2799

NBSIR 83-~~2687~~

National Bureau
of Standards

AN OVERVIEW OF ARTIFICIAL INTELLIGENCE AND ROBOTICS

VOLUME1—ARTIFICIAL INTELLIGENCE

PART A—THE CORE INGREDIENTS

January 1984

Prepared for

National Aeronautics and Space Administration Headquarters
Washington, D.C. 20546

NBSIR 83-2687

AN OVERVIEW OF ARTIFICIAL INTELLIGENCE AND ROBOTICS

VOLUME1—ARTIFICIAL INTELLIGENCE

PART A—THE CORE INGREDIENTS

William B. Gevarter*

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Manufacturing Engineering
Industrial Systems Division
Metrology Building, Room A127
Washington, DC 20234

January 1984

Prepared for:
National Aeronautics and Space Administration
Headquarters
Washington, DC 20546

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

*Research Associate at the National Bureau of Standards Sponsored by NASA Headquarters

PREFACE

Artificial Intelligence (AI) is a field with over a quarter of a century of history. However, it wasn't until the dawn of the decade of the 80's that AI really burst forth—finding economic and popular acclaim. Intelligent computer programs are now emerging from the laboratory into practical applications.

This report endeavors to indicate what AI is, the foundations on which it rests, the techniques utilized, the applications, the participants and finally its state-of-the-art and future trends.

Due to the scope of AI, this volume is issued in three parts:

Part A: The Core Ingredients

I. Artificial Intelligence—What It Is
II. The Rise, Fall and Rebirth of AI
III. Basic Elements of AI
IV. Applications
V. The Principal Participants
VI. State-of-the-Art
VII. Towards the Future

Part B: Fundamental Application Areas - NASA TM 85838, October 1983

I. Expert Systems
II. Computer Vision
III. Natural Language Processing
IV. Speech Recognition and Speech Understanding
V. Speech Synthesis
VI. Problem-Solving and Planning

Part C: Basic AI Topics - NASA TM 85839, December 1983

I. Artificial Intelligence and Automation
II. Search-Oriented Automated Problem Solving and Planning
III. Knowledge Representation
IV. Computational Logic

To facilitate the use of this report by those readers who need to find information on a specific topic area, each chapter has been made reasonably self-contained. Thus, a certain amount of repetition of information between chapters exists, but should pose little problem for those who desire to read the entire report. This volume is part of the NBS/NASA series of overviews on AI and Robotics.

ACKNOWLEDGMENTS

I wish to thank the many people and organizations who have contributed to this report, both in providing information, and in reviewing the report and suggesting corrections, modifications and additions. I particularly would like to thank Jerry Cronin of U.S. Army Signal Warfare Lab., Ted Hopp and Len Haynes of NBS, Bob Hong and his associates at Grumman Aerospace Corp., Karen Hagedorn of Symbolics, Mache Creeger of LISP Machine, Inc., Fred Blair of IBM T. J. Watson Research Center, Jude Franklin of the U.S. Navy Center for Applied Research in AI, and David H. Brown of SRI International for their review of this report and their many helpful suggestions. However, the responsibility of any remaining errors or inaccuracies must remain with the author.

It is not the intent of NASA or the National Bureau of Standards to recommend or endorse any of the manufacturers or organizations named in this report, but simply to attempt to provide an overview of the AI field. However, in a diverse and rapidly changing field such as AI, important activities, organizations and products may not have been mentioned. Lack of such mention does not in any way imply that they are not also worthwhile. The author would appreciate having any such omissions or oversights called to his attention so that they can be considered for future reports.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF TABLES

I. ARTIFICIAL INTELLIGENCE—WHAT IT IS

Definition

Artificial Intelligence* (AI) is an emerging technology that has recently attracted considerable publicity. Many applications are now under development. One simple view of AI is that it is concerned with devising computer programs to make computers smarter. Thus, research in AI is focused on developing computational approaches to intelligent behavior. This research has two goals: 1) making machines more useful and 2) understanding intelligence. This report is primarily concerned with the first goal.

The computer programs with which AI is concerned are primarily symbolic processes involving complexity, uncertainty, and ambiguity. These processes are usually those for which algorithmic solutions do not exist and search is required. Thus, AI deals with the types of problem solving and decision making that humans continually face in dealing with the world.

This form of problem solving differs markedly from scientific and engineering calculations that are primarily numeric in nature and for which solutions are known that produce satisfactory answers. In contrast, AI programs deal with words and concepts and often do not guarantee a correct solution—some wrong answers being tolerable as in human problem solving.

