beginning and will be told perhaps at a later NCC. The surprising punchline to my synopsis is that the current state of the PUFF system as described above was achieved in less than 50 hours of interaction with the expert and less than 10 man-weeks of effort by the knowledge engineers. We have learned much in the past decade of the art of engineering knowledge-based intelligent agents!

In the remainder of this essay, I would like to discuss the route that one research group, the Stanford Heuristic Programming Project, has taken, illustrating progress with case studies, and discussing themes of the work.

ARTIFICIAL INTELLIGENCE & KNOWLEDGE ENGINEERING

The dichotomy that was used to classify the collected papers in the volume *Computers and Thought* still characterizes well the motivations and research efforts of the AI community. First, there are some who work toward the construction of intelligent artifacts, or seek to uncover principles, methods, and techniques useful in such construction. Second, there are those who view artificial intelligence as (to use Newell's phrase) "theoretical psychology," seeking explicit and valid information processing models of human thought.

For purposes of this essay, I wish to focus on the motivations of the first group, these days by far the larger of the two. I label these motivations "the intelligent agent viewpoint" and here is my understanding of that viewpoint:

"The potential uses of computers by people to accom-

plish tasks can be 'one-dimensionalized' into a spectrum representing the nature of instruction that must be given the computer to do its job. Call it the WHAT-to-HOW spectrum. At one extreme of the spectrum, the user supplies his intelligence to instruct the machine with precision exactly HOW to do his job, step-by-step. Progress in Computer Science can be seen as steps away from the extreme 'HOW' point on the spectrum: the familiar panoply of assembly languages, subroutine libraries, compilers, extensible languages, etc. At the other extreme of the spectrum is the user with his real problem (WHAT he wishes the computer, as his instrument, to do for him). He aspires to communicate WHAT he wants done in a language that is comfortable to him (perhaps English); via communication modes that are convenient for him (including perhaps, speech or pictures); with some generality, some vagueness, imprecision, even error; without having to lay out in detail all necessary subgoals for adequate performance—with reasonable assurance that he is addressing an intelligent agent that is using knowledge of his world to understand his intent, to fill in his vagueness, to make specific his abstractions, to correct his errors, to discover appropriate subgoals, and ultimately to translate WHAT he really wants done into processing steps that define HOW it shall be done by a real computer. The research activity aimed at creating computer programs that act as "intelligent agents" near the WHAT end of the WHAT-To-HOW spectrum can be viewed as the long-range goal of AI research." (Feigenbaum, 1974)

Our young science is still more art than science. Art: "the principles or methods governing any craft or branch of learning." Art: "skilled workmanship, execution, or agency." These the dictionary teaches us. Knuth tells us that the endeavor of computer programming is an art, in just these ways. The art of constructing intelligent agents is both part of and an extension of the programming art. It is the art of building complex computer programs that represent and reason with knowledge of the world. Our art therefore lives in symbiosis with the other worldly arts, whose practitioners—experts of their art—hold the knowledge we need to construct intelligent agents. In most "crafts or branches of learning" what we call "expertise" is the essence of the art. And for the domains of knowledge that we touch with our art, it is the "rules of expertise" or the rules of "good judgment" of the expert practitioners of that domain that we seek to transfer to our programs.

Lessons of the past

Two insights from previous work are pertinent to this essay.

The first concerns the quest for generality and power of the inference engine used in the performance of intelligent acts (what Minsky and Papert [see Goldstein and Papert, 1977] have labeled "the power strategy"). We must hypothesize from our experience to date that the problem solving power exhibited in an intelligent agent's performance is primarily a consequence of the specialist's knowledge employed by the agent, and only very secondarily related to the generality and power of the inference method employed. Our agents must be knowledge-rich, even if they are methods-poor. In 1970, reporting the first major summary-of-results of the DENDRAL program (to be discussed later), we addressed this issue as follows:

> ". . . general problem-solvers are too weak to be used as the basis for building high-performance systems. The behavior of the best general problem-solvers we know, human problem-solvers, is observed to be weak and shallow, except in the areas in which the human problem-solver is a specialist. And it is observed that the transfer of expertise between specialty areas is slight. A chess master is unlikely to be an expert algebraist or an expert mass spectrum analyst, etc. In this view, the expert is the specialist, with a specialist's knowledge of his area and a specialist's methods and heuristics." (Feigenbaum, Buchanan and Lederberg, 1971, p. 187)

Subsequent evidence from our laboratory and all others has only confirmed this belief.

AI researchers have dramatically shifted their view on generality and power in the past decade. In 1967, the canonical question about the DENDRAL program was: "It sounds like good chemistry, but what does it have to do with AI?" In 1977, Goldstein and Papert write of a paradigm shift in AI:

> "Today there has been a shift in paradigm. The fundamental problem of understanding intelligence is not the identification of a few powerful techniques, but rather the question of how to represent large amounts of knowledge in a fashion that permits their effective use and interaction." (Goldstein and Papert, 1977).

The second insight from past work concerns the nature of the knowledge that an expert brings to the performance of a task. Experience has shown us that this knowledge is largely heuristic knowledge, experiential, uncertain—mostly "good guesses" and "good practice," in lieu of facts and rigor. Experience has also taught us that much of this knowledge is private to the expert, not because he is unwilling to share publicly how he performs, but because he is unable. He knows more than he is aware of knowing. [Why else is the Ph.D. or the Internship a guild-like apprenticeship to a presumed "master of the craft?" What the masters really know is not written in the textbooks of the masters.] But we have learned also that this private knowledge can be uncovered by the careful, painstaking analysis of a second party, or sometimes by the expert himself, operating in the context of a large number of highly specific performance problems. Finally, we have learned that expertise is multifaceted, that the expert brings to bear many and varied sources of knowledge in performance. The approach to capturing his expertise must proceed on many fronts simultaneously.

The knowledge engineer

The knowledge engineer is that second party just discussed. She works intensively with an expert to acquire domain-specific knowledge and organize it for use by a program. Simultaneously she is matching the tools of the AI workbench to the task at hand—program organizations, methods of symbolic inference, techniques for the structuring of symbolic information, and the like. If the tool fits, or nearly fits, she uses it. If not, necessity mothers AI invention, and a new tool gets created. She builds the early versions of the intelligent agent, guided always by her intent that the program eventually achieve expert levels of performance in the task. She refines or reconceptualizes the system as the increasing amount of acquired knowledge causes the AI tool to "break" or slow down intolerably. She also refines the human interface to the intelligent agent with several aims: to make the system appear "comfortable" to the human user in his linguistic transactions with it; to make the system's inference processes understandable to the user; and to make the assistance controllable by the user when, in the context of a real problem, he has an insight that previously was not elicited and therefore not incorporated.

In the next section, I wish to explore (in summary form) some case studies of the knowledge engineer's art.

CASES FROM THE KNOWLEDGE ENGINEER'S WORKSHOP

I will draw material for this section from the work of my group at Stanford. Much exciting work in knowledge engineering is going on elsewhere. Since my intent is not to survey literature but to illustrate themes, at the risk of appearing parochial I have used as case studies the work I know best.

My collaborators (Professors Lederberg and Buchanan) and I began a series of projects, initially the development of the DENDRAL program, in 1965. We had dual motives: first, to study scientific problem solving and discovery, particularly the processes scientists do use or should use in inferring hypotheses and theories from empirical evidence; and second, to conduct this study in such a way that our experimental programs would one day be of use to working scientists, providing intelligent assistance on important and difficult problems. By 1970, we and our co-workers had gained enough experience that we felt comfortable in laying out a program of research encompassing work on theory formation, knowledge utilization, knowledge acquisition, explanation, and knowledge engineering techniques. Although there were some surprises along the way, the general lines of the research are proceeding as envisioned.

THEMES

As a road map to these case studies, it is useful to keep in mind certain major themes:

Generation-and-test: Omnipresent in our experiments is the "classical" generation-and-test framework that has been the hallmark of AI programs for two decades. This is not a consequence of a doctrinaire attitude on our part about heuristic search, but rather of the usefulness and sufficiency of the concept.

Situation⇒Action Rules: We have chosen to represent the knowledge of experts in this form. Making no doctrinaire claims for the universal applicability of this representation, we nonetheless point to the demonstrated utility of the rule-based representation. From this representation flow rather directly many of the characteristics of our programs: for example, ease of modification of the knowledge, ease of explanation. The essence of our approach is that a rule must capture a "chunk" of domain knowledge that is meaningful, in and of itself, to the domain specialist. Thus our rules bear only a historical relationship to the production rules used by Newell and Simon (1972) which we view as "machine-language programming" of a recognize⇒act machine.

The Domain-Specific Knowledge: It plays a critical role in organizing and constraining search. The theme is that in the knowledge is the power. The interesting action arises from the knowledge base, not the inference engine. We use knowledge in rule form (discussed above), in the form of inferentially-rich models based on theory, and in the form of tableaus of symbolic data and relationships (i.e., frame-like structures). System processes are made to conform to natural and convenient representations of the domain-specific knowledge.

Flexibility to modify the knowledge base: If the so-called "grain size" of the knowledge representation is chosen properly (i.e., small enough to be comprehensible but large enough to be meaningful to the domain specialist), then the rule-based approach allows great flexibility for adding, removing, or changing knowledge in the system.

Line-of-reasoning: A central organizing principle in the design of knowledge-based intelligent agents is the maintenance of a line-of-reasoning that is comprehensible to the domain specialist. This principle is, of course, not a logical necessity, but seems to us to be an engineering principle of major importance.

Multiple Sources of Knowledge: The formation and maintenance (support) of the line-of-reasoning usually require the integration of many disparate sources of knowledge. The representational and inferential problems in achieving a smooth and effective integration are formidable engineering problems.

Explanation: The ability to explain the line-of-reasoning in a language convenient to the user is necessary for application and for system development (e.g., for debugging and for extending the knowledge base). Once again, this is an engineering principle, but very important. What con-

stitutes. "an explanation" is not a simple concept, and considerable thought needs to be given, in each case, to the structuring of explanations.

CASE STUDIES

In this section I will try to illustrate these themes with various case studies.

DENDRAL: Inferring chemical structures

Historical note

Begun in 1965, this collaborative project with the Stanford Mass Spectrometry Laboratory has become one of the longest-lived continuous efforts in the history of AI (a fact that in no small way has contributed to its success). The basic framework of generation-and-test and rule-based representation has proved rugged and extendable. For us the DENDRAL system has been a fountain of ideas, many of which have found their way, highly metamorphosed, into our other projects. For example, our long-standing commitment to rule-based representations arose out of our (successful) attempt to head off the imminent ossification of DENDRAL caused by the rapid accumulation of new knowledge in the system around 1967.

Task

To enumerate plausible structures (atom-bond graphs) for organic molecules, given two kinds of information: analytic instrument data from a mass spectrometer and a nuclear magnetic resonance spectrometer; and user-supplied constraints on the answers, derived from any other source of knowledge (instrumental or contextual) available to the user.

Representations

Chemical structures are represented as node-link graphs of atoms (nodes) and bonds (links). Constraints on search are represented as subgraphs (atomic configurations) to be denied or preferred. The empirical theory of mass spectrometry is represented by a set of rules of the general form:

Situation: Particular atomic configuration (subgraph)
↓
Probability, P, of occurring
↓
Action: Fragmentation of the particular configuration (Breaking links)

Rules of this form are natural and expressive to mass spectrometrists.

Sketch of method

DENDRAL's inference procedure is a heuristic search that takes place in three stages, without feedback: plan-generate-test.

"Generate" (a program called CONGEN) is a generation process for plausible structures. Its foundation is a combinatorial algorithm (with mathematically proven properties of completeness and non-redundant generation) that can produce all the topologically legal candidate structures. Constraints supplied by the user or by the "Plan" process prune and steer the generation to produce the plausible set (i.e., those satisfying the constraints) and not the enormous legal set.

"Test" refines the evaluation of plausibility, discarding less worthy candidates and rank-ordering the remainder for examination by the user. "Test" first produces a "predicted" set of instrument data for each plausible candidate, using the rules described. It then evaluates the worth of each candidate by comparing its predicted data with the actual input data. The evaluation is based on heuristic criteria of goodness-of-fit. Thus, "test" selects the "best" explanations of the data.

"Plan" produces direct (i.e., not chained) inference about likely substructure in the molecule from patterns in the data that are indicative of the presence of the substructure. (Patterns in the data trigger the left-hand-sides of substructure rules). Though composed of many atoms whose interconnections are given, the substructure can be manipulated as atom-like by "generate." Aggregating many units entering into a combinatorial process into fewer higher-level units reduces the size of the combinatorial search space. "Plan" sets up the search space so as to be relevant to the input data. "Generate is the inference tactician; "Plan" is the inference strategist. There is a separate "Plan" package for each type of instrument data, but each package passes substructures (subgraphs) to "Generate." Thus, there is a uniform interface between "Plan" and "Generate." User-supplied constraints enter this interface, directly or from user-assist packages, in the form of substructures.

Sources of knowledge

The various sources of knowledge used by the DENDRAL system are:

Valences (legal connections of atoms); stable and unstable configurations of atoms; rules for mass spectrometry fragmentations; rules for NMR shifts; experts' rules for planning and evaluation; user-supplied constraints (contextual).

Results

DENDRAL's structure elucidation abilities are, paradoxically, both very general and very narrow. In general, DENDRAL handles all molecules, cyclic and tree-like. In pure structure elucidation under constraints (without instrument data), CONGEN is unrivaled by human performance. In structure elucidation with instrument data, DENDRAL's performance rivals expert human performance only for a small number of molecular families for which the program has been given specialist's knowledge, namely the families of interest to our chemist collaborators. I will spare this computer science audience the list of names of these families. Within these areas of knowledge-intensive specialization, DENDRAL's performance is usually not only much faster but also more accurate than expert human performance.

The statement just made summarizes thousands of runs of DENDRAL on problems of interest to our experts, their colleagues, and their students. The results obtained, along with the knowledge that had to be given to DENDRAL to obtain them, are published in major journals of chemistry. To date, 25 papers have been published there, under a series title "Applications of Artificial Intelligence for Chemical Inference: (specific subject)" (see for example, the Buchanan, Smith, et al., 1976, reference).

The DENDRAL system is in everyday use by Stanford chemists, their collaborators at other universities and collaborating or otherwise interested chemists in industry. Users outside Stanford access the system over commercial computer/communications network. The problems they are solving are often difficult and novel. The British government is currently supporting work at Edinburgh aimed at transferring DENDRAL to industrial user communities in the UK.

Discussion

Representation and extensibility. The representation chosen for the molecules, constraints, and rules of instrument data interpretation is sufficiently close to that used by chemists in thinking about structure elucidation that the knowledge base has been extended smoothly and easily, mostly by chemists themselves in recent years. Only one major reprogramming effort took place in the last 9 years—when a new generator was created to deal with cyclic structures.

Representation and the Integration of multiple sources of knowledge. The generally difficult problem of integrating various sources of knowledge has been made easy in DENDRAL by careful engineering of the representations of objects, constraints, and rules. We insisted on a common language of compatibility of the representations with each other and with the inference processes: the language of molecular structure expressed as graphs. This leads to a straightforward procedure for adding a new source of knowledge, say, for example, the knowledge associated with a new type of instrument data. The procedure is this: write rules that describe the effect of the physical processes of the instrument on molecules using the situation⇒action form with molecular graphs on both sides; any special inference process using these rules must pass its results to the generator only (!) in the common graph language.

It is today widely believed in AI that the use of many diverse sources of knowledge in problem solving and data interpretation has a strong effect on quality of performance. How strong is, of course, domain-dependent, but the impact of bringing just one additional source of knowledge to bear on a problem can be startling. In one difficult (but not unusually difficult) mass spectrum analysis problem,* the program using its mass spectrometry knowledge alone would have generated an impossibly large set of plausible candidates (over 1.25 million!). Our engineering response to this was to add another source of data and knowledge, proton NMR. The addition on a simple interpretive theory of this NMR data, from which the program could infer a few additional constraints, reduced the set of plausible candidates to one, the right structure! This was not an isolated result but showed up dozens of times in subsequent analyses.