Table I-1 provides a comparison between AI and conventional computer programs. A key characteristic of AI programs is "heuristic search." Baraiko (1982, p. 448) quotes Minsky as saying "If you can't tell a computer how best to do something, program it to try many approaches." However, in complex problems the number of possible solution paths can be enormous. Thus, AI problem solving is usually guided by empirical rules—rules of thumb—referred to as "heuristics"—which help constrain the search.

TABLE I-1. Comparison of AI with Conventional Programming.

Artificial Intelligence	Conventional Computer Programming
• Primarily symbolic processes	• Often primarily numeric
• Heuristic search (solution steps implicit)	• Algorithmic (solution steps explicit)
• Control structure usually separate from domain knowledge	• Information and control integrated together
• Usually easy to modify, update and enlarge	• Difficult to modify
• Some incorrect answers often tolerable	• Correct answers required
• Satisfactory answers usually acceptable	• Best possible solution usually sought

*Also sometimes referred to as machine intelligence or heuristic programming. The relationship of AI to automation is discussed in Chapter I of Part C of this report.

Another aspect of AI programs is the extensive use of "domain knowledge." Intelligence is heavily dependent on knowledge. This knowledge must be available for use when needed during the search. It is common in AI programs to separate this knowledge from the mechanism that controls the search. In this way, changes in knowledge only require changes in the knowledge base. In contrast, domain knowledge and control in conventional computer programs are integrated together. As a result, conventional computer programs are difficult to modify, as the implications of the changes made in one part of the program must be carefully examined for the impacts and the changes required in other parts of the program.

The Basic Elements of AI

Nilsson (1982, see also Brown, 1981), a pioneer in AI and currently head of the SRI AI Center, likes to characterize the components of AI in terms of what he calls the onion model (see Figure 1). The inner ring depicts the basic elements from which the applications shown in the next ring are composed. We will first consider the quadrant designated as heuristic search.

Heuristic Search

Much of the early work in AI was focused on deriving programs that would search for solutions to problems. Note that every time one makes a decision, the situation is changed opening up new opportunities for further decisions. Therefore there are always branch points. Thus, one of the usual ways of representing problem solving in AI is in terms of a tree (see, e.g., Figure 1, Chapter III), starting at the top with an initial condition and branching every time a decision is made. As one continues down the tree many different decision possibilities open up, so that the number of branches at the bottom can get to be enormous for problems requiring many solution steps. Therefore, some way is needed to efficiently search the trees.

Initially, there were "blind" methods for searching trees. These were orderly search approaches that assured that the same solution path would not be tried more than once. However for problems more complex than games and puzzles, these approaches were inadequate. Therefore, rules of thumb (empirical rules), referred to as "heuristics," were needed to aid in choosing the most likely branches, so as to narrow the search. As an example, a simple heuristic to help choose which roads to follow when driving in the evening on back roads from Washington, DC to San Francisco is: "head for the setting sun." This may not produce the most optimum path, but can serve to help advance one toward one's goal. Heuristic rules like this can help guide search—reducing search enormously.

Knowledge Representation

Early on, AI researchers discovered that intelligent behavior is not so much due to the methods of reasoning, as it is dependent on the knowledge one has to reason with. (As humans go through life they build up tremendous reservoirs of knowledge.) Thus, when substantial knowledge has to be brought to bear on a problem, methods are needed to efficiently model this knowledge so that it is readily accessible. The result of this emphasis on knowledge is that knowledge representation is one of the most active areas of research in AI today. The needed knowledge is not easy to represent, nor is the best representation obvious for a given task.

Common Sense Reasoning and Logic

AI researchers found that common sense (virtually taken for granted in humans) is the most difficult thing to model in a computer. It was finally concluded that common sense is low level reasoning, based on a wealth of experience. In acquiring common sense we learn to expect that when we drop something it falls, and in general what things to anticipate in everyday events. How to represent common sense in a computer is a key AI issue that is unlikely to be soon solved.

Another area that is very important in AI is logic. How do we deduce something from a set of facts? How can we prove that a conclusion follows from a given set of premises? Computational logic was one of the early golden hopes in AI to provide a universal problem solving method. However, solution convergence proved to be difficult with complex problems, resulting in a diminishing of interest in logic. Logic is now enjoying a revival based on new formulations and the use of heuristics to guide solutions.

AI Languages and Tools

In computer science, specific high level languages have been developed for different application domains. This has also been true for AI. Currently, LISP and PROLOG are the principal AI programming languages. To date, LISP (List Processing Language, developed in the late 50's by John McCarthy then at M.I.T.) has been the prime language in the U.S. for AI. Utilizing LISP, software tools have been devised for expressing knowledge, formulating expert systems, and basic programming aids.

Principal AI Application Areas

Based on these basic elements, Nilsson identified four principal AI application areas (shown in the outer ring of Figure I-1.)