DENDRAL and data. DENDRAL's robust models (topological, chemical, instrumental) permit a strategy of finding solutions by generating hypothetical "correct answers" and choosing among these with critical tests. This strategy is opposite to that of piecing together the implications of each data point to form a hypothesis. We call DENDRAL's strategy largely model-driven, and the other data-driven. The consequence of having enough knowledge to do model-driven analysis is a large reduction in the amount of data that must be examined since data is being used mostly for verification of possible answers. In a typical DENDRAL mass spectrum analysis, usually no more than about 25 data points out of a typical total of 250 points are processed. This important point about data reduction and focus-of-attention has been discussed before by Gregory (1968) and by the vision and speech research groups, but is not widely understood.

Conclusion. DENDRAL was an early herald of AI's shift to the knowledge-based paradigm. It demonstrated the point of the primacy of domain-specific knowledge in achieving expert levels of performance. Its development brought to the surface important problems of knowledge representation, acquisition, and use. It showed that, by and large, the AI tools of the first decade were sufficient to cope with the demands of a complex scientific problem-solving task, or were readily extended to handle unforeseen difficulties. It demonstrated that AI's conceptual and programming tools were capable of producing programs of applications interest, albeit in narrow specialties. Such a demonstration of competence and sufficiency was important for the credibility of the AI field at a critical juncture in its history.

META-DENDRAL: inferring rules of mass spectrometry

Historical note

The META-DENDRAL program is a case study in automatic acquisition of domain knowledge. It arose out of our

* The analysis of an acyclic amine with formula $C_{20}H_{45}N$.

DENDRAL work for two reasons: first, a decision that with DENDRAL we had a sufficiently firm foundation on which to pursue our long-standing interest in processes of scientific theory formation; second, by a recognition that the acquisition of domain knowledge was the bottleneck problem in the building of applications-oriented intelligent agents.

Task

META-DENDRAL's job is to infer rules of fragmentation of molecules in a mass spectrometer for possible later use by the DENDRAL performance program. The inference is to be made from actual spectra recorded from known molecular structures. The output of the system is the set of fragmentation rules discovered, summary of the evidence supporting each rule, and a summary of contra-indicating evidence. User-supplied constraints can also be input to force the form of rules along desired lines.

Representations

The rules are, of course, of the same form as used by DENDRAL that was described earlier.

Sketch of method

META-DENDRAL, like DENDRAL, uses the generation-and-test framework. The process is organized in three stages: Reinterpret the data and summarize evidence (INTSUM); generate plausible candidates for rules (RULEGEN); test and refine the set of plausible rules (RULEMOD).

INTSUM: gives every data point in every spectrum an interpretation as a possible (highly specific) fragmentation. It then summarizes statistically the "weight of evidence" for fragmentations and for atomic configurations that cause these fragmentations. Thus, the job of INTSUM is to translate data to DENDRAL subgraphs and bond-breaks, and to summarize the evidence accordingly.

RULEGEN: conducts a heuristic search of the space of all rules that are legal under the DENDRAL rule syntax and the user-supplied constraints. It searches for plausible rules, i.e., those for which positive evidence exists. A search path is pruned when there is no evidence for rules of the class just generated. The search tree begins with the (single) most general rule (loosely put, "anything" fragments from "anything") and proceeds level-by-level toward more detailed specifications of the "anything." The heuristic stopping criterion measures whether a rule being generated has become too specific, in particular whether it is applicable to too few molecules of the input set. Similarly there is a criterion for deciding whether an emerging rule is too general. Thus, the output of RULEGEN is a set of candidate rules for which there is positive evidence.

RULEMOD: tests the candidate rule set using more complex criteria, including the presence of negative evidence. It removes redundancies in the candidate rule set; merges rules that are supported by the same evidence; tries further specialization of candidates to remove negative evidence; and tries further generalization that preserves positive evidence.

Results

META-DENDRAL produces rule sets that rival in quality those produced by our collaborating experts. In some tests, META-DENDRAL re-created rule sets that we had previously acquired from our experts during the DENDRAL project. In a more stringent test involving members of a family of complex ringed molecules for which the mass spectral theory had not been completely worked out by chemists, META-DENDRAL discovered rule sets for each subfamily. The rules were judged by experts to be excellent and a paper describing them was recently published in a major chemical journal (Buchanan, Smith, et al, 1976).

In a test of the generality of the approach, a version of the META-DENDRAL program is currently being applied to the discovery of rules for the analysis of nuclear magnetic resonance data.

MYCIN and TEIRESIAS: Medical diagnosis

Historical note

MYCIN originated in the Ph.D. thesis of E. Shortliffe (now Shortliffe, M.D. as well), in collaboration with the Infectious Disease group at the Stanford Medical School (Shortliffe, 1976). TEIRESIAS, the Ph.D. thesis work of R. Davis, arose from issues and problems indicated by the MYCIN project but generalized by Davis beyond the bounds of medical diagnosis applications (Davis, 1976). Other MYCIN-related theses are in progress.

Tasks

The MYCIN performance task is diagnosis of blood infections and meningitis infections and the recommendation of drug treatment. MYCIN conducts a consultation (in English) with a physician-user about a patient case, constructing lines-of-reasoning leading to the diagnosis and treatment plan.

The TEIRESIAS knowledge acquisition task can be described as follows:

In the context of a particular consultation, confront the expert with a diagnosis with which he does not agree. Lead him systematically back through the line-of-reasoning that produced the diagnosis to the point at which he indicates the analysis went awry. Interact with the expert to modify offending rules or to acquire new rules. Rerun the consultation to test the solution and gain the expert's concurrence.

Representations:

MYCIN's rules are of the form:

IF (conjunctive clauses) THEN (implication)

Here is an example of a MYCIN rule for blood infections.

RULE 85

IF:
1) The site of the culture is blood, and
2) The gram stain of the organism is gramneg, and
3) The morphology of the organism is rod, and
4) The patient is a compromised host

THEN:
There is suggestive evidence (.6) that the identity of the organism is pseudomonas-aeruginosa

TEIRESIAS allows the representation of MYCIN-like rules governing the use of other rules, i.e., rule-based strategies. An example follows.

METARULE 2

IF:
1) the patient is a compromised host, and
2) there are rules which mention in their premise pseudomonas
3) there are rules which mention in their premise klebsiellas

THEN:
There is suggestive evidence (.4) that the former should be done before the latter.

Sketch of method

MYCIN employs a generation-and-test procedure of a familiar sort. The generation of steps in the line-of-reasoning is accomplished by backward chaining of the rules. An IF-side clause is either immediately true or false (as determined by patient or test data entered by the physician in the consultation); or is to be decided by subgoaling. Thus, "test" is interleaved with "generation" and serves to prune out incorrect lines-of-reasoning.

Each rule supplied by an expert has associated with it a "degree of certainty" representing the expert's confidence in the validity of the rule (a number from 1 to 10). MYCIN uses a particular ad-hoc but simple model of inexact reasoning to cumulate the degrees of certainty of the rules used in an inference chain (Shortliffe and Buchanan, 1975).

It follows that there may be a number of "somewhat true" lines-of-reasoning—some indicating one diagnosis, some indicating another. All (above a threshold) are used by the system as sources of knowledge indicating plausible lines-of-reasoning.

TEIRESIAS' rule acquisition process is based on a record of MYCIN's search. Rule acquisition is guided by a set of rule models that dictate the form and indicate the likely content of new rules. Rule models are not given in advance, but are inferred from the knowledge base of existing rules.

Language of interaction

The language used looks like it might be English but is actually the dialect "Doctor-ese" used by members of the tribe of healing arts practitioners. Reasonably simple language processing methods suffice. When ambiguities of interpretation are encountered, they are fed back to the user for decisions about meaning.

Example diagnosis and treatment plan

The following is an example of a diagnosis reported by MYCIN to one of its users following a consultation. An example consultation will not be shown, but one can be found in Shortliffe's book (Shortliffe, 1976).

My therapy recommendations will be designed to treat for organisms that are either very likely or, although less likely, would have a significant effect on therapy selection if they were present. It is important to cover for the following probable infection(s) and associated organism(s):

INFECTION-1 is CYSTITIS
(Item 1) PSEUDOMONAS-CEPACIA [ORGANISM-5]
(Item 2) CITROBACTER-DIVERSUS [ORGANISM-4]
(Item 3) E.COLI [ORGANISM-3]

INFECTION-2 is UPPER-RESPIRATORY-INFECTION
(Item 4) LISTERIA [ORGANISM-2]

INFECTION-3 is BACTEREMIA
(Item 5) ENTEROCOCCUS [ORGANISM-1]

[REC-1] My preferred therapy recommendation is as follows:

In order to cover for Items (1 2 3 4 5):
Give the following in combination:
1) KANAMYCIN
Dose: 750 mg (7.5 mg/kg) q12h IM (or IV) for 28 days
Comments: Modify dose in renal failure
2) PENICILLIN
Dose: 2,500,000 units (25000 units/kg) q4h IV for 28 days

Results

No clinical evaluation of MYCIN is currently being done. In a preliminary evaluation, done outside of the clinical setting, a panel of expert judges was asked to rate MYCIN's performance along a number of dimensions (e.g., infecting organism identification, organism significance). In 90 percent of the cases submitted to the judges, a majority of the judges said that the program's decisions were the-same-as or as-good-as the decisions they would have made.

Explanation

MYCIN can expose its line of reasoning by offering explanations of various kinds. "Why" and "How" questions can be asked by the user during the consultation. "Why" is the user's way of inquiring, "Why do you want to know that?" "How" is his way of asking, "How did you conclude that. . . ?" In a post-consultation, he can similarly inquire about how the diagnostic conclusions were reached, and inquire about the reasons why certain other conclusions were ruled out. Examples of the MYCIN-TEIRESIAS explanation capabilities follow:

MYCIN: Is the patient's illness with ORGANISM-1 a hospital-acquired infection?
USER: WHY
TEIRESIAS: (i.e., WHY is it important to determine whether or not the infection with ORGANISM-1 was acquired while the patient was hospitalized?)

[1.0] This will aid in determining the category of ORGANISM-1.

It has already been established that

[1.1] the gram stain of ORGANISM-1 is gramneg, and
[1.2] the morphology of ORGANISM-1 is rod, and
[1.3] the aerobicity of ORGANISM-1 is facultative

Therefore, if

[1.4] the infection with ORGANISM-1 was not acquired while the patient was hospitalized

then

there is strongly suggestive evidence (.8) that the category of ORGANISM-1 is enterobacteriacese

Also: there is weakly suggestive evidence (.1) that the identity of ORGANISM-1 is pseudomonas-aeruginosa [RULE 37]

USER: HOW 1.3

TEIRESIAS: [i.e. HOW was it established that the aerobicity of ORGANISM-1 is facultative?]

The following were used:

[3.1] RULE 27 indicated there is strongly suggestive evidence (.8) that the aerobicity of ORGANISM-1 is facultative

[3.2] RULE 27 indicated there is weakly suggestive evidence (.2) that the aerobicity of ORGANISM-1 is anaerobic

Since this gave a cumulative CF of (.8) for facultative, and (.2) for anaerobic, it has been established that the aerobicity of ORGANISM-1 is facultative.

USER: WHY DIDN'T YOU CONSIDER STREPTOCOCCUS AS A POSSIBILITY FOR ORGANISM-1?

TEIRESIAS: The following rule could have been used to determine that the identity of ORGANISM-1 was streptococcus: RULE 33

But clause 2 ("the morphology of the organism is coccus") was already known to be false for ORGANISM-1, so the rule was never tried.

Discussion

A few short comments aimed at knowledge engineering are in order.

English is the language most natural and comfortable for our experts and users, and was therefore the language chosen for interactive consultation, explanation, and external representation of the rules (the internal format is INTERLISP). This situation is not peculiar to doctors: in most areas of application of intelligent agents I believe that English (i.e., natural language) will be the language of choice. Programming an English language processor and front-end to such systems is not a scary enterprise because:

(a) the domain is specialized, so that possible interpretations are constrained.

(b) specialist-talk is replete with standard jargon and stereotyped ways of expressing knowledge and queries—just right for text templates, simple grammars and other simple processing schemes.

(c) the ambiguity of interpretation resulting from simple schemes can be dealt with easily by feeding back interpretations for confirmation. If this is done with a pleasant "I didn't quite understand you. . ." tone, it is not irritating to the user.

English may be exactly the wrong language for representation and interaction in some domains. It would be awkward, to say the least, to represent DENDRAL's chemical structures and knowledge of mass spectrometry in English, or to interact about these with a user.

Simple explanation schemes have been a part of the AI scene for a number of years and are not hard to implement. Really good models of what explanation is as a transaction between user and agent, with programs to implement these models, will be the subject (I predict) of much future research in AI.

Without the explanation capability, I assert, user acceptance of MYCIN would have been nil, and there would have been a greatly diminished effectiveness and contribution of our experts.

MYCIN was the first of our programs that forced us to deal with what we had always understood: that experts' knowledge is uncertain and that our inference engines had to be made to reason with this uncertainty. It is less important that the inexact reasoning scheme be formal, rigorous, and uniform than it is for the scheme to be natural to and easily understandable by the experts and users.

All of these points can be summarized by saying that MYCIN and its TEIRESIAS adjunct are experiments in the design of a see-through system, whose representations and processes are almost transparently clear to the domain specialist. "Almost" here is equivalent to "with a few minutes of introductory description." The various pieces of MYCIN—the backward chaining, the English transactions, the explanations, etc.—are each simple in concept and realization. But there are great virtues to simplicity in system design: and viewed as a total intelligent agent system, MYCIN/TEIRESIAS is one of the best engineered.

SU/X: signal understanding

Historical note

SU/X is a system design that was tested in an application whose details are classified. Because of this, the ensuing discussion will appear considerably less concrete and tangible than the preceding case studies. This system design was done by H. P. Nii and me, and was strongly influenced by the CMU Hearsay II system design (Lesser and Erman, 1977).

Task

SU/X's task is the formation and continual updating, over long periods of time, of hypotheses about the identity, location, and velocity of objects in a physical space. The output desired is a display of the "current best hypotheses" with full explanation of the support for each. There are two types of input data: the primary signal (to be understood); and auxiliary symbolic data (to supply context for the understanding). The primary signals are spectra, represented as descriptions of the spectral lines. The various spectra cover the physical space with some spatial overlap.

Representations

The rules given by the expert about objects, their behavior, and the interpretation of signal data from them are all represented in the situation⇒action form. The "situations" constitute invoking conditions and the "actions" are processes that modify the current hypotheses, post unresolved issues, recompute evaluations, etc. The expert's knowledge of how to do analysis in the task is also represented in rule form. These strategy rules replace the normal executive program.

The situation-hypothesis is represented as a node-link graph, tree-like in that it has distinct "levels," each representing a degree of abstraction (or aggregation) that is natural to the expert in his understanding of the domain. A node represents an hypothesis; a link to that node represents support for that hypothesis (as in HEARSAY II, "support from above" or "support from below"). "Lower" levels are concerned with the specifics of the signal data. "Higher" levels represent symbolic abstractions.

Sketch of method

The situation-hypothesis is formed incrementally. As the situation unfolds over time, the triggering of rules modifies or discards existing hypotheses, adds new ones, or changes support values. The situation-hypothesis is a common workspace ("blackboard," in HEARSAY jargon) for all the rules.

In general, the incremental steps toward a more complete and refined situation-hypothesis can be viewed as a sequence of local generate-and-test activities. Some of the rules are plausible move generators, generating either nodes or links. Other rules are evaluators, testing and modifying node descriptions.

In typical operation, new data is submitted for processing (say, N time-units of new data). This initiates a flurry of rule-triggerings and consequently rule-actions (called "events"). Some events are direct consequences of data; other events arise in a cascade-like fashion from the triggering of rules. Auxiliary symbolic data also cause events, usually affecting the higher levels of the hypothesis. As a consequence, support-from-above for the lower level processes is made available; and expectations of possible lower level events can be formed. Eventually all the relevant rules have their say and the system becomes quiescent, thereby triggering the input of new data to reenergize the inference activity.

The system uses the simplifying strategy of maintaining only one "best" situation-hypothesis at any moment, modifying it incrementally as required by the changing data. This

approach is made feasible by several characteristics of the domain. First, there is the strong continuity over time of objects and their behaviors (specifically, they do not change radically over time, or behave radically differently over short periods). Second, a single problem (identity, location and velocity of a particular set of objects) persists over numerous data gathering periods. (Compare this to speech understanding in which each sentence is spoken just once, and each presents a new and different problem.) Finally, the system's hypothesis is typically "almost right," in part because it gets numerous opportunities to refine the solution (i.e., the numerous data gathering periods), and in part because the availability of many knowledge sources tends to over-determine the solution. As a result of all of these, the current best hypothesis changes only slowly with time, and hence keeping only the current best is a feasible approach.

Of interest are the time-based events. These rule-like expressions, created by certain rules, trigger upon the passage of specified amounts of time. They implement various "wait-and-see" strategies of analysis that are useful in the domain.

Results

In the test application, using signal data generated by a simulation program because real data was not available, the program achieved expert levels of performance over a span of test problems. Some problems were difficult because there was very little primary signal to support inference. Others were difficult because too much signal induced a plethora of alternatives with much ambiguity.

A modified SU/X design is currently being used as the basis for an application to the interpretation of x-ray crystallographic data, the CRYSALIS program mentioned later.

Discussion

The role of the auxiliary symbolic sources of data is of critical importance. They supply a symbolic model of the existing situation that is used to generate expectations of events to be observed in the data stream. This allows flow of inferences from higher levels of abstraction to lower. Such a process, so familiar to AI researchers, apparently is almost unrecognized among signal processing engineers. In the application task, the expectation-driven analysis is essential in controlling the combinatorial processing explosion at the lower levels, exactly the explosion that forces the traditional signal processing engineers to seek out the largest possible number-cruncher for their work.

The design of appropriate explanations for the user takes an interesting twist in SU/X. The situation-hypothesis unfolds piecemeal over time, but the "appropriate" explanation for the user is one that focuses on individual objects over time. Thus the appropriate explanation must be synthesized from a history of all the events that led up to the current hypothesis. Contrast this with the MYCIN-TEIRESIAS reporting of rule invocations in the construction of a reasoning chain.

Since its knowledge base and its auxiliary symbolic data give it a model-of-the-situation that strongly constrains interpretation of the primary data stream, SU/X is relatively unperturbed by errorful or missing data. These data conditions merely cause fluctuations in the credibility of individual hypotheses and/or the creation of the "wait-and-see" events. SU/X can be (but has not yet been) used to control sensors. Since its rules specify what types and values of evidence are necessary to establish support, and since it is constantly processing a complete hypothesis structure, it can request "critical readings" from the sensors. In general, this allows an efficient use of limited sensor bandwidth and data acquisition processing capability.

Other case studies

Space does not allow more than just a brief sketch of other interesting projects that have been completed or are in progress.

AM: mathematical discovery

AM is a knowledge-based system that conjectures interesting concepts in elementary mathematics. It is a discoverer of interesting theorems to prove, not a theorem proving program. It was conceived and executed by D. Lenat for his Ph.D. thesis, and is reported by him in these proceedings.

AM's knowledge is basically of two types: rules that suggest possibly interesting new concepts from previously conjectured concepts; and rules that evaluate the mathematical "interestingness" of a conjecture. These rules attempt to capture the expertise of the professional mathematician at the task of mathematical discovery. Though Lenat is not a professional mathematician, he was able successfully to serve as his own expert in the building of this program.

AM conducts a heuristic search through the space of concepts creatable from its rules. Its basic framework is generation-and-test. The generation is plausible move generation, as indicated by the rules for formation of new concepts. The test is the evaluation of "interestingness." Of particular note is the method of test-by-example that lends the flavor of scientific hypothesis testing to the enterprise of mathematical discovery.

Initialized with concepts of elementary set theory, it conjectured concepts in elementary number theory, such as "add," "multiply" (by four distinct paths!), "primes," the unique factorization theorem, and a concept similar to primes but previously not much studied called "maximally divisible numbers."

MOLGEN: planning experiments in molecular genetics

MOLGEN, a collaboration with the Stanford Genetics Department, is work in progress. MOLGEN's task is to

provide intelligent advice to a molecular geneticist on the planning of experiments involving the manipulation of DNA. The geneticist has various kinds of laboratory techniques available for changing DNA material (cuts, joins, insertions, deletions, and so on); techniques for determining the biological consequences of the changes; various instruments for measuring effects; various chemical methods for inducing, facilitating, or inhibiting changes; and many other tools.

Some MOLGEN programs under development will offer planning assistance in organizing and sequencing such tools to accomplish an experimental goal. Other MOLGEN programs will check user-provided experiment plans for feasibility; and its knowledge base will be a repository for the rapidly expanding knowledge of this specialty, available by interrogation.

In MOLGEN the problem of integration of many diverse sources of knowledge is central since the essence of the experiment planning process is the successful merging of biological, genetic, chemical, topological, and instrument knowledge. In MOLGEN the problem of representing processes is also brought into focus since the expert's knowledge of experimental strategies—proto-plans—must also be represented and put to use.

One MOLGEN program (Stefik, 1978) solves a type of analysis problem that is often difficult for laboratory scientists to solve. DNA structures can be fragmented by chemicals called restriction enzymes. These enzymes cut DNA at specific recognition sites. The fragmentation may be complete or partial. One or more enzymes may be used. The fragmented segments of the DNA are collected and sorted out by segment length using a technique called gel electrophoresis. The analytical problem is similar to that faced by DENDRAL: given an observed fragmentation pattern, hypothesize the best structural explanation of the data. More precisely the problem is to map the enzyme recognition sites of a DNA structure from complete or partial "digests".

The program uses the model-driven approach that is similar to DENDRAL's and is discussed earlier. The method is generate-and-test. A generator is initiated that is capable of generating all the site-segment maps in an exhaustive, irredundant fashion. Various pruning rules are used to remove whole classes of conceivable candidates in light of the data. Some of the pruning rules are empirical and judgmental. Others are formal and mathematically based.

The program solves simpler problems of this type of analysis better than laboratory scientists. The harder problems, however, yield only to the broader biological knowledge known by the scientists and not yet available to the program's reasoning processes. In a recent test case, a problem whose solution space contained approximately 150,000,000 site-fragment "maps" was solved in 27 seconds of PDP-10 time using the INTERLISP programming system.

Interestingly, the computer scientist's formal understanding of the nature of the problem, his formal representation of the knowledge used for pruning out inappropriate candidates, and the computational power available to him enabled him to suggest a few new experiment designs to his geneticist collaborators that were not previously in their repertoire.

CRYSALIS: inferring protein structure from electron density maps

CRYSALIS, too, is work in progress. Its task is to hypothesize the structure of a protein from a map of electron density that is derived from x-ray crystallographic data. The map is three-dimensional, and the contour information is crude and highly ambiguous. Interpretation is guided and supported by auxiliary information, of which the amino acid sequence of the protein's backbone is the most important. Density map interpretation is a protein chemist's art. As always, capturing this art in heuristic rules and putting it to use with an inference engine is the project's goal.

The inference engine for CRYSALIS is a modification of the SU/X system design described above. The hypothesis formation process must deal with many levels of possibly useful aggregation and abstraction. For example, the map itself can be viewed as consisting of "peaks," or "peaks and valleys," or "skeleton." The protein model has "atoms," "amide planes," "amino acid sidechains," and even massive substructures such as "helices." Protein molecules are so complex that a systematic generation-and-test strategy like DENDRAL's is not feasible. Incremental piecing together of the hypothesis using region-growing methods is necessary.

The CRYSALIS design (alias SU/P) is described in a recent paper by Nii and Feigenbaum (1977).

SUMMARY OF CASE STUDIES

Some of the themes presented earlier need no recapitulation, but I wish to revisit three here: generation-and-test; situation⇒action rules; and explanations.

Generation and test

Aircraft come in a wide variety of sizes, shapes, and functional designs and they are applied in very many ways. But almost all that fly do so because of the unifying physical principle of lift by airflow; the others are described by exception. If there is such a unifying principle for intelligent programs and human intelligence it is generation-and-test. No wonder that this has been so thoroughly studied in AI research!

In the case studies, generation is manifested in a variety of forms and processing schemes. There are legal move generators defined formally by a generating algorithm (DENDRAL's graph generating algorithm); or by a logical rule of inference (MYCIN's backward chaining). When legal move generation is not possible or not efficient, there are plausible move generators (as in SU/X and AM). Sometimes generation is interleaved with testing (as in MYCIN, SU/X, and AM). In one case, all generation precedes testing (DENDRAL). One case (META-DENDRAL) is mixed, with some testing taking place during generation, some after.

Test also shows great variety. There are simple tests (MYCIN: "Is the organism aerobic?"; SU/X: "Has a spectral line appeared at position P?") Some tests are complex heuristic evaluations (AM: "Is the new concept 'interesting'?"; MOLGEN: "Will the reaction actually take place?") Sometimes a complex test can involve feedback to modify the object being tested (as in META-DENDRAL).

The evidence from our case studies supports the assertion by Newell and Simon that generation-and-test is a law of our science (Newell and Simon, 1976).

Situation⇒action rules

Situation⇒Action rules are used to represent experts' knowledge in all of the case studies. Always the situation part indicates the specific conditions under which the rule is relevant. The action part can be simple (MYCIN: conclude presence of particular organism; DENDRAL: conclude break of particular bond). Or it can be quite complex (MOLGEN: an experiential procedure). The overriding consideration in making design choices is that the rule form chosen be able to represent clearly and directly what the expert wishes to express about the domain. As illustrated, this may necessitate a wide variation in rule syntax and semantics.

From a study of all the projects, a regularity emerges. A salient feature of the Situation⇒Action rule technique for representing experts' knowledge is the modularity of the knowledge base, with the concomitant flexibility to add or change the knowledge easily as the experts' understanding of the domain changes. Here too one must be pragmatic, not doctrinaire. A technique such as this cannot represent modularity of knowledge if that modularity does not exist in the domain. The virtue of this technique is that it serves as a framework for discovering what modularity exists in the domain. Discovery may feed back to cause reformulation of the knowledge toward greater modularity.

Finally, our case studies have shown that strategy knowledge can be captured in rule form. In TEIRESIAS, the metarules capture knowledge of how to deploy domain knowledge; in SU/X, the strategy rules represent the experts' knowledge of "how to analyze" in the domain.

Explanation

Most of the programs, and all of the more recent ones, make available an explanation capability for the user, be he end-user or system developer. Our focus on end-users in applications domains has forced attention to human engineering issues, in particular making the need for the explanation capability imperative.

The Intelligent Agent viewpoint seems to us to demand that the agent be able to explain its activity; else the question arises of who is in control of the agent's activity. The issue is not academic or philosophical. It is an engineering issue that has arisen in medical and military applications of intelligent agents, and will govern future acceptance of AI work in applications areas. And on the philosophical level one might even argue that there is a moral imperative to provide accurate explanations to end-users whose intuitions about our systems are almost nil.

Finally, the explanation capability is needed as part of the concerted attack on the knowledge acquisition problem. Explanation of the reasoning process is central to the interactive transfer of expertise to the knowledge base, and it is our most powerful tool for the debugging of the knowledge base.

ACKNOWLEDGMENT

The work reported herein has received long-term support from the Defense Advanced Research Projects Agency (DAHC 15-73-C-0435). The National Institutes of Health (5R24-RR00612, RR-00785) has supported DENDRAL, META-DENDRAL, and the SUMEX-AIM computer facility on which we compute. The National Science Foundation (MCS 76-11649, DCR 74-23461) has supported research on CRYSALIS and MOLGEN. The Bureau of Health Sciences Research and Evaluation (HS-10544) has supported research on MYCIN. I am grateful to these agencies for their continuing support of our work.

I wish to express my deep admiration and thanks to the faculty, staff and students of the Heuristic Programming Project, and to our collaborators in the various worldly arts, for the creativity and dedication that has made our work exciting and fruitful.

REFERENCES

General

Feigenbaum, E. A. "Artificial Intelligence Research: What is it? What has it achieved? Where is it going?," invited paper, *Symposium on Artificial Intelligence*, Canberra, Australia, 1974.

Feigenbaum, E. A. and J. Feldman, *Computers and Thought*, New York, McGraw-Hill, 1963.

Goldstein, I. and S. Papert, "Artificial Intelligence, Language, and the Study of Knowledge," *Cognitive Science*, Vol. 1, No. 1, 1977.

Gregory, R., "On How so Little Information Controls so Much Behavior," Bionics Research Report No. 1, Machine Intelligence Department, University of Edinburgh, 1968.

Lesser, V. R. and L. D. Erman, "A Retrospective View of the HEARSAY-II Architecture," *Proceedings of the Fifth International Artificial Intelligence-1977*, Massachusetts Institute of Technology, Cambridge, Massachusetts, August 22-25, 1977, Vol. 1.

Newell, A. and H. A. Simon, *Human Problem Solving*, Prentice-Hall, 1972.

Newell, A. and H. A. Simon, "Computer Science as Empirical Inquiry: Symbols and Search," *Com ACM*, 19, 3, March, 1976.

DENDRAL and META-DENDRAL

Feigenbaum, E. A., B. G. Buchanan, and J. Lederberg, "On Generality and Problem Solving: a Case Study Using the DENDRAL Program," *Machine Intelligence 6*, Edinburgh Univ. Press, 1971.

Buchanan, B. G., A. M. Duffield, and A. V. Robertson, "An Application of Artificial Intelligence to the Interpretation of Mass Spectra," *Mass Spectrometry Techniques and Applications*, G. W. A. Milne, Ed., John Wiley & Sons, Inc., p. 121, 1971.

Michie, D. and B. G. Buchanan, "Current Status of the Heuristic DENDRAL Program for Applying Artificial Intelligence to the Interpretation of Mass Spectra," *Computers for Spectroscopy*, R. A. G. Carrington, ed., London: Adam Hilger, 1974.

Buchanan, B. G., "Scientific Theory Formation by Computer," Nato Advanced Study Institutes Series, Series E: Applied Science, 14:515, Noordhoff-Leyden, 1976.

Buchanan, B. G., D. H. Smith, W. C. White, R. J. Gritter, E. A. Feigenbaum, J. Lederberg, and C. Djerassi, "Applications of Artificial Intelligence for Chemical Inference XXII. Automatic Rule Formation in Mass Spectrometry by Means of the Meta-DENDRAL Program," *Journal of the ACS*, 98:6168, 1976.

MYCIN

Shortliffe, E. *Computer-based Medical Consultations: MYCIN*, New York, Elsevier, 1976.

Davis, R., B. G. Buchanan, and E. H. Shortliffe, "Production Rules as a Representation for a Knowledge-Based Consultation Program," *Artificial Intelligence*, 8, 1, February, 1977.

Shortliffe, E. H. and B. G. Buchanan, "A Model of Inexact Reasoning in Medicine," *Mathematical Biosciences*, 23:351, 1975.

TEIRESIAS

Davis, R., "Applications of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases," Memo HPP-76-7, Stanford Computer Science Department, Stanford, CA, 1976.

Davis, R., "Interactive Transfer of Expertise I: Acquisition of New Inference Rules," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence-1977*, Massachusetts Institute of Technology, Cambridge, Massachusetts, August 22-25, 1977.

Davis, R. and B. G. Buchanan, "Meta-Level Knowledge: Overview and Applications," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence-1977*, Massachusetts Institute of Technology, Cambridge, Massachusetts, August 22-25, 1977.

SUIX

Nii, H. P. and E. A. Feigenbaum, "Rule Based Understanding of Signals," *Proceedings of the Conference on Pattern-Directed Inference Systems*, 1978 (forthcoming), also Memo HPP-77-7, Stanford Computer Science Department, Stanford, CA, 1977.

AM

Lenat, D., "AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search," Memo HPP-76-8, Stanford Computer Science Department, Stanford, CA, 1976.

MOLGEN

Martin, N., P. Friedland, J. King, and M. Stefik, "Knowledge Base Management for Experiment Planning in Molecular Genetics," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence-1977*, Massachusetts Institute of Technology, Cambridge, Massachusetts, August 22-25, 1977.

Stefik, M., "Inferring DNA Structures from Segmentation Data," *Artificial Intelligence*, 1978 (in press).