Natural Language Processing (NLP)

NLP is concerned with natural language front ends to computer programs, computer-based speech understanding, text understanding and generation, and related applications. A detailed overview of NLP is given in Gevarter (1983).

Computer Vision

Computer Vision is concerned with enabling a computer to see—to identify or understand what it sees, to locate what it is looking for, etc. A detailed overview of Computer Vision is given in Gevarter (1982B).

Expert Systems

Expert Systems is perhaps the "hottest" topic in AI today. How do we make a computer act as if it was an expert in some domain? For example, how do we get a computer to perform medical diagnosis or VLSI design? A detailed overview of Expert Systems is given in Gevarter (1982A).

Problem Solving and Planning

There are many problems for which there are no experts, but nevertheless computer programs for their solutions are needed. In addition there are some basic planning systems that are more concerned with solution techniques than with knowledge. A comprehensive overview of problem solving and planning is given in Chapter VI of Part B and Chapter II of Part C of this report.

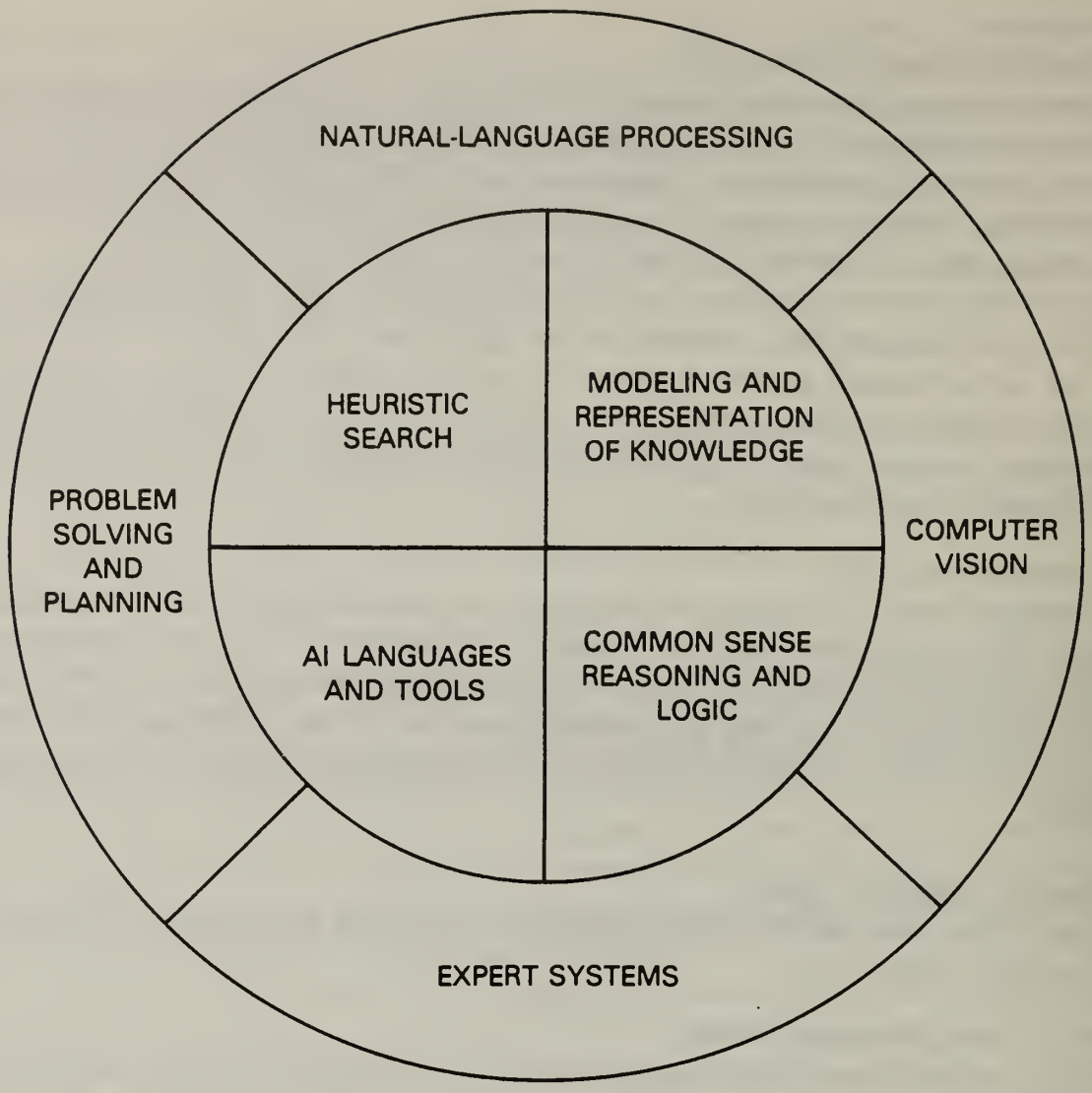


Figure I-1. Elements of AI.