CRYSALIS

Engelmore, R. and H. P. Nii, "A Knowledge-Based System for the Interpretation of Protein X-Ray Crystallographic Data," Memo HPP-77-2, Department of Computer Science, Stanford, CA, 1977.

Appendix F

Allen Newell. "Remarks on the relationship between artificial intelligence and cognitive psychology," pp 363-400, in *Theoretical Approaches to Non-Numerical Problem Solving*, Part IV, edited by R. Banerji and M. D. Mesarovic, copyrighted 1970. Reprinted by permission of Springer-Verlag, New York.

Appendix F

Remarks on the relationship between artificial intelligence and cognitive psychology.*

Allen Newell
Carnegie-Mellon University
Pittsburgh, Pennsylvania
1970

This research was supported by Research Grant MH-07732 from the National Institutes of Health.

*This paper was published in R. Banerji and M. D. Mesarovic (eds.) Theoretical Approaches to Non-Numerical Problem Solving, Part IV, pp 363-400, Springer-Verlag, New York, 1970.
I would like to thank Richard Young for his comments on this paper.

REMARKS ON THE RELATIONSHIP BETWEEN ARTIFICIAL INTELLIGENCE AND COGNITIVE PSYCHOLOGY

Allen Newell

Carnegie-Mellon University
Pittsburgh, Pennsylvania

1. INTRODUCTION

Shortly after I agreed to participate in this conference, I received a letter from a psychologist friend, who had been working in the area of cognitive simulation. He had become discouraged, feeling that less and less work was going on. He felt that attempts to simulate cognitive functioning were a dead end and he was leaving the field. He wanted to let me know.

Now, my own impression is that matters stand rather well in the use of information processing models in psychology. The dissonance between this letter and my own view led to considerable reflection over the next several months. This seems an appropriate occasion to pass on these reflections. Thus, I wish to address myself to the relationship between artificial intelligence and cognitive psychology. I will not provide here any survey of the research being done. Nor will I be reporting any new research (though in fact some of the odd pieces I will mention are fairly recent).

Furthermore, these are reflections on the relationship. I shall not attempt any systematic argument. For that would be, in effect, to argue the necessity of my own world view -- my own Weltanshauung. And I agree with the substance of Churchman's paper in this conference, that one cannot argue such things directly.

Let me set the stage by two preliminaries, before moving to the points themselves.

2. THE POSSIBLE RELATIONSHIPS

We list in Figure 1 a number of possibilities that cover the range of relationships that might exist between artificial intelligence and psychology. The list moves roughly from weak to strong relationship as one moves from top to bottom.

Thus, right at the top, there may be no relationship at all between artificial intelligence and psychology. This is certainly a possible view:

> Since the theory rests on analogies between the human and the mechanical process, Newell et al take some pains to produce comparisons between human problem solving and the behavior of the machine. In this effort [LT] they draw upon previously published descriptions of relevant human behavior. They add nothing to our further understanding of the living mechanisms, but they do provide a better understanding of the computer. (T. Kendler, 1961, pp. 451-452.)

The next stage is where one feels that artificial intelligence provides metaphors, thus making psychologists attend to new phenomena in appropriate ways. This view is the interpretation many scientists put on cybernetics through the forties and fifties. And many people hold it about artificial intelligence now:

> Psychology and the study of artificial intelligence are both concerned with intelligent behavior, but otherwise they are not necessarily related except to the extent that metaphors borrowed from one discipline may be stimulating to the other. (A.G. Oettinger, 1969, p. 30.)

No relationship
Metaphor / Attention focussing
Forces operationality
Provides language
Provides base (ideal) models
Sufficiency analysis
Theoretical psychology
Self sufficient

Figure 1: Possible Relationships between AI and Psychology

The next step of engagement is that emphasis on programs and mechanisms forces the psychologist to become operational, that is, to avoid the fuzziness of using mentalistic terms. It is a sort of mental hygiene. Behaviorism is in part a similar sort of mental hygiene, but one that achieves its effect by remaining in the observation language of the experiment (i.e., the behaviors that can be observed). Artificial intelligence offers an operationalism with respect to theory. This view has been very popular, as the following quotations testify:

> The advantage of playing this kind of game lies solely in the fact that, if you talk about machines, you are more certain to leave out the subjective, anthropomorphic hocus-pocus of mentalism. ...
>
> There is still a further step possible along this same road: the design and construction of actual robots who perform different human functions as well or better than a man can do. ... The only use that lies in designing an actual robot is to make sure that, in stating the properties of a function, we have not left in unwittingly some mystic ambiguous mentalistic term. (E. Boring, 1946, p. 191.)
>
> ... On the other hand, the computer program allows us to specify with complete precision, complex models that certainly embody what we are vaguely point to with these words. We can then, as with the concepts "active memory" and "learning" briefly discussed here, study our models to get a better idea of what we have been talking about.
>
> The computer is just a powerful tool for clearly specifying rules that mechanisms must follow in carrying out procedures that process information. (L. Uhr, 1969, p. 297.)

The next stage sees the language as the major connection: The language of programs and data structures (e.g., list structures) is the appropriate vehicle for describing the behavior of humans, in contradistinction, say, to classical mathematics. An analogous view was strongly held a decade ago in arguing that for the social sciences the appropriate mathematics was that of finite structures (matrix analysis, markov processes, graph theory), as opposed to the mathematics of the continuum (i.e., differential equations). Perhaps, the clearest statements of the language view with respect to artificial intelligence have been made by George Miller:

> The computer program can play a double role in psychology: as a model of an intelligent system and, even more broadly, as a kind of language in which theories can be expressed. Everyone recognizes the importance of holding a good theory; the advantages of speaking a good language, however, are not so often recognized. (p. 9)

> There is much that the psychologist can learn from a study of computing machines and the structure of their programs. Programming languages seem to offer an excellent medium for the expression of psychologicial theories, even though using such languages implies that men and machines are in some deep sense considered to be equivalent -- functionally, if not structurally. (G. Miller, 1962, p. 21.)

The stages of metaphor, operationality and language are somehow content free. That is, the gains to psychology are in various behaviors and disciplines of the psychologist. The next stage finally accords the product of the artificial intelligence models significance, even if not their content. Here artificial intelligence is used to provide base lines against which to view actual behavior. These base lines are in the direction of optimum behavior, rather than in the direction of random behavior as in the base lines usually provided for by statistics). Such ideal types are used fruitfully in several places in science. In psychology a good

example is the work of Ward Edwards on behavior in uncertain situations, where humans are consistently conservative compared to the optimal solution, as computed from Bayes theorem. Without this comparison with an ideal system, a significant aspect of the data would be missed. In artificial intelligence this view is perhaps less common than might be suspected, given that computers are programmed to do the best job possible. Nevertheless, one finds the attitude expressed occasionally:

> The computer analogues used in some of the model of human information processing and thought depict ideal intellectual slaves, experiencing practically no time lag, no loss of memory, and no reluctance to consider all of the available evidence. The human to whom our formulations are meant to apply do unfortunately experience considerable limitations in these regards. (W.J. McGuire, 1968, p. 159).

The next turn of the screw reflects a unique feature of human cognitive behaviors, namely that they constitute performances for which often we do not know <u>any</u> way that they can be accomplised. Thus, it becomes of interest to discover systems that perform these tasks. If, in addition, no mechanisms are used in these systems that clearly go beyond the capacities of the human, then an initial theory has been provided. This level has been called <u>suffjciency analysis</u>, since it seeks to show that a sufficient set of mechanisms exists for a particular intellectual task. To illustrate, if one develops a chess program that examines 800,000 positions in deciding on a move, then one has not made a contribution; since excellent evidence exists that no human could consider 800,000 separate items of information in ten minutes. But if the chess program only considers around 100 positions, and if there are no other ways in which the program radically violates the general character of human processing capacities, then it may be taken as a first model. An example of this view is the following:

> The definitions are both nominal and ostensive in the sense that when we speak, for example, of "pathogenic conflict" we can

> point to a precise procedure in the program which computes whether two beliefs are in conflict or not. We must postpone the question, which eventually must be faced, of how closely this corresponds to the nature of pathogenic conflict in real persons. But at this point we can say there is a rough match between the output of the program and typical behavior of patients in psychotherapeutic sessions. (K.M. Colby and J.P. Gilbert, 1964, p. 417)

This view has a certain value in itself, since psychology has in general ignored the question of explaining how it is that humans can perform the acts of intelligence they routinely accomplish. Thus, it adds a new mode of analysis.

With the next turn, we get artificial intelligence as theoretical psychology. This is analogous to the view of the mathematics of differential equations as theoretical physics. Thus the actual theories of cognitive psychology are to be expressed as artificial intelligence systems. We would expect to find artificial intelligence systems of direct empirical relevance, and also artificial intelligence systems being developed for their own sake, just as in mathematics there is concern with the differential equations of physical interest (e.g., the Mathieu equation) and also the pure theory of differential equations. This view has been often expressed; for instance:

> Quite typically, these models express psychological propositions in terms of individual operations for matching, generating, transforming, and retrieving information. These operations are knit together to form systems of complexly organized structures and processes. Since the structures and processes are represented explicitly, such models enable us to go behond measures of the quantifiable and statistical properties of behavior to investigations of the specific sequences of stimuli and responses involved. ... By comparing model-generated behavior with data from humans, we can decide unambiguously

> whether the model is sufficient to account for the phenomena we are investigating. Concerned as they are with the microstructure of behavior, information-processing psychologists often prefer to work with extensive sequential data from individual subjects. (W. Reitman, 1969, p. 246.)

There is yet one more twist -- a radical one, but not totally implausible. One can view artificial intelligence as sufficient within itself for the entire task of understanding the nature of human intelligence. Thus, the behavioral data now being gathered and analyzed in psychological laboratories are taken to be irrelevant. With our long standing involvement in an empiricist view of science, this may seem like nonsense. But consider that the constraints on intelligent behavior in our world may be such that there exists in essence, only one type of system that can accomplish it. Then we might be able to discover that system by direct analysis, knowing only the nature of the world (the organism's task environment) and the general kinds of performances of which it is capable. The plausibility of this can be enhanced considerably if two conditions are added. First, the basic system itself must have arisen by evolution. Second, the system must be able to develop from a basic system (capabilities unknown, but fundamentally simple) to one with full intelligence. There are few who subscribe to this viewpoint totally. However a hint can be found in the following quotation:

> Nor is it true that psychologists take the experimental evidence into account but that others [engineers working on pattern recognition] do not, for it is not clear that much really firm evidence has been collected, except for a few scattered findings, chiefly from neurophysiology. As horrifying as it may sound to some, the chief sources of specification of a model for pattern recognition are intuition and introspection, and in this we all draw upon our own resources as human beings. Since these are two functions that have made twentieth century psychology

> especially uneasy, there is no reason to think that psychologists are terribly adept at them. (L. Uhr, 1966, p. 291.)

I have laid out this array of viewpoints to locate myself and the nature of my comments. I wish to focus on the strong end -- namely, on artificial intelligence as theoretical psychology. (I do not, however, go to the last stage.) Thus, I am much concerned with the use of artificial intelligence systems as theories for detailed and explicit bodies of data on human cognitive behavior.

The literature that talks about simulation of cognitive processes speaks mostly from views down toward the weak end, as I have tried to indicate with the quotations. While I think that artificial intelligence can be relevant to psychology in all of these ways, I have always felt that quoting them smacked a bit of damning with faint praise. If it is not possible to do the real job -- i.e., to be theory in the full sense -- then one must settle for the advantages that do exist.† (To be fair to those who have espoused these various advantages -- including myself -- clarity about the role of a new development is achieved only slowly.)

3. WHAT IS ARTIFICIAL INTELLIGENCE?

The second preliminary is to fix what I mean by artificial intelligence for the purpose of this paper. As shown in Figure 2 there is a very large encompassing domain labeled variously cybernetic systems, information processing systems, control systems, etc. -- this entire familiar interrelated scientific and technological domain that has arisen since World War II. One major subdomain is that of symbolic systems, which is pretty much coterminous with the systems of interest to computer science. Symbolic systems are to be distinguished from discrete systems, as the control theorist uses that term, in having symbols that have referential structure. Programming and linguistic systems would be another set of names for the same area.

† Psychology itself has a nice example. One often hears that a good theory is one that leads to good new experiments. While true, this virtue often has to serve in the absence of more substantial advantages, such as predictive and explanatory power.

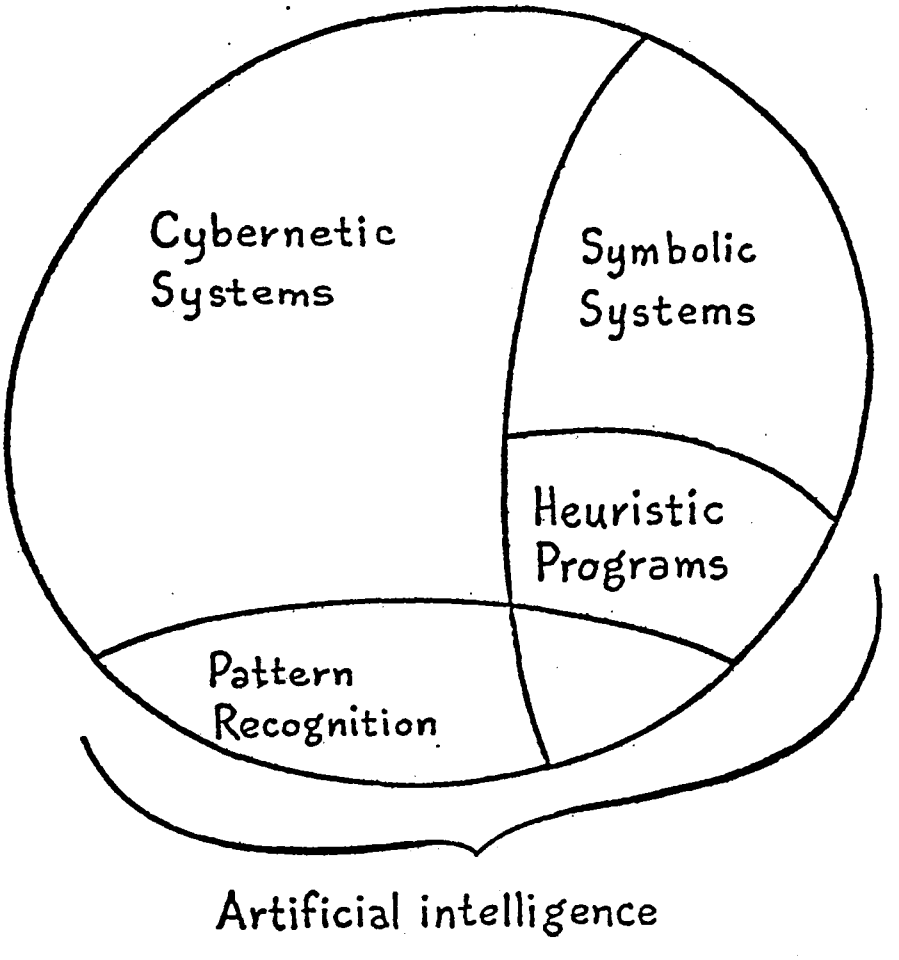


FIGURE 2: Cybernetic Systems and its Subdomains.

Within symbolic systems there is a subdomain called heuristic programming, e.g., programs for problem solving, theorem proving, game playing, induction, etc. This is part of artificial intelligence, as the term is commonly used. There are also other parts of artificial intelligence, such as pattern recognition. Some pattern recognition systems are symbolic, e.g., the work of Uhr (1961). But other pattern recognition systems are discrete, though not symbolic (e.g., neural nets), and some are not even discrete (e.g., holographic systems).

With Figure 2 as background, then, when I refer to artificial intelligence I will mean heuristic programming -- that is, symbolic systems for performing intellectual functions. I will exclude such areas as pattern recognition -- not because they are any less important, but because they are a different story for a different time.

More important, I wish to broaden my concern from artificial intelligence to the whole of symbolic systems. For the right question to ask is not about the relation of psychology to artificial intelligence systems, but about the relation of psychology to symbolic systems. In fact, this larger view already has a name -- it is called information processing psychology. It is to be distinguished from the flurry within psychology some years ago on the use of information theory, as developed by Shannon (e.g., see Attneave, 1959). Information processing psychology is concerned essentially with whether a successful theory of human behavior can be found within the domain of symbolic systems.