Is AI Difficult?

The popular view that the study of AI is difficult has been partly due to the awe associated with the notion of intelligence*. It has also been due to the nomenclature used in AI and to the large size of some of the AI computer programs. However, the basic ideas of AI are readily understandable, even though in complex applications, the "bookkeeping" associated with such programs can be arduous.

Before we go into details on these basic ideas, it is illuminating to review the history of AI, which we will do in Chapter II. Then in Chapter III, the basic elements of AI will be discussed in some

*Indeed researchers can not even agree on a definition of intelligence itself. One paradoxical definition is "Intelligence is what an intelligence test measures."

detail (and elaborated upon further in Part C of this report). Chapters IV and V present the principle players in the AI drama and an indication of the unfolding applications area. (A more complete overview of AI applications is given in Part B of this report.) The current state of the art in AI and its future directions will be covered in Chapters VI and VII.

A list of sources for further information and a glossary of AI terms are provided at the end of this report.

CHAPTER I REFERENCES

- Boraiko, A.A., "The Chip," *National Geographic,* Oct. 1982, pp. 421-456.
- Brown, D.R., "Applications of Artificial Intelligence in Space Tracking and Data Systems," SRI Project 2203, SRI International, Menlo Park, CA, April 1981.
- Gevarter, W.B., *An Overview of Computer-Based Natural Language Processing*, NASA TM 85635 (also NBS 83-2687), NASA, Wash., D.C., April 1983.
- Gevarter, W.B., *An Overview of Computer Vision*, NBSIR-2582, National Bureau of Standards, Wash., D.C., September 1982.
- Gevarter, W.B., *An Overview of Expert Systems*, NBSIR-2505, National Bureau of Standards, Wash., D.C., May 1982 (Revised October 1982).
- Nilsson, N.J., "Artificial Intelligence: Engineering, Science or Slogan," *AI Magazine,* Vol. 3, No. 1, Winter 1981/1982, pp. 2-9.

II. THE RISE, FALL AND REBIRTH OF AI*

The First 15 Years

In 1956, ten scientists convened a conference at Dartmouth College from which emerged the present field of AI. The predictions made by those scientists were that in 25 years, we would all be involved in recreational activities, while computers would be doing all the work. In 1981, at the International Joint Conference on AI in Vancouver, Canada, a panel of five of these same scientists recalled that conference and their over-optimistic forecasts.

In 1956, it was assumed that intelligent behavior was primarily based on smart reasoning techniques and that bright people could readily devise ad hoc techniques to produce intelligent computer programs.

Figure II-1 lists some of the key AI activities during the first 15 years. The major initial activity involved attempts at machine translation. It was thought that natural language translation could be readily accomplished using a bilingual dictionary and some knowledge of grammar. However, this approach failed miserably because of factors such as multiple word senses, idioms and syntactic

*An interesting history of the early years of AI is given by McCorduck (1979).

Activities

- Attempts at Machine Translation
- ELIZA—Key Word and Template Matching
- Symbolic Integration
- Game Playing—Checkers, Chess
- Pattern Recognition
- Computational Logic
- General Problem Solver

Lessons Learned

- AI Much More Difficult Than Expected
- Heuristic Search Required To Limit Combinatorial Explosion
- Lack of Contextual Knowledge Severely Limits Capability
- Expectation Is a Human Characteristic of Intelligence
- Difficult To Handle a Broad Domain (e.g., Common Sense)

Figure II-1. A Condensed History of AI—1956-1970.

ambiguities. A popular story is that the saying "The spirit is willing but the flesh is weak," when translated into Russian and back again into English, came out "The wine is good but the meat is spoiled." Schwartz (1980, p. 27) reports that "Twenty million dollars of mechanical translation brought results so disappointing, that . . . by 1967 opinion had soured so dramatically that the National Academy of Sciences all but created a tombstone over the research." In fact it has only been recently that substantial work in mechanical language translation has reappeared.

Weizenbaum (1966) at MIT designed a natural language understanding program that simulated a non-directive psychotherapist. The program (ELIZA) bluffed its way through the interaction by picking up on key words and providing stock answers. When it did not find a recognizable key word, it would select a reply such as "Please continue." Though Weizenbaum wrote the program in part to show how ridiculous it was to expect true natural language understanding by a machine, the program nevertheless became popular and some of its basic techniques are used in commercial Natural Language Interfaces today.