The reason for the expansion is clear if you view the matter from psychology's vantage point, which wants to construct theories to describe and explain human behavior. Symbolic systems provide a possible class of systems within which such theories might be formed. Some of the behaviors of interest are primarily problem solving -- e.g., a man playing a game of chess. But much behavior of interest is not intellectually demanding -- e.g., learning new information, interpreting a command in natural language, retrieving a relevant fact. But these tasks are also susceptible to an analysis in terms of symbolic systems and information processing. Thus, artificial intelligence covers only a part of the relevant systems.

I am insisting on the importance of the general type of system used to form specific theories of human behavior -- in our case, symbolic systems. It is, then, worthwhile to note that psychology has searched for its theories mostly in terms of classes of systems other than symbolic systems. Behaviorism is in general coupled with a view of systems of stimulus and response associations. Gestalt psychology is coupled with a view of continuous fields which reorganize themselves. Psychoanalytic theory is framed in terms of energy constructs, with conservation laws a major organizing feature. All of these views -- and the three of them account for a large fraction of psychological theory -- are quite distinct from symbolic systems.

This emphasis on the substantive content of information processing models is in sharp contradistinction to the neutrality of computer simulation per se. This latter has been emphasized by many people. It can be seen in the earlier quote of Uhr in connection on operationality. Here is another:

> I should like to conclude with this final comment: My insistence that a theoretical formulation be rendered in such a manner that it could be converted into a computer program does not in itself predispose us toward any particular type of theory. ... The model resides wholly in the program supplied to the computer and not at all in the hardware of the computer itself. For this reason any model can be programmed -- provided only that it is sufficiently explicit. (Shepard, 1963, p. 67.)

My own insistence does not conflict with the above statement. Rather, it reflects an additional product of the growth of computer science, namely, that of a theoretical model of symbolic behavior. After the fact, one can see that such a theory might have emerged within psychology (or linguistics) without the advent of the computer. In historical fact, the theory emerged by trying to program the computer to do non-numerical tasks and by trying to construct abstract theories of computation and logic.

With this background, let me now make a series of points.

4. POINT ONE: PENETRATION INTO EXPERIMENTAL PSYCHOLOGY

The first point is that the penetration of information processing theories into experimental psychology is very substantial. To see this, one must take the broader view I have just emphasized. Information processing, not artificial intelligence, is the critical issue, simply because most tasks investigated in psychology are not problem solving or complex learning.

Furthermore, the total range of work that now operates within an information processing framework by no means derives from a single source. More precisely, the wider domain, which we labeled cybernetic systems in Figure 2, has been the common source of all the work (especially if we understand it to include developments in operational mathematics, such as decision theory and game theory). But this broad development has permitted many parallel developments in psychology, all converging on the class of information processing systems. Let me briefly identify these main lines of development.

Perhaps the most important one in terms of number of investigators is that concerned with the study of immediate memory. In terms familiar to this audience, the basic problem is to discover the logical design of the short term memory. Actually, there appear to be several such memories, some of the order of hundreds of milliseconds half life, at least one of the order of several seconds. Since no anatomical or physiological data exist on these memories, their existence and characteristics must be inferred entirely from behavior. Thus, there is even controversy over what memories exist (Melton, 1962).

Now the concern with the logical design of a system does not necessarily imply concern with a symbolic system. And, indeed, the genesis of this work goes back to communications engineering and information theory. The book by Broadbent on Perception and Communication (1958), which was one of the milestones in this area, shows this very well: signal processing, but not symbol processing.

What changed this was the discovery that the human immediate memory appears to hold symbols -- chunks, to use the term introduced by George Miller in his well-known paper on the magic number seven (Miller, 1956). This established that one

should consider the human as an information processing system with a short term memory of constant capacity, measured in number of symbols. By now, this view permeates all work, as can be seen in the numerous models of short term memory that are now available (many of them summarized in Norman (1969)).

A second development is in psycholinguistics, where the work of Chomsky has had a very large impact. First, observe that Chomskian linguistics implies a symbolic system. One can emphasize, as have the linguistics, that performance should be distinguished from competence, so that a model of the linguistic ability (i.e., the set of syntactical rules) does not imply that language is in fact processed in a person by a machine that takes the rule system as input. However, if one wants to draw any inspiration from linguistics for psychology, then it will still be a system of this kind -- i.e., some kind of a system that deals with discrete symbols with rules and transformations on those symbols.

This is exactly what has happened in psycholinguistics, where many studies are being performed, taking seriously the notions of linguistic transformation and the encoding of meaning (semantics) in the so-called deep structure (Chomsky, 1967). The attempt to characterize the development of children's grammars, which thereby attributes to them a (simple) system of rule following behavior on symbol structures (language utterances), is part of the same picture (Smith and Miller, 1966).

Problem solving. A third development is the simulation of cognitive processes in problem solving by means of computer programs. This is the development associated with (intimately entwined with, would be a better phrase) artificial intelligence. The problem solver is viewed as a symbolic system, capable of following strategies of search, applying heuristics, calculating results, both symbolic and (on occasion) numeric, and evaluating partial results. The efforts referred to here are those one would also consider psychology (in line with the choices with respect to Figure 1), namely, those where direct comparison is made between the symbolic system and data from human behavior. Good representatives of this work can be found in the well-known collection by Feigenbaum and Feldman (1963) (see also Reitman, 1965).

Concept formation. A fourth area of development is in the study of concept formation. Work in this area, of course, goes back many years (e.g., to Hull, 1920). A major turning point is symbolized by the book by Bruner, Goodnow and Austin (1956), which made extensive use of the notions of strategy and hypothesis formation, as well as cognitive strain (being essentially the dynamic memory load needed to carry out various strategies). The system implied there was very much a symbolic system, though its inspiration came out of decision and game theory, rather than computer science.

However, though there has been substantial work in artificial intelligence on concept formation (inspired in large part by the Bruner, Goodnow and Austin analysis) and even on information processing models for its psychology (e.g., Hunt, 1962; Hunt, Marin and Stone, 1966), most of the upsurge of work that followed in the late fifties and early sixties could not reasonably be seen as working within an information processing framework. It would be better characterized as a straightforward experimental investigation of psychological phenomena, in which various limited questions were posed and investigated without any deep commitments to the type of processing system implied. For example, studies were done to show that there was a systematic effect of the number of relevant versus irrelevant dimensions in the stimulus configurations; and to show the effect of the availability of past information (Bourne, 1966).

However, gradually more explicit assumptions have been made about the nature of the subject's processing -- first in terms of hypothesis testing (Restle, 1962), more recently in terms of general rule-following behavior (Haggard and Bourne, 1965). These shifts imply a symbolic processing system.

Summary. My purpose in quickly going over these lines of development is not to establish them in any detail -- for this I have hardly done. It is to call your attention to the use of symbolic models in many places throughout experimental psychology. It suggests (and I maintain) that a shift in the Zeitgeist in psychology has taken place toward a view of man as an information processor.

In fact, I have left out several additional lines of development, for example the work in verbal learning. Although the non-psychologist can be pardoned for

thinking that this is coextensive with psycholinguistics, in fact it is a separate experimental tradition going back to Ebbinghaus and his use of nonsense syllable learning (1885). Work on the learning of verbal materials -- serial lists, paired associates, and free recall -- have been one of the bastions of S-R psychology, since the phenomena lend themselves well to explanation in terms of the formation of associations.

Let me quote a paragraph from a recent study by a psychologist who has long worked in this area. The study is entitled, "Image as a mediator in one-trial paired-associated learning." It seeks to investigate the use of mneumonic devices in memorization. It has long been known that if you want to memorize a list of, say, ten items, then a good way to proceed is by having an already learned list of associations, say, 1-bun, 2-shoe, 3-tree, 4-door, ... 10-hen, and then (to memorize the new material) forming a bizarre visual scene involving each of the items and the word in the permanent list. That is, if the first item to be memorized was whale, then visualize the whale with a bun in its mouth; if the second was a bicycle, then visualize the bike riding down the toe of the shoe, and so on. It will then be found (so goes the lore) that the k^{th} item can be reliably recalled by going from the number, say 4, to its word, say door, and then to the visual scene, from which the object can be recalled. (The "1-bun,..." list is memorized once and can be used for a lifetime.)

> The present study is a preliminary effort to make some experimental contact with the hypothetical construct of visual image with no immediate intent to assert the reality of such a phenomenon. In the present study S's were instructed to form visual images and to use them in memorizing lists of words. Whether or not they did so may remain in question. The fact that they accepted the instructions and maintained that they followed them cannot be denied. In this report the term "image" will be used to refer to the processes S's said they followed when instructed to "picture"

> the 10 articles mentioned in connection with the previously learned list of 10 words that rhyme with the first 10 numbers. (B.R. Bugelski, E. Kidd and J. Segmen, 1968, p. 70.)

This quotation accurately reflects the present state of verbal learning research. It is still much enmeshed in a behavioristic stance, which views with alarm attempts to deal with internal processes (symbolic or otherwise). But they are being driven to such attempts, in large part because of a shift in view to an organism as an active, symbol manipulating system. In the decades prior to the current one, such notions as imagemediated paired associate learning simply did not call for investigation. The current attempts testify to the shift in the Zeitgeist.

A final comment: if one looks at where the excitement has been over the last ten years in psychology -- the places where rapid growth is taking place and which people talk about when asked "what's new" -- a substantial fraction of these turn out to be connected to this shift towards information processing models. The work on immediate memory is one; the rise of a linguistically oriented psycholinguistics is another; the study of children's grammar (within psycholinguistics) is a third. (Possibly the work on problem solving is yet another, but that is more difficult for me to assess, since I am so involved in it.)

5. POINT TWO: FROM IMMEDIATE MEMORY TO IMMEDIATE PROCESSOR

In the discussion of the possible relationships of information processing models to psychology we opted for the use of such models as detailed theories of behavior, rather than, say, metaphors or exercises in the discipline of operationalism. Even taking for granted the extent of the activity discussed above, there is still the question of its nature. Does the work on immediate memory use the notions of information processing only as a metaphor, rather than theory? After all, in a primarily experimental discipline (such as psychology still remains), one can play fast and loose with many verbal formulations and many metaphors, so long as they lead to asking interesting experimental questions.

Let me pursue this question with respect to the work on immediate memory. To understand this area you must know some background. The behaviorist era in psychology, which reigned in its various forms for the thirty years prior to World War II, moved the question of learning to be the central question of an objective psychology. The study of sensation and perception gradually came to take subordinate places. Even more so, the study of memory became simply an aspect of learning. When work on immediate memory was restimulated in the fifties and sixties, it was largely as a re-emphasis within the notion of learning. Thus, these studies could be conducted with only the issues of memory in mind -- the nature of acquisition, retrieval, capacity, reliability, etc.

If I were to suggest to this audience that they study the structure of an unknown information processing system, then certainly the kinds of memories would be of prime importance, i.e., their capacities and access characteristics. But the nature of the rest of the central processor would be of equal interest, i.e., the control structure and the basic processing operations. Almost none of this concern with processing, as opposed to memory, is evident in the earlier psychological literature on immediate memory. But recently -- within the sixties -- there has been a shift toward such concern. And this shift carries with it the use of information processing theories in detail.

Some brief examples are appropriate to show this situation. I will not attempt any historical comparison, but rather give examples of current work that uses information processing assumptions, not as metaphor but as a theory.

If we ask a subject "What is the 7th letter after G in the alphabet?" (Answer: N), it will take him about a second and a half to respond. If we vary this question by changing the starting letter and the number, then we get a curve, such as that shown in Figure 3 for subject RS. If we kept at our subject long enough, we might expect him to memorize all the answers (there are only 26x25 = 650 distinct questions), in which case the time to respond might be independent of the details of the question. But barring that, the subject must somehow generate the answer. The figure immediately suggests that he does this by counting down the

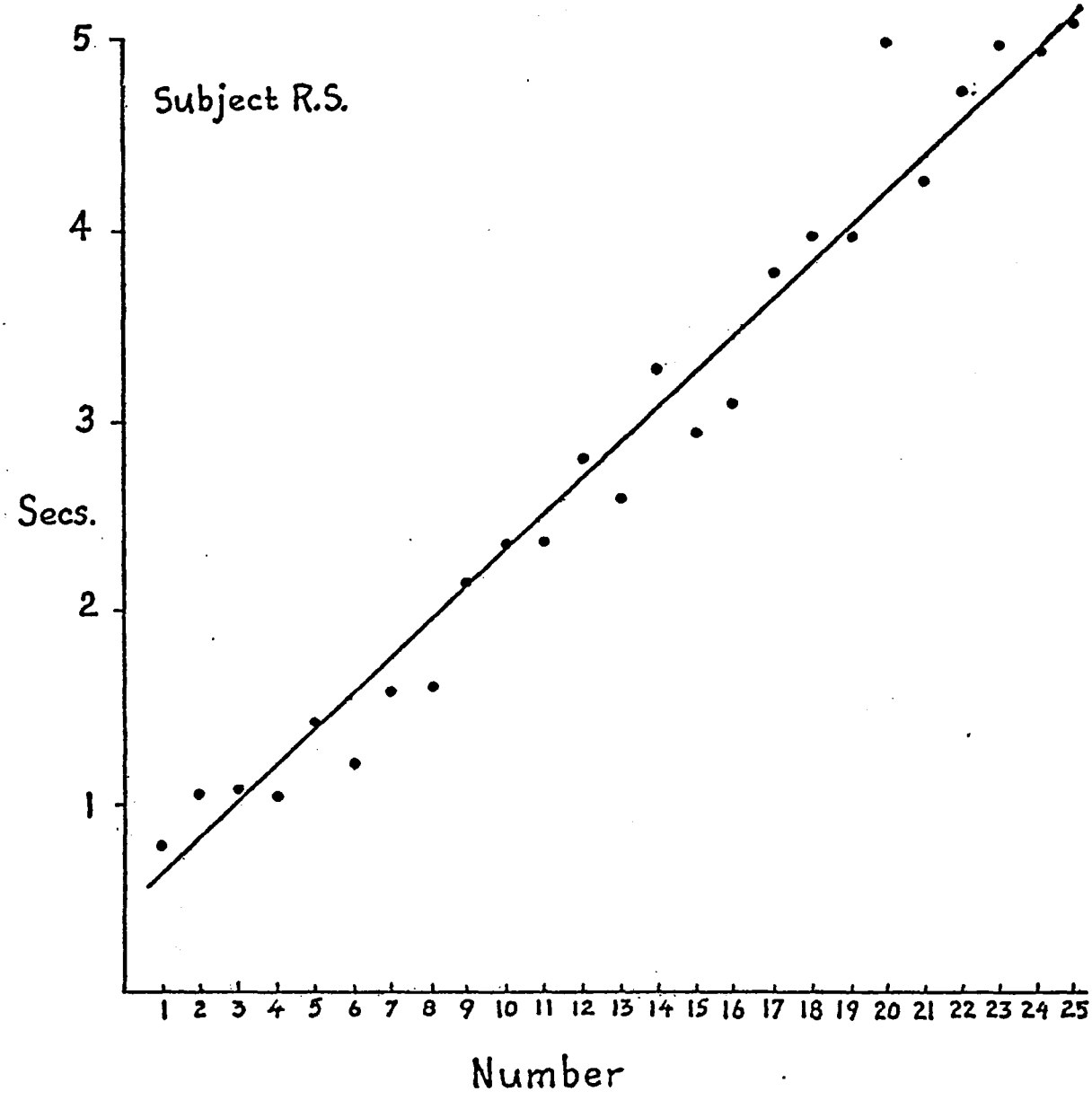


FIGURE 3: Average reaction time to count down alphabet (adapted from Olshavsky, 1965, Fig. 2).

alphabet at a constant rate (he is silent during the interval between question and answer, so may proceed any way he wants). That is, we model our subject as a simple serial processing system which has operations of "get next," "add 1," "test if tally = n" and "speak result," along with some control structure for integrating the performance into a repetitive loop. The linearity arises because the same operations are being performed repetitively.