In 1961, Slagle at M.I.T. devised a heuristic computer program to do symbolic integration. This proved to be the forerunner of a successful series of symbolic mathematical programs culminating in MACSYMA, in use at M.I.T. today and available over the "ARPA Net" to other AI researchers.

Game playing was also one of the early areas of AI research, with Samuel's (1963) work at IBM on machine learning in checkers proving to be one of the early successes.

Solving puzzles was another area of early success in AI, leading to the development of problem solving techniques based on 1) search and 2) reducing difficult problems into easier subproblems.

Early work in vision involved image processing and pattern recognition (which was concerned with classifying two-dimensional patterns). Pattern recognition split off from AI and became a field in itself, but now the two disciplines have become much more unified.

The pioneering work in computer vision was Robert's (1965) program designed to understand polyhedral block scenes. This program found the edges of the blocks using the spatial derivatives of image intensity, and from the resulting edge elements produced a line drawing. It then utilized simple features, such as the numbers of vertices, to relate the objects in the line drawing to stored 3D models of blocks. The resulting candidate model was then scaled, rotated, and projected onto the line drawing to see if the resultant match was adequate for recognition.

Another important area was computational logic. Resolution, an automatic method for determining if the hypothesized conclusion indeed followed from a given set of premises, was one of the early golden hopes of AI for universal problem solving by computer. Using resolution, Green (1969) devised a general-purpose, question-answering system, QA3, that solved simple problems in a number of domains such as robot movements, puzzles, and chemistry. Unfortunately, resolution, though it guarantees a solution, devises so many intermediate steps that turn out not to be needed for the final solution, that for large problems its use results in a combinatorial explosion of search possibilities.

Another approach, originally thought to have broad applicability, was the General Problem Solver (GPS) devised by Newell et al. (1960). The generality resulted from GPS being the first problem solver to separate its problem-solving methods from knowledge of the specific task currently being considered. The GPS approach was referred to as "means-ends analysis." The idea was that the differences between the current problem state and the goal state could be measured and classified into types. Then, appropriate operators could be chosen to reduce these differences, resulting in new problem states closer to the goal states. This procedure would then be iteratively repeated until the goal was reached. The series of operators used would then form the solution plan. Unfortunately, classifying differences and finding appropriate operators turned out to be more difficult than expected for non-trivial problems. In addition, computer running times and memory requirements rapidly become excessive for the more difficult problems.

AI proved much more difficult than originally expected. By 1970, AI had had only limited success. Natural Language Translation had already collapsed. "Toy" problems or well constructed problems such as games proved tractable, but real complex problems proved to be beyond the techniques thus far devised, or resulted in combinatorially explosive search that exceeded the then current computer capabilities. Similarly, real world computer vision efforts tended to be overwhelmed by the noise and complexities in real scenes.

In 1971, Sir Lighthill of Cambridge University was called upon by the British Government to review the AI field. The Lighthill Report (1972) found that "In no part of the field have the discoveries made so far produced the major impact that was promised." Further, he found that respected AI scientists were then predicting that ". . . possibilities in the 1980's include an all-purpose intelligence on a human-scale knowledge base; that awe-inspiring possibilities suggest themselves based on machine intelligence exceeding human intelligence by the year 2000"—the same sort of forecasts as were made 15 years earlier. Lighthill saw no need for a separate AI field and found no organized body of techniques that represented such a field. He felt that the work in automation and computer science would naturally come together to bridge whatever gap existed. The Lighthill report eventually brought work in AI in England to a virtual halt and cast a pall over AI work in the U.S.

However, the AI efforts of the 1950's and 1960's were not without merit. A great deal was learned about what really had to be done to make AI successful.

It was found that expectation is a human characteristic of intelligence. That perception, both visual and in language, is based upon knowledge, models and expectations of the perceiver. Thus communication via language was found to be based upon shared knowledge between the participants and that only cues are needed to actualize the models (in the receiver's head) from which to construct the complete message.

Thus in attempting communication or problem solving, lack of contextual knowledge was found to severely limit capability. Reasoning techniques alone proved inadequate. Knowledge is central to intelligence. Lacking this knowledge, it is difficult to handle a broad domain. An example is "common sense," found to be elementary reasoning based upon massive amounts of experiential knowledge.

It was also found that heuristics are necessary to guide search to overcome the combinatorial explosion of possible solutions that pervade complex problems—for each time one makes a decision, one opens up new possibilities.

The Decade of the 70's

As indicated in Figure II-2, in the 1970's AI researchers began to capitalize on the lessons learned. New knowledge representation techniques appeared. Search techniques began to mature. Interactions with other fields such as medicine, electronics and chemistry took place. Feasible approaches were demonstrated for language processing, speech understanding, computer vision, and computer programs that could perform like experts.

SHRDLU was a natural language program at M.I.T. devised by Terry Winograd (1972) to interface with an artificial "blocks world." It was the first program to successfully deal in an integrated way with natural language by combining syntactic and semantic analysis with a body of world knowledge.

From 1971-1976, ARPA sponsored a five-year speech understanding program. HEARSAY II at Carnegie Mellon University was a winner, being able to understand sentences, with 90% accuracy, from continuous speech based on a 1000-word vocabulary. (The "blackboard" system architecture, devised for HEARSAY II to deal with multiple knowledge sources, has since found use in other AI applications.) A compiled network architecture system called HARPY, which handled the

Activities

- Feasible Approaches Demo'd for:
  - Language Processing
  - Computer Vision
  - Expert Systems
  - Speech Understanding
- New Knowledge Representation Techniques Appear
- Search Techniques Begin To Mature
- Interaction with Other Fields Takes Place

Lessons Learned

- Knowledge Central to Intelligence
- Future Complex Systems Proved Feasible

Figure II-2. The Decade of the 70's.

same vocabulary as HEARSAY II, was able to achieve a 95% accuracy. (A more detailed review is given in Part B of this report.)

At SRI, Gleason and Agin (1979) developed the SRI Vision Module as a prototype system for use in industrial vision systems. This system, which used special lighting to produce a binary image (silhouette) of an industrial workpiece, was able to extract edges by a simple continuous scan process, and was to prove the basis for several sophisticated commercial vision systems.

In the 70's, following an earlier successful effort called DENDRAL, a variety of prototype computer programs—called Expert Systems—designed to capture and utilize the expertise of a human expert in a narrow domain (such as medical diagnosis, crystallography, electrical circuitry, prospecting, etc.) made their appearance. MYCIN, a medical diagnosis and treatment consultant, devised by Shortliffe (1976) at Stanford University has been one of the most publicized.

Thus, the 70's found the AI research community developing the basic tools and techniques needed, and demonstrating their applicability in prototype systems. Future complex systems were proved feasible. The emphasis on knowledge, as essential to intelligence, led to the subfield of "Knowledge Engineering" associated with the building of expert systems.

1980 to the Present

The decade of the 70's set the framework from which the successes of the 80's emerged. In the 80's, expert systems proliferated. Dozens of prototype expert systems were devised in such areas as medical diagnosis, chemical and biological synthesis, mineral and oil exploration, circuit analysis, tactical targeting, and equipment fault diagnosis.

But the big news of the 80's (see Figure II-3) is that AI has gone commercial. AI companies (founded mostly by AI researchers) have formed to exploit applications. Computer, electronic, oil, and large diversified companies have set up AI groups. The military has also joined the fray, setting up their own AI groups and seeking early applications. The U.S. Defense Science Board views AI as one of the technologies that has the potential for an order of magnitude improvement in mission effectiveness.

Activities

- Expert Systems Proliferate
- AI Goes Commercial
  - Expert Systems: RI, DIP-METER ADVISOR, MOLGEN
  - Natural Language Front Ends—INTELLECT
  - Speech Output—Speak and Spell
  - Vision Systems
  - AI Groups and Companies Form To Exploit Applications
  - LISP Machines Become Available
- AI Technology Becoming Codified
  - AI Handbook
  - Individual Technology Texts: Natural Language, Vision, etc.
  - NBS/NASA Overviews

Conclusions

- AI Tools and Systems Become Available
- Logic Systems (Heuristically Guided) Reemerge—PROLOG
- AI Techniques Sufficiently Perfected for Early Applications

Figure II-3. 1980-Present.

In the expert systems area, DEC reports that RI—a system designed to configure VAX computer systems—is already saving them some 20 million dollars a year. MOLGEN—a system for planning molecular genetic experiments—is in regular commercial use. Shlumberger—a multi-billion dollar oil industry advisory company—seeing AI as a key to the company's growth in the 80's, has established four separate AI groups. The Palo Alto group has already created the expert system DIPMETER ADVISOR, to evaluate oil-drilling core samples.

In natural language front ends, some half dozen systems are now commercially available, with INTELLECT from Artificial Intelligence Corporation already boasting well over a hundred installations.

Highlighted by Texas Instruments' Speak and Spell, many commercial speech output systems have appeared. Limited speech recognition systems are also on the market, some using signal processing rather than AI techniques.

Hundreds of companies are now involved in computer vision systems, with dozens of commercial products already on the market for simplified vision applications.

Personal computers that are specially designed to run LISP—the List Processing Language favored by the U.S. AI community—are now commercially available from several companies.