This particular figure, taken from a Masters thesis at CMU (Olshavsky, 1965), is not an isolated example. It shows several things that characterize much of the experimental work on the immediate processor. First, the task is very simple, thus illustrating the earlier point that information processing systems, not artificial intelligence systems should be our main concern. Second, the response measure is reaction time, so that the task is to infer the structure of a complex process from the time it takes to perform it. Third, a population of tasks is used, so that some gross aspect, such as the linearity in Figure 3, contains the essential induction from data to mechanism. Since, in fact, reaction times are highly variable, it is this last feature (initiated by Neisser, 1963) which distinguishes current work from a long history of earlier work on reaction times that didn't bear such fruit.

Figure 4, from a study by Sternberg (1967), reinforces these points. He gave his subject a set of digits, say (1, 3, 7), and then asked them if a specified digit, say 8, was a member of the set. He finds, as the figure shows, that not only does it take longer to answer the question for larger sets, but the relationship is linear. Thus, again, the natural interpretation is according to a processing system engaged in repetitive search. (Though the search here is through immediate memory, whereas it was through long term memory in Figure 3.) Now the point of showing this second example is that Sternberg goes on to use this basic result in an ingenious way. In one condition he presents the subject with a fuzzy, degraded image. What should happen?

We know, independently, that it takes longer to compare a degraded image than a clear one to a known digit. One possibility is that the subject works with the

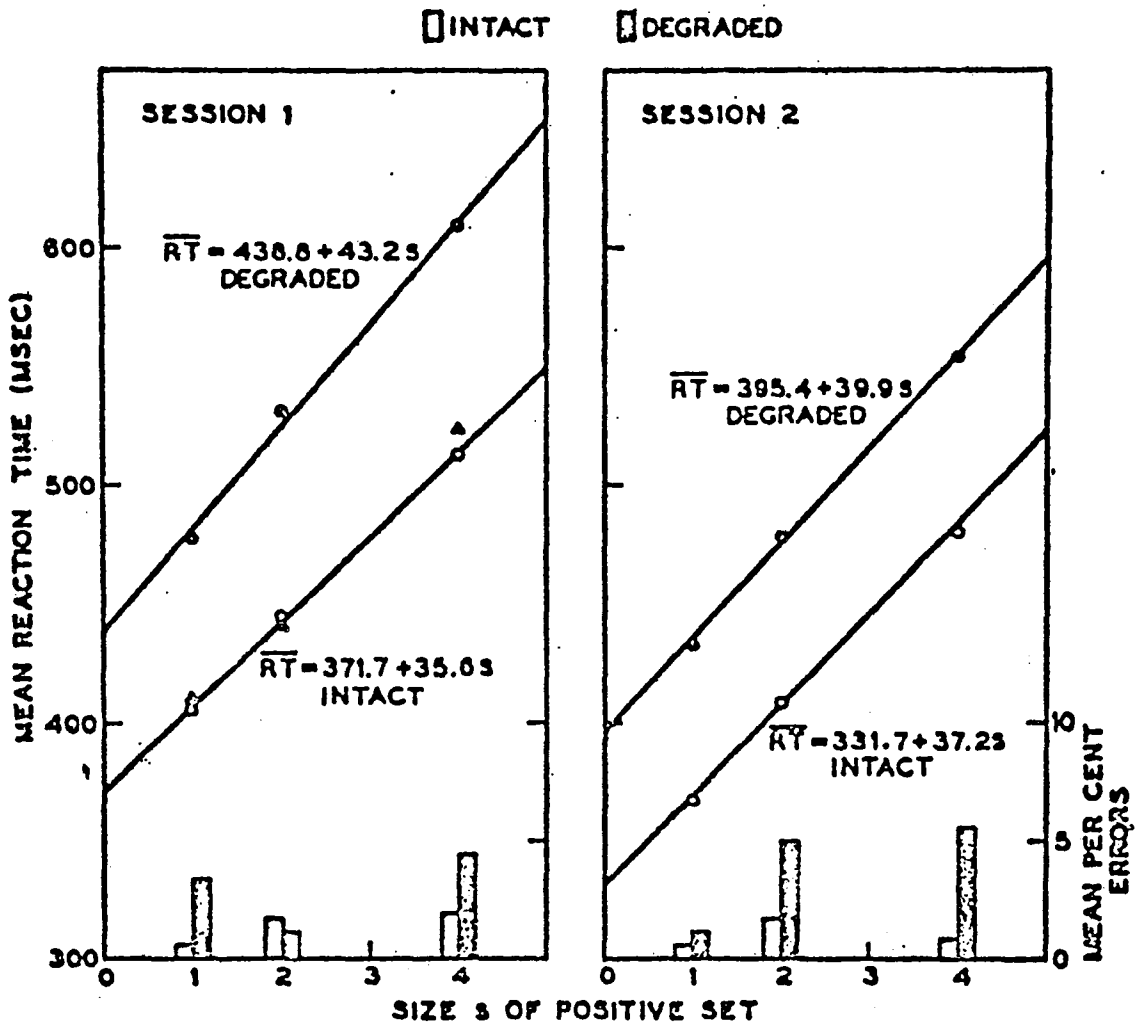


FIGURE 4: Average reaction time to identify membership in set (from Sterpberg, 1967, Fig. 4).

image, thus having to make the more difficult comparison at each step of the search. If this were the case, the slope of the data line should be greater for the fuzzy image than for the clear image. A second possibility is that the subject initially identifies which digit the fuzzy image represents and then compares an internal representation on each stage of the search. In this case, the slope should be the same, but there should be extra time for initialization. As Figure 4 shows, the latter clearly prevails. Thus we can infer that the operation of perceptual identification occurs prior to the search in immediate memory.

The point of this study, for us, is to see how definitely Sterberg is working with a processing model. The situation is so simple that the key properties can be inferred without creating a program to simulate the subject. But the dependence on the detailed theory is no less for that.

I will present you one more example, since I really wish to convince you of the extent to which information processing theories are taking hold at the level of studying the immediate processor. This is work done by Donald Dansereau in a Ph.D. thesis just completed at Carnegie-Mellon (Dansereau, 1969). He studied the process of mental multiplication, e.g., "Multiply 27 by 132 in your head and when you are through, give the answer." His subjects were all highly practiced; even so, it takes a substantial length of time--e.g., about 130 seconds for 27x132. Again, as with these other studies, time was the measure, and he gave his subjects a large population of tasks.

Now the fundamental fact about mental multiplication is that any crude processing model does quite well. That is, a reasonable count of the steps required by the method the subject uses (e.g., 62x943 requires 5 holds for the given digits, 6 single-digit multiplications, 9 additions, 4 carries and 11 holds for a total difficulty factor of 35) does quite well in predicting the time taken. Figure 5 shows actual times taken versus this difficulty factor for a particular subject. The linear regression accounts for about 90% of the variance. However, this result is not at all sensitive to the exact assumptions. Other work has gotten similar results with quite different measures (Thomas, 1963), though in all cases they are crude processing models.

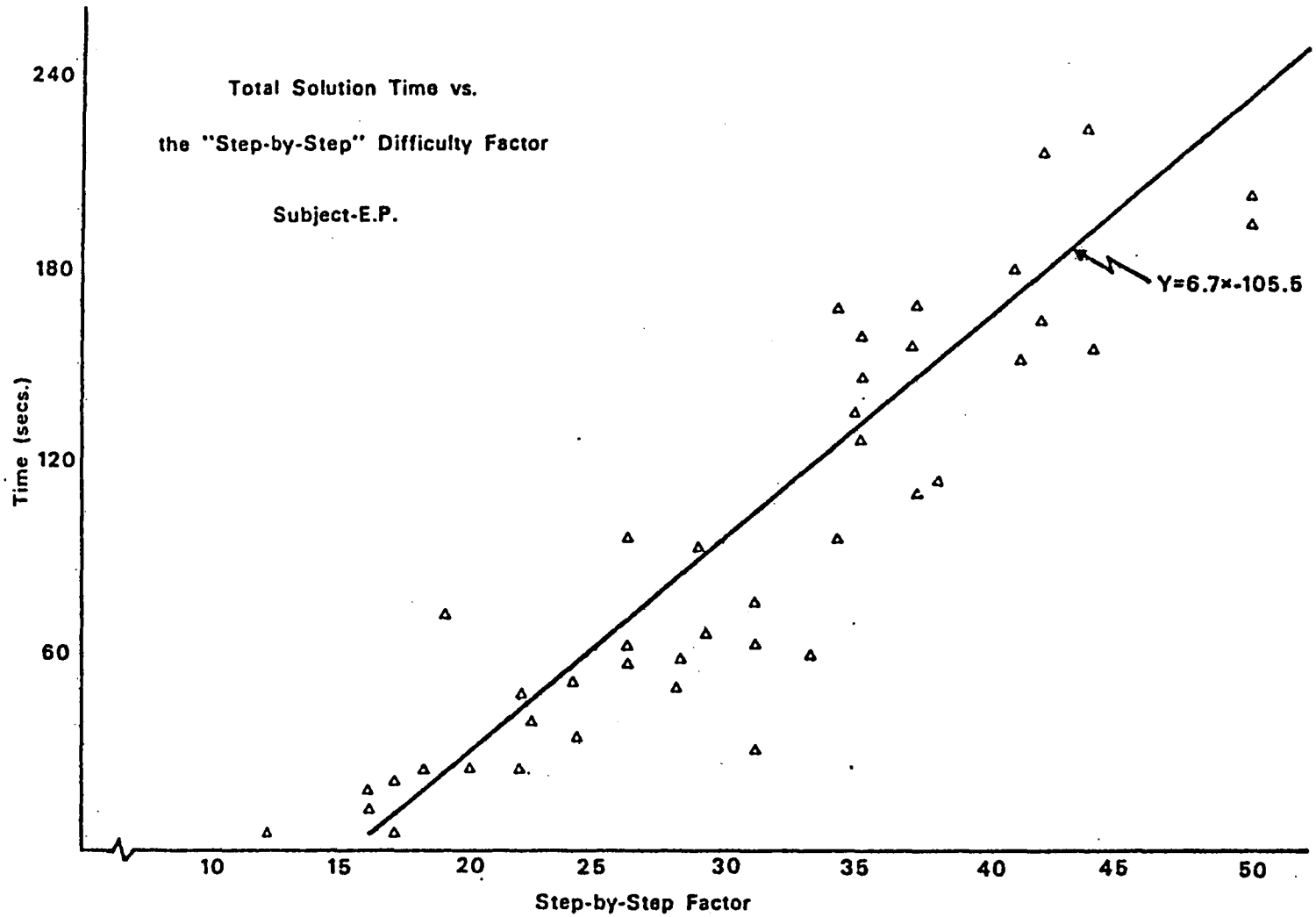


FIGURE 5: Memory transfer rates in mental multiplication, ordered by difficulty factor (from D. Dansereau, 1969, Fig. 4).

Dansereau went on to construct a more refined model, in which he postulated several kinds of memories with associated transfer times between and within memories. There was an image store, where operands had to be positioned, as in a template, in order to be added or multiplied. There was a short term memory that held a small number of digits, e.g., the definition of the problem or intermediate results. Finally, there was a long term memory in which information could be fixated for an indefinite period of time. The transfer times are shown in Figure 6. They were obtained from independent experiments, either already in the literature or done by Dansereau. Thus, these times are not parameters to be estimated from the primary data on performance.

Figure 7 shows the results of this model. The system is complex enough to require simulation. The times taken by the simulation are shown as open circles and the actual times by the solid circles. Both are plotted against the difficulty factor used in the prior figure. (Thus there are many dots for a given difficulty factor, since there are many different multiplication problems with the same factor.) It can be seen clearly that the simulation has provided a next order of improvement, fitting the "staircase" effect of the actual data. This fit is not due to an excess of parameters, since the only parameter used to fit the data was a scale change. All others, as remarked above, were estimated independently from other data.

Although we have no space to discuss it, the model shows that very little time is spent in the act of multiplying or adding. Rather, significant amounts of time are spent in memorizing intermediate results (which we expected) and in positioning operands (which we did not expect).

This work shows clearly the shift from models of memory to models of the immediate processor. Memory, of course, remains central to the system, but there is much more as well. Furthermore, we have moved to where an explicit theory must be built of the situation (the simulation), even though the task is still not one that artificial intelligence finds of much interest per se.

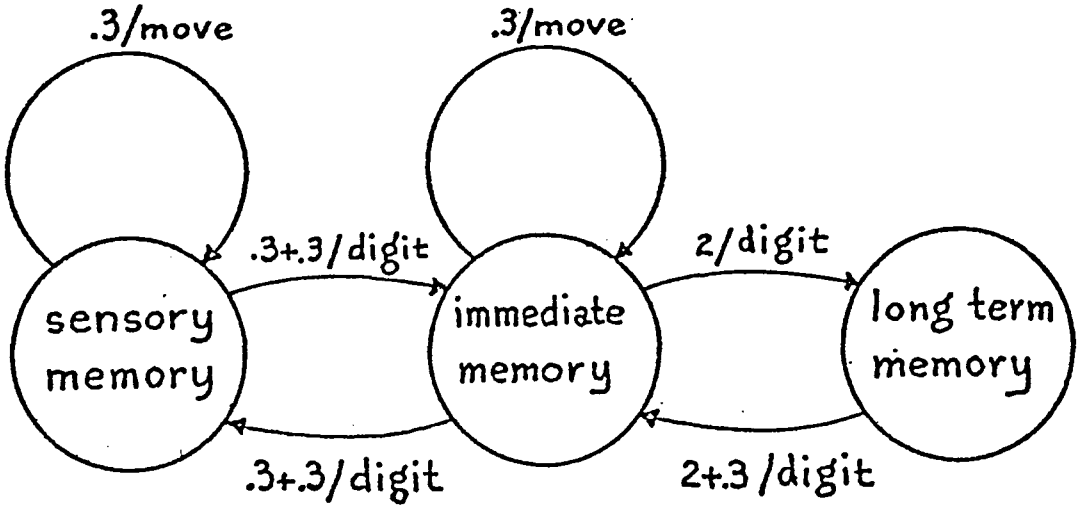


FIGURE 6: Memory transfer rates in mental multiplication model (after D. Dansereau, 1969, Table 3).

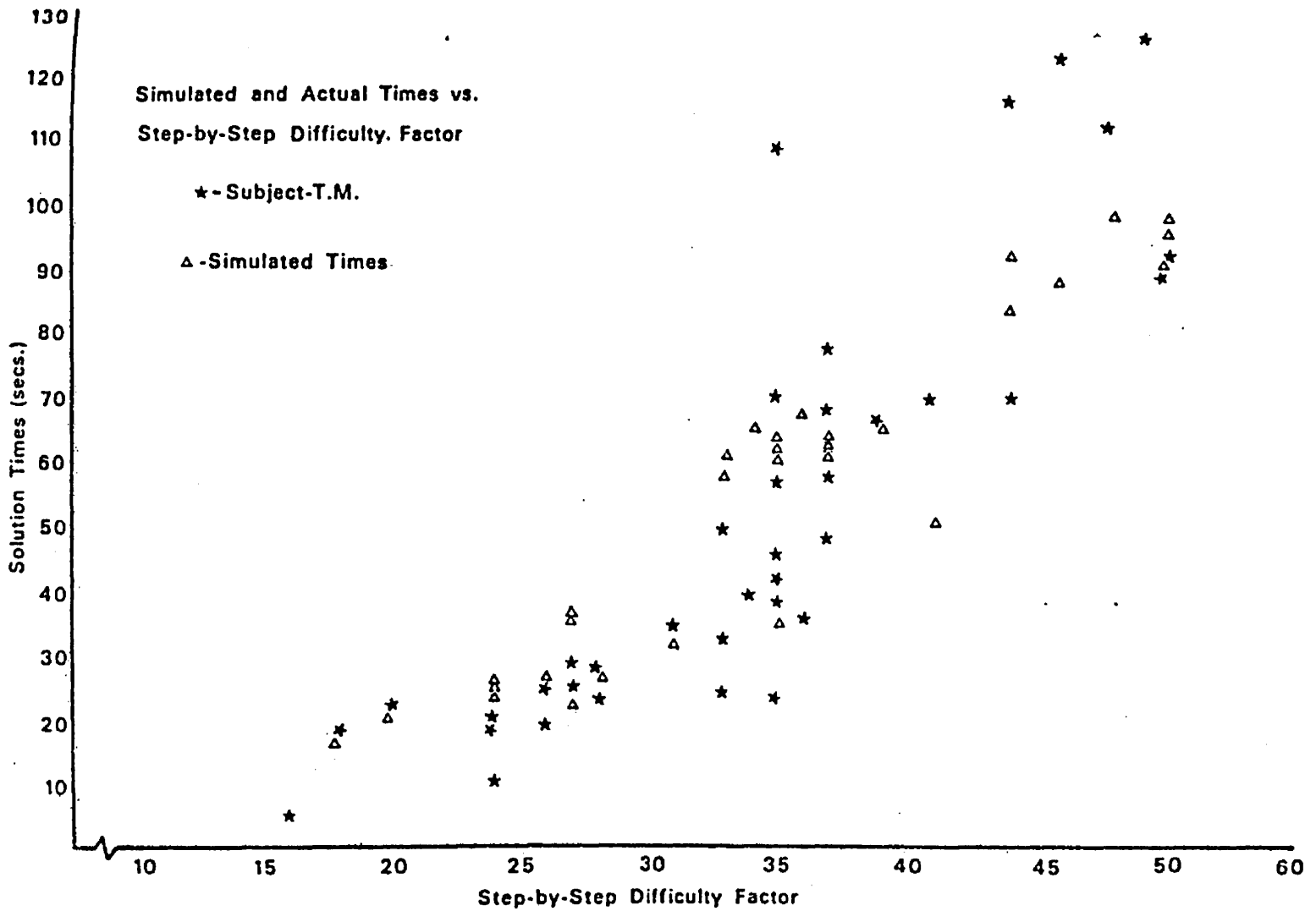


FIGURE 7: Average reaction time for mental multiplication for subject TM (*) and Simulation (o) (from D. Dansereau, 1969, Fig. 19).