The other indication that AI has now emerged as a viable discipline is that the existing AI technology is now becoming codified and therefore made broadly available to everyone, not just the core group of several hundred researchers of the 70's.

ARPA sponsored a three volume *AI Handbook* which was published in 1981 and 1982. Individual technology texts—in Vision, Natural Language, Expert Systems and LISP—are beginning to appear in numbers.

NASA has sponsored this NBS set of overviews in Artificial Intelligence and Robotics. The other volumes of this series are:

An Overview of Artificial Intelligence and Robotics

Vol II—Robotics, NBSIR 82-2479, March 1982

Vol III—Expert Systems, NBSIR 82-2505, May 1982 (Revised Oct 1982)

Vol IV—Computer Vision, NBSIR-82-2582, Sept 1982

Vol V—Computer-based Natural Language Processing, NBSIR 83-2687, Apr 1983. NASA TM 85635, Apr 1983.

Computer software tools for structuring knowledge and constructing expert systems are also becoming available.

In 1982, the Japanese officially began a 10 year, one-half billion dollar, research project to create a *Fifth Generation Computer.* The main features of this computer are that it is to have 1) intelligent interfaces (speech, text, graphics, etc.) 2) knowledge base management and 3) automatic problem solving and inference capabilities. All these capabilities are predicated on the use of AI techniques. The machine itself is visualized as a non-Von Neumann computer featuring parallel processing and having the capability of one billion logical inferences per second.

The Japanese are now considering the European AI language—PROLOG (Programming in Logic)—as the basis for their machine. Using PROLOG, logic problem-solving systems (heuristically guided) are reemerging (from the earlier failure of pure resolution) to handle complex problems.

With the advent of the Japanese Fifth Generation Computer Project, European nations, such as France and Britain, as well as the U.S., are putting renewed effort into their AI activities (Warren, 1982).

In summary then, we can conclude that AI tools and systems are now becoming available, and AI techniques are now sufficiently perfected for early applications. Further, the importance of AI is being recognized internationally and substantial sums of money in the U.S. and abroad are now beginning to be committed to developing AI applications.

CHAPTER II REFERENCES

- Gleason, G.J. and Agin, G.J., "A Modular System for Sensor-Controlled Manipulation and Inspection," *Proc. of the Ninth Inter. Symp. of Ind. Robots,* SME and RIA, Wash., DC, 1979, pp. 57-70.
- Green, C.C., "The Application of Theorem-Proving to Question Answering Systems," *IJCAI-1,*1969, pp. 219-237.
- Lighthill, J., "Artificial Intelligence: A General Survey," Scientific Research Council of Britain, SRC 72-72, March 1972.
- McCorduck, P., *Machines Who Think,* San Francisco, W.H. Freeman, 1979.
- Newell, A., Shaw, J.C. and Simon, H.A., "A Variety of Intelligent Learning in a General Problem Solver," In M.C. Yovits and S. Cameron (Eds.), *Self Organizing Systems,* New York, Pergamon Press, 1960, pp. 153-189.
- Roberts, L., "Machine Perception of Three-Dimensional Solids," In J. Tippitt (Ed.), *Optical and Electro-Optical Information Processing,* Cambridge, Mass, MIT Press, 1965, pp. 159-197.
- Samuel, A.L., "Some Studies in Machine Learning Using the Game of Checkers," In E.A. Feigenbaum and J. Feldman (Eds.), *Computers and Thought,* New York, McGraw Hill, 1963.
- Schwartz, R.D., "Refocus and Resurgence in Artificial Intelligence," *IEEE Proc. of the Nat. Aerospace and Electronic Conference: NACON-1980,* Dayton, Ohio, May 20-22.
- Shortliffe, E.H., *Computer-Based Medical Consultations*, MYCIN, New York: American Elsevier, 1976.
- Slagle, J.R., "A Heuristic Program that Solves Symbolic Integration in Freshman Calculus: Symbolic Automatic Integrator (SAINT)," Rep. No. 5G-001, Lincoln Lab., M.I.T., Cambridge, Mass., 1961.
- Weizenbaum, J., "Eliza—A Computer Program for the Study of Natural Language Communication Between Man and Machine," *CACM* 9, 1966, pp. 36-45.
- Winograd, T., *Understanding Natural Language,* New York, Academic Press, 1972.
- Warren, D.H.D., "A View of the Fifth Generation and Its Impact," *The AI Magazine,* Vol. III, No. 4, Fall 1982, pp. 34-39.

Heuristic Search

AI problem solving can often be viewed as a search among alternative choices. It is thus possible to represent the resulting search space as a hierarchical structure called a tree, an example of which is shown in Figure III-1. (Figure III-1 is a search tree for the elementary problem of finding the simplest route, from city A to the destination city D, from among the network of roads illustrated by the state graph of Figure III-2.) The solution paths run from the initial state (root node) along the branches of the tree and terminate on the leaves (terminal nodes) labeled "goal state."