6. POINT THREE: ON BEING SERIOUS

I have tried to illustrate with examples from one area, immediate memory, that theories of man as an information processor are being used in serious and detailed ways. I would now like to turn this conclusion around. There have always been two feelings held by workers in artificial intelligence about themselves: (1) they were proceeding independently of any concern with human behavior (i.e., not simulating); alternatively (2), they were in fact being relevant to how man thinks. Both these views are, in my mind, legitimate -- including their conjunction, which has been my personal position in some of our work (e.g., GPS).

I wish to address myself to those of the second (simulating) persuasion. By now, anyone who is serious about the psychological relevance of his work in artificial intelligence had better be prepared to deal with detailed data of humans in specific situations, experimental or otherwise. As we discussed in connection with Figure 1, there are many ways in which a work in artificial intelligence could be considered relevant to the study of human behavior. All these ways remain legitimate. But the gradual success of the detailed use of information processing theories means that none of the less demanding ways carry much punch (though there will always be exceptions, naturally).

This same point was reached some years ago with respect to neural modeling and physiology. No neural modeling is of much interest anymore, unless it faces the detail of real physiological data. The novelty and difficulty of the tasks undertaken by heuristic programming has tended to push the corresponding day of reckoning off by a few years. The development of symbolic systems that would behave in any way intelligently produced sufficiency analyses that were in fact relevant to the psychology of thinking. But the automatic relevance of such efforts seems to me about past.

Let me illustrate this point briefly. In the last few years Ross Quillian has developed a model of semantic memory (Quillian, 1965, 1969). Many of you are undoubtedly aware of it; Bob Simmons discussed it to some extent in his paper at this conference. The essential features are (1) each concept is a node in a

semantic net (as in several other programs, such as SIR (Raphael, 1964); and (2) a complex structure encodes the definition (as in dictionary definition) of the word, thus relating it to the other concepts used in its definition. In his original work he used the task of giving the system two words, e.g., FIRE and BURN, and having it state the relationship between these concepts; e.g., FIRE IS CONDITION WHICH BURN, also TO BURN CAN BE TO DESTROY SOMETHING ON FIRE.

Now this program is an example of sufficiency analysis, as we have used the phrase. For the system is not intended as a detailed model of human memory and it was never tested as such. But it is relevant to psychology, because he was able to make (and demonstrate via the living program) conceptual progress in how human memory might be structured for tasks where we understand by general experience what performance can typically be expected of humans. Indeed, the work was a Ph.D. dissertation in Psychology at Carnegie-Mellon (Quillian, 1966).

There is a sequel to this work -- and it makes my point. Quillian is, indeed, interested in the psychology of human memory. Thus, he followed up this work in sufficiency analysis with an attempt to explore whether human memory could be modeled by such a structure (Collins and Quillian, 1969). The essential feature of a semantic net is that information about a concept is not all localized at the node corresponding to that concept, but is distributed through the network. Thus, that a canary can sing, might be located at canary, but that a canary can fly is probably not located at canary, but at bird, since it is a property of all birds. Similarly, that a canary has skin is probably not even located at bird, but rather an animal. If this were the case, then it should take longer for such a system to answer yes or no to such questions (when embedded in a population of other questions, such as "Does a house sing?", "Does a cat fly," etc.). Further, if the net is homogeneous in its structure, then there should be a constant operation time to go from node to node in the net.

Figure 8 shows the results of asking these questions experimentally of humans, using reactions times. The points are averages over populations of questions of similar type. The quantity of interest is the difference between points, as

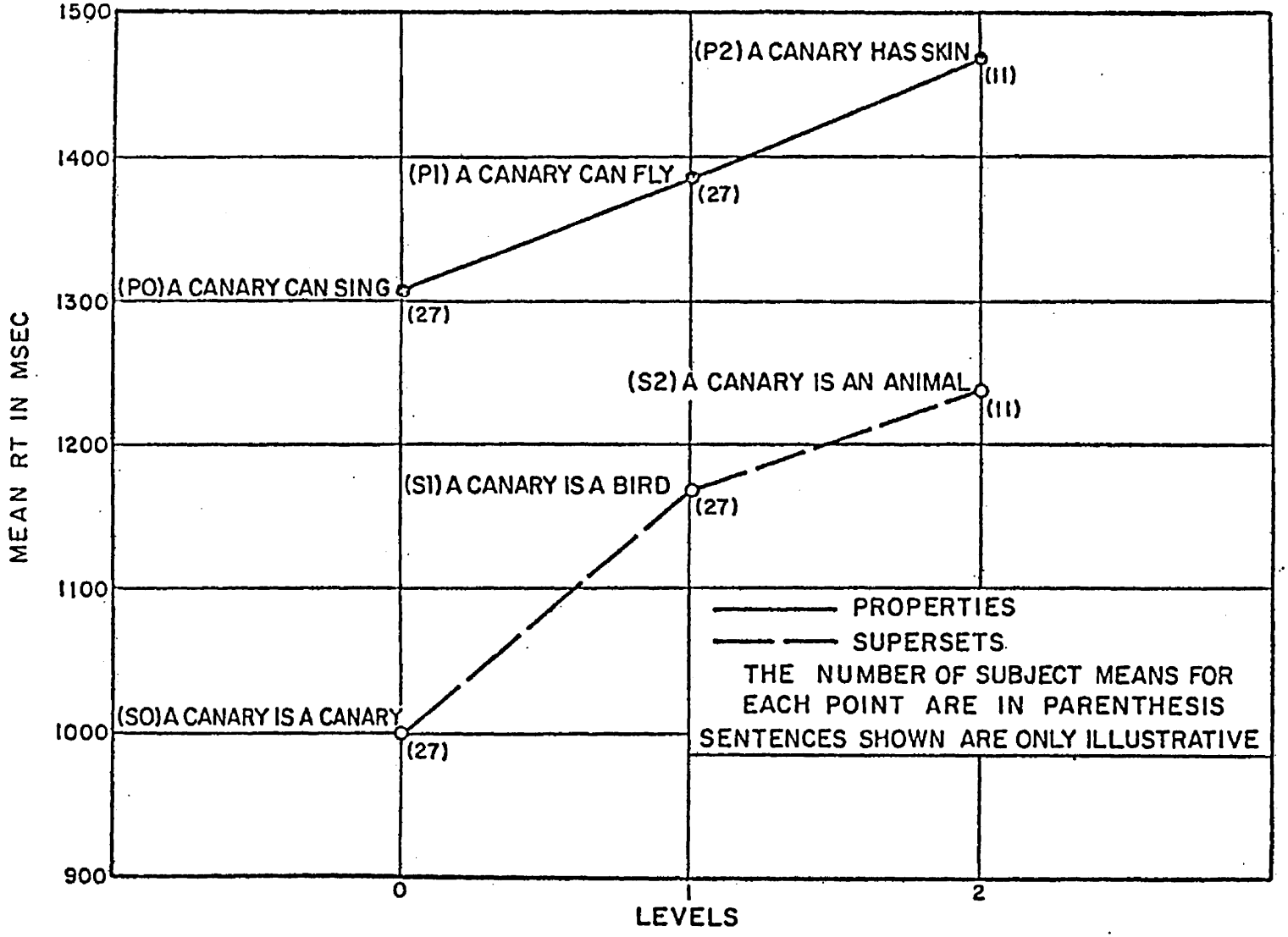


FIGURE 8: Average reaction times for different types of true sentences (from Collins and Quillian, 1969, Fig. 2).

indicated above. Three of them are essentially identical at the 80 ms. The fourth, from "A canary is a canary" to "A canary is a bird" is too large, due, it appears, to the reaction to the former being too fast. But this is the one question that admits of an answer by perceptual matching, avoiding the meaning of the word "canary" altogether (hence the time to go from image node in the net). There is other evidence in the literature that indicates the same phenomena, e.g., it takes longer to recognize that A and a are the same than that A and A are the same, since (apparently) the latter can be done by a perceptual match and the other not (Posner, 1968).

Before leaving Quillian's work, let me note that a number of assumptions are embedded in Figure 8. Thus, nothing guarantees that the particular words are related as the experimental analysis assumes, even if the structure of memory is precisely of this postulated type, i.e., the information that a canary can fly could be as close to "canary" as that it could sing. Thus, Collins and Quillian attempted to select a population of words that had a high prior plausibility of being this way -- and were successful, as you can see.

To make the point of this work for us explicit: (1) if you want to assert the relevance of a theoretical information structure to psychology, then you had better build a bridge from it making use of experimental data; (2) it can be done. The bridge built by Collins and Quillian is a frail one, of course, since it only addresses one tiny aspect of the total memory system and it is compatible with many other similar structures. Indeed, as you would expect, neither Quillian nor anyone else thought the original structure was right in detail, and his second iteration is a much modified system (Quillian, 1969). But with the experimental work, it has passed well beyond metaphor.

Even more briefly, let me present one more example. This is taken from my own work with H.A. Simon on problem solving. I insert it, both because I'm always inclined to mention some of my favority psychology, especially when it fits the story so well, and because none of our examples, with the possible exception of Quillian's memory structure, are from artificial intelligence.

We work intensively with cryptarithmetic tasks, such as the SEND+MORE=MONEY that Fikes used to describe the operation of his program, REF-ARF in this conference. It is a puzzle that still retains modest challenges as a task for various problem solving programs. We could simply work with programs for solving this task, say like REF-ARF, or even some that are more like we imagine a human processor to be constructed, e.g., with a short term memory, a long term memory, etc. From these we could draw various conclusions about the general character of human problem solving. In fact, we did exactly this with the Logic Theorist, our original program (Newell, Shaw and Simon, 1958). But for some time now, since the first work with GPS (Newell, Shaw and Simon, 1960, 1961), we have taken an attitude much as I am trying to prescribe here.

Thus, our typical operation is to present a subject with the task, asking him to talk aloud as he works on it. A fragment of the result, called a protocol, is shown in Figure 9, where the task is DONALD+GERALD=ROBERT and D=5 is given as initial information (Newell, 1966, 1967). We then attempt to construct a processing system that mirrors the behavior of the subject and agrees with what we know about human processing capabilities.

Typical collegiate subjects in this task can be described with excellent fidelity as working in a problem space, whose elements are the states of knowledge the subject can have about the task, and whose structure is given by a small set of operators that work on a given state of knowledge to produce a new state of knowledge. Problem solving is search through this problem space. This search, what we call the problem behavior graph (PBG), is shown in Figure 10 for the fragment of protocol shown in Figure 9. The four operators used by this subject are Process a column (PC), Assign a digit to a letter (AV), Generate the possible values for a letter (GN), and Test if a digit - letter assignment is legal (TD). (Actually, these are four specific variants of processes that meet these four general functional descriptions.)

The problem space, with its operators, is a reasonable description only if there exists an information processing system that describes the way the subject's

FIGURE 9: Fragment of Proofread or Subject 3 in Donald + Gerald = Robert (adapted from Newell, 1967).

Crypt-arithmetic Subject 3 Problem

```
  DONALD
+GERALD
-------
 ROBERT
```

D=5

B1	Each letter has one and only one numerical value --
B2	Exp: One numerical value
B3	There are ten different letters
B4	and each of them has one numerical value.
B5	Therefore, I can, looking at the two D's --
B6	each D is 5;
B7	therefore, T is zero.
B8	So I think I'll start by writing that problem here.
B9	I'll write 5, 5 is zero.
B10	Now, do I have any other T's?
B11	No.
B12	But I have another D.
B13	That means I have a 5 over the other side.
B14	Now I have 2 A's
B15	and 2 L's
B16	that are each --
B17	somewhere --
B18	and this R --
B19	3 R's --
B20	2 L's equal an R --
B21	Of course I'm carrying a 1.
B22	Which will mean that R has to be an odd number.
B22.1	
B23	Because the 2 L's --
B24	any two numbers added together has to be an even number
B25	and 1 will be an odd number.
B26	So R can be 1,
B27	3,
B28	not 5,
B29	7,
B30	or 9.
B30.1	
B31	Exp: What are you thinking now?
B32	Now G --
B33	Since R is going to be an odd number
B34	and D is 5,
B35	G has to be an even number.
B35.1	
B36	I'm looking at the left side of this problem here where it says D + G.
B37	Oh, plus possibly another number
B38	if I have to carry 1 from the E + O.
B39	I think I'll forget about that for a minute.
B40	Possibly the best way to get to this problem is to try different possible solutions.
B41	I'm not sure whether that would be the easiest way or not.

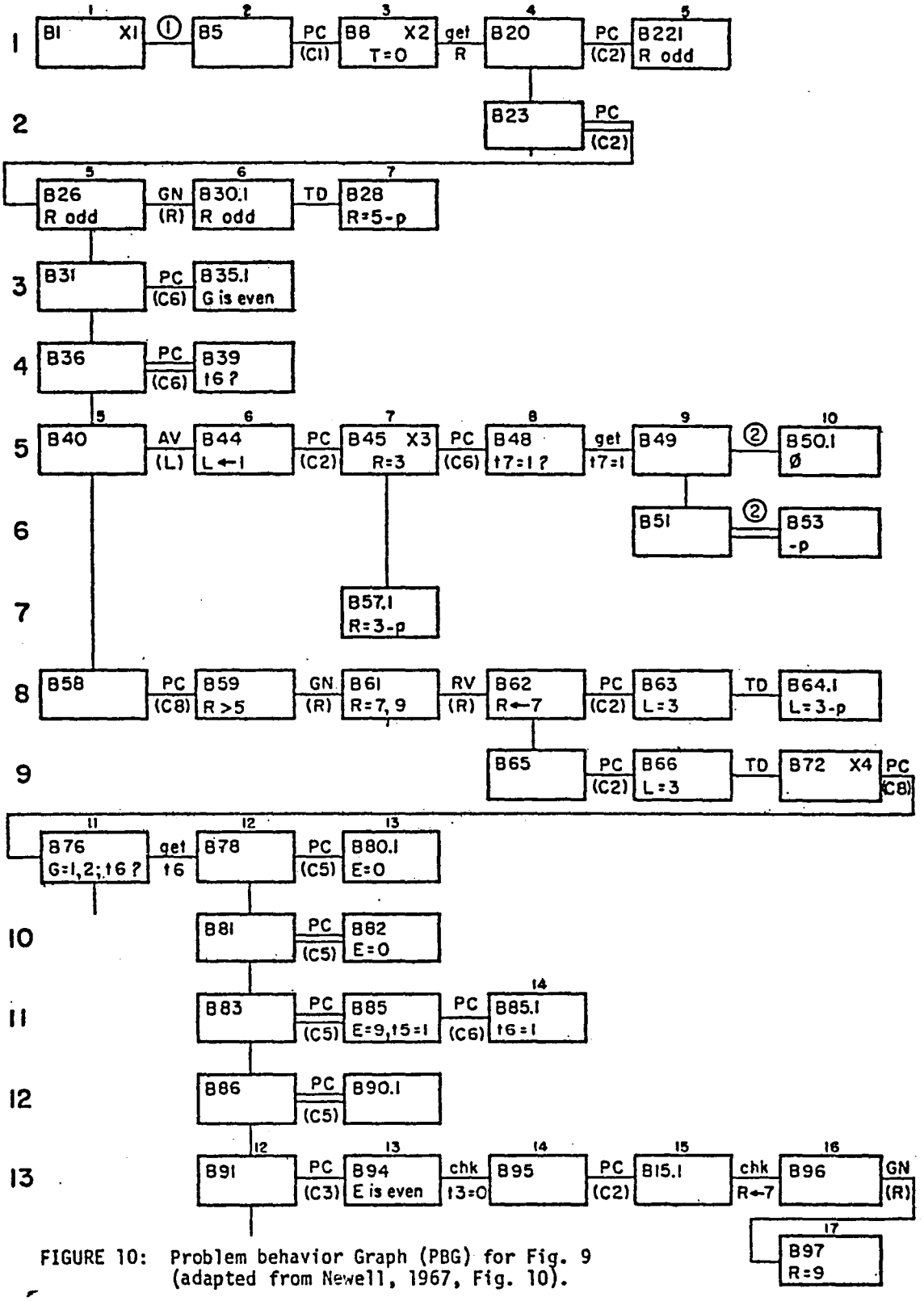


FIGURE 10: Problem behavior Graph (PBG) for Fig. 9 (adapted from Newell, 1967, Fig. 10).

search goes in this problem space. It is our fashion to write these programs as production systems, meaning an ordered set of condition-action expressions with a control structure that continually executes the first expression whose condition is true. (The actual expression being executed changes as the action parts continually modify the immediate memory, which is what the conditions test.) Again, there is no space to describe a complete production system for our subject. A typical production, available in almost all subjects can be paraphrased, "If a new equality expression has just occurred, then find a column that contains the letter involved and process that column." In symbols:

<letter> = <digit> new → Find-column(<letter>); Process-column (column).

The left side is written as a BNF grammar so that any expression in the knowledge state of the right form will trigger the production (e.g., 'T = 0 new').

Given a production system (the one for the subject of Figure 9 and 14 productions), one can attempt an accounting of the nodes of the problem behavior graph that are described successfully by a production. Figure 11 shows this accounting for the total protocol, in which the productions are ordered in terms of decreasing marginal utility. There are two kinds of errors. First, a certain fraction of the nodes in the problem behavior graph are not covered (errors of omission); this gradually drops to 11% with the total set of productions. Second, the wrong production gets evoked at a node in the graph, due to the ordering (errors of commission); these gradually rise as the set of productions increases to about 9% or 14% (two levels of error were counted in the original work).

This brief description is not meant to do more than indicate a scientific style, namely, one where programs (the production systems) are constructed to deal with specific instances of human data, with quantitative assessment of the comparison. Not revealed in this brief account is the concern for understanding how the programs of different subjects compare, so that a general theory of human problem solving emerges. The example should reinforce the story told by the earlier ones -- that if one is serious about the implications of work in artificial intelligence for psychology, then one must come to terms with data on human behavior.

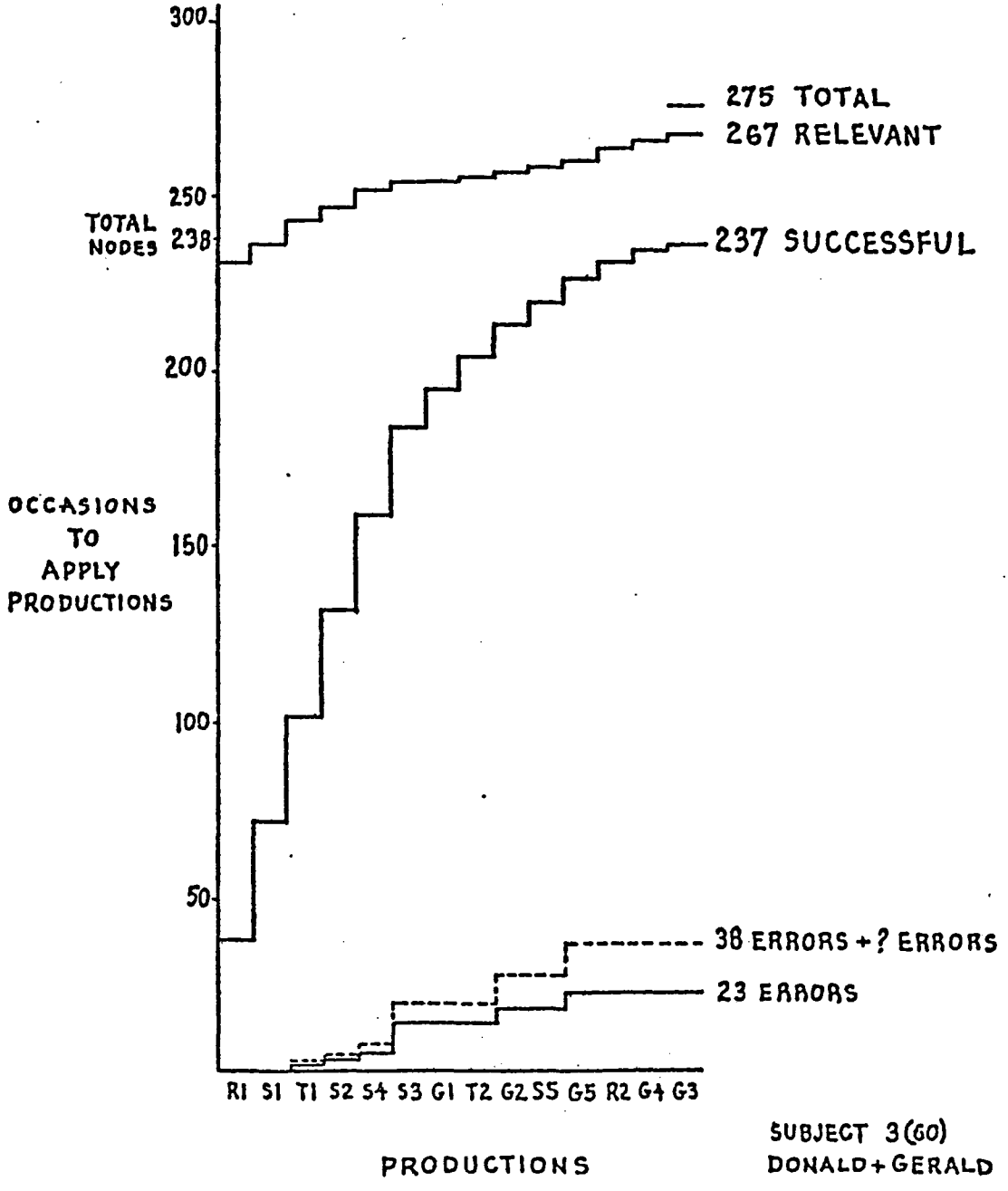


FIGURE 11: Summary of performance or production system for Donald and Gerald (from Newell, 1967, Fig. 16).

7. FINAL POINT: ON PSYCHOLOGY'S PREFERENCES

The main points are now made, in response to the letter from my psychologist friend. First, the frame of reference must be expanded from artificial intelligence to information processing systems on symbolic structures. Then, it is found that a very substantial penetration has occurred of information processing theories into psychology. Second, there is beginning a shift in experimental psychology from a concern exclusively with immediate memory to a concern with the whole of the immediate processor. This testifies to the use of information processing theories at a detailed level, and not just as experiment-guiding metaphor. Third, the coupling has become intimate enough that artificial intelligencers must take experimental data seriously if they want to claim any direct relevance of their work to psychology.

With these points made, I should rest content. But there is one place where I have finessed my letter-writing friend, and I must give him his due. For he did mean artificial intelligence and not information processing psychology in general (at least, so I believe). Now, whereas I would insist on the latter view, there is one respect in which his concern remains justified. The higher mental processes in American psychology have always held a secondary place. There are many reasons for this: the impact of behaviorism that simply denied the relevance of the mental; the feeling that the important thing was the elements (i.e., the basic act of learning), out of which complexity would grow automatically; the somewhat fanatical adherence to the scientific canon of simplicity first; the actual messiness of the area, as revealed by the relatively few, but continual, forays that have occurred. The reasons make little difference. The psychology of thinking and problem solving makes up a rather sad chapter in the history of psychology.

It is one of my fond hopes that the situation has finally changed -- that we have the techniques to deal with integrated goal oriented behavior on a par with any other part of psychology. But I am afraid that my correspondent is correct: that psychologists will not shift in mass to the study of thinking and problem solving. These areas will remain relatively lightly populated -- and this was in

part what he was decrying, for he expected it to be different some ten years after the initial work.

I think he is correct in his depression, since, as we have seen, there is large contact between information processing theories and psychology at the level of immediate memory (and psycholinguists too, though I didn't stress it as much). These are extremely exciting areas at the moment, as noted earlier. They can be attacked with relatively small amounts of theory and large amounts of sophisticated experimental techniques. They fit psychology's image of the proper thing to study: basic structure and not too much complexity. Thus, as experimental psychologists move towards assimilating information processing theories, they will gravitate towards the study of the immediate processor and basic language structure, and not toward thinking and complex problem solving. It is not without significance that only one out of five of my examples came from the heartland of artificial intelligence.

REFERENCES

1. Attneave, F., Applications of Information Theory to Psychology, Holt-Dryden, 1959.
2. Boring, E., "Mind and mechanism," American J. Psychology, 59, 2, April, 1946.
3. Bourne, L.E., Jr., Human Conceptual Behavior, Allyn and Bacon, 1966.
4. Broadbent, D., Perception and Communication, Pergamon, 1958.
5. Bruner, J.S., Goodnow, J.J. and Austin, G.A., A Study of Thinking, Wiley, 1956.
6. Bugelski, B.R., Kidd, E. and Segmen, J., "Image as a mediator in one-trial paired-associate learning," J. Experimental Psychology, 76, 69-73, 1968.
7. Chomsky, N., Aspects of Syntax, MIT Press, 1965.
8. Churchman, W., "The role of Weltanschauung in problem solving and inquiry," These Proceedings, 1970.
9. Colby, K.M. and Gilbert, J.P., "Programming a computer model of neurosis," J Math. Psychology, 1, 2, 1964.
10. Collins, A.M. and Quillian, M.R., "Retrieval time from semantic memory," J. Verbal Learning and Verbal Behavior, 8, 204-247, 1969.
11. Dansereau, D., An Information Processing Model of Mental Multiplication, Unpublished Ph.D. dissertation, Carnegie-Mellon University, 1969.

12. Ebbinghaus, H., Uber das Gedactnis, Dunker and Humbolt, 1885, translated as Memory, Dover, 1964.

13. Edwards, W., "Conservatism in human information processing," in B. Kleinmuntz (ed.) Formal Representation of Human Judgment, Wiley, 1968.

14. Feigenbaum, E.A. and Feldman, J. (eds.) Computers and Thought, McGraw-Hill, 1963.

15. Fikes, R., "Stating problems as procedures to a general problem solving program," These Proceedings, 1970.

16. Haygood, R.C. and Bourne, L.E., Jr., "Attribute- and rule-learning aspects of conceptual behavior," Psychological Rev., 72, 3, 175-195, 1965.

17. Hull, C., "Quantitative aspects of the evolution of concepts," Psychological Monographs, No. 123, 1920.

18. Hunt, E.B., Concept Learning: an information processing problem, Wiley, 1962.

19. Hunt, E.B., Marin, J.K. and Stone, P., Experiments in Induction, Academic, 1965.

20. Kendler, T., "Concept formation," in Annual Review of Psychology, 1961.

21. McGuire, W.J., "Theory of the structure of human thought," in R.P. Abelson, et al, (eds.) Theories of Cognitive Consistency: a Source Book, Rand McNally, 1968.

22. Melton, A., "Implications of short-term memory for a general theory of memory," J. Verbal Learning and Verbal Behavior, 2, 1-21, 1963.

23. Miller, G.A., "The magical number seven, plus or minus two," Psychological Rev. 63, 81-97, 1956.

24. Miller, G.A., "The study of intelligent behavior," Annals of the Computation Laboratory of Harvard University, 31, 1962.

25. Minsky, M. (ed.) Semantic Information Processing, MIT Press, 1968.

26. Neisser, U., "Decision time without reaction time: experiments in visual scanning," Amer. J. Psychology, 76, 376-385, 1963.

27. Newell, A., "On the analysis of human problem solving protocols," in J.C. Gardin and B. Jaulin (eds.) Calcul et Formalization dans les Sciences de l'Homme, Centre National de la Recherche Scientifique, Paris, 1968.

28. Newell, A., Studies in Problem Solving: Subject 3 on the Cryptarithmetic Task DONALD + GERALD = ROBERT, Carnegie-Mellon University, 1967.

29. Newell, A., Shaw, J.C. and Simon, H.A., "Elements of a theory of human problem solving," Psychological Rev., 65, 3, 151-166, 1958.

30. Newell, A. and Simon, H.A., "GPS, a program that simulates human thought," in Lernende Automaten, Oldenbourg, 1961; reprinted in Feigenbaum and Feldman, 1963.

31. Newell, A. and Simon, H.A., "Programs as theories of higher mental processes," in R.W. Stacy and B. Waxman (eds.), Computers in Biomedical Research, vol. 2, Academic Press, 1965.

32. Norman, D., Memory and Attention: an introduction to human information processing, Wiley, 1968.

33. Oettinger, A.G., Run Computer Run, Harvard U., 1969.

34. Olshavsky, R., Reaction Time Measures of Information Processing Behavior, Unpublished M.S. thesis, Psychology Department, Carnegie-Mellon University, 1965.

35. Posner, M.I. and Mitchell, R.F., "Chronometric analysis of classification," Psychological Rev. 74, 392-409, 1967.

36. Quillian, M.R., Semantic Memory, Ph.D. dissertation, Carnegie-Mellon University, 1966; reprinted in Minsky, 1968.

37. Quillian, M.R., "The Teachable Language Comprehender: a simulation program and theory of language," Comm. ACM, 12, 459-476, 1969.

38. Raphael, B., SIR, A Computer Program for Semantic Information Retrieval, Ph.D. dissertation, MIT, 1964; reprinted in Minsky, 1968.

39. Reitman, W., Cognition and Thought, Wiley, 1965.

40. Reitman, W., "Information processing models, computer simulation and the psychology of thinking," in J.F. Voss (ed.), Approaches to Thought, Merrill, 1969.

41. Restle, F., "The selection of strategies in cue learning," Psychological Rev. 69, 329-343, 1962.

42. Shepard, R., "Comments on Professor Underwood's paper," in C.N. Cofer and B.S. Musgrave (eds.), Verbal Behavior and Learning, McGraw-Hill, 1963.

43. Simmons, R.F., "Natural language question answering systems: 1969," These Proceedings, 1970.

44. Smith, F. and Miller, G.A. (eds.) The Genesis of Language, MIT Press, 1966.

45. Sternberg, S., "Two operations in character recognition," Perception and Psychophysics, 2, 43-53, 1967.

46. Thomas, H.B.G., "Communication theory and the constellation theory of calculation," Quart. J. Experimental Psychology, 15, 173-191, 1963.

47. Uhr, L., "Computer simulations of thinking are just (working, complete, big, complex, powerful, messy) theoretical models," in J.F. Voss (ed.) Approaches to Thought, Merrill, 1969.

48. Uhr, L. (ed.) Pattern Recognition, Wiley, 1966.

49. Uhr, L. and Vossler, C., "A pattern recognition program that generates, evaluates and adjusts its own operators," AFIPS Proc. WJCC, 19, 1961; reprinted in Feigenbaum and Feldman, 1963.

☆U.S. GOVERNMENT PRINTING OFFICE: 1980-311-451/2659