For a large complex problem, it is obviously too cumbersome to explicitly draw such trees of all the possibilities and directly examine them for the best solution. Thus, the tree is usually implicit; the computer generating branches and nodes as it searches for a solution.

In searching for a solution we may reason forward as in Figure III-1, or backward from the goal (searching an equivalent tree where the root node is the goal).

For fairly simple problems, a straightforward, but time-consuming, approach is blind search, where we select some ordering scheme for the search and apply it until the answer is found. There

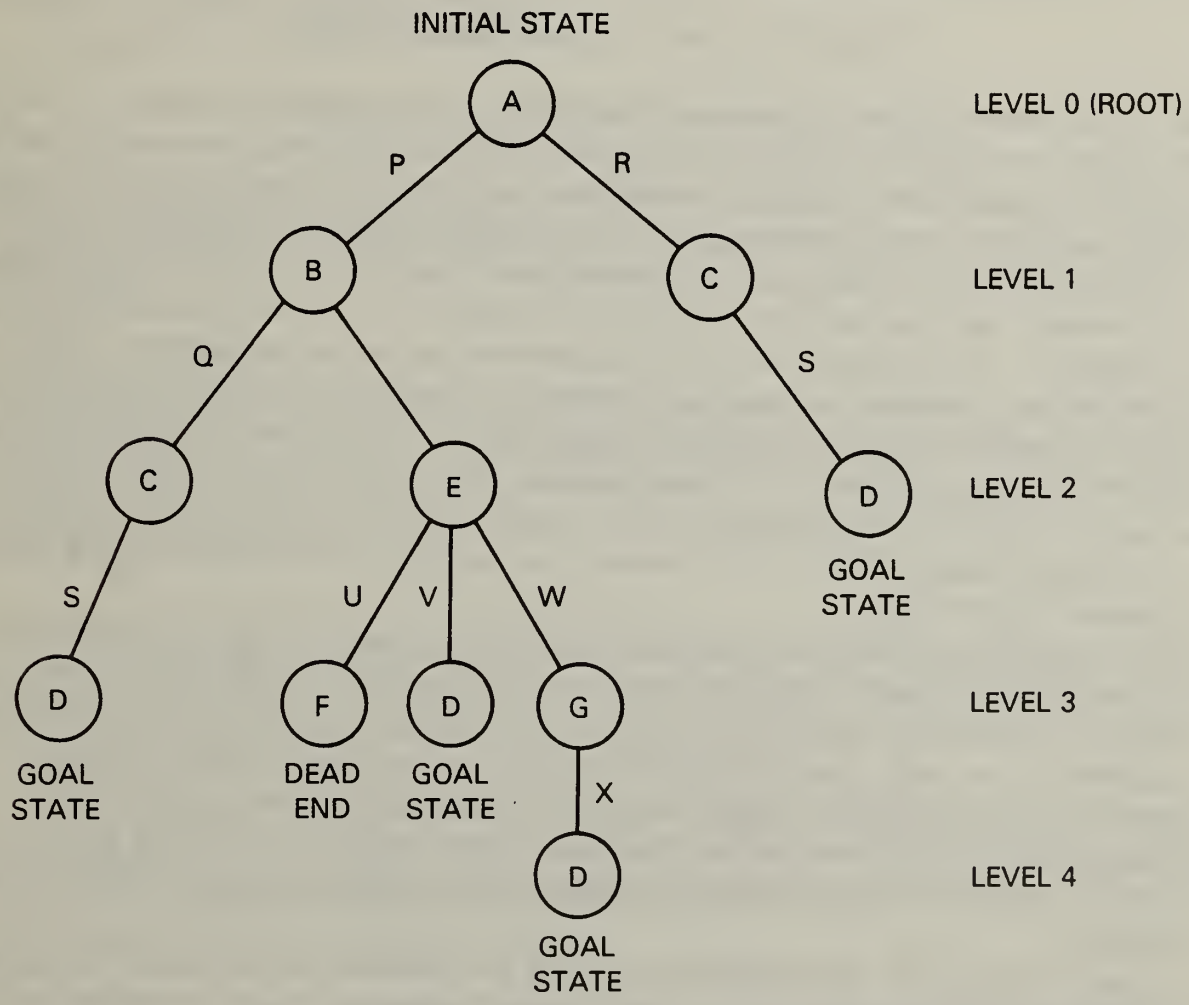


Figure III-1. Tree Representation of Paths Through the State Graph of Figure III-2.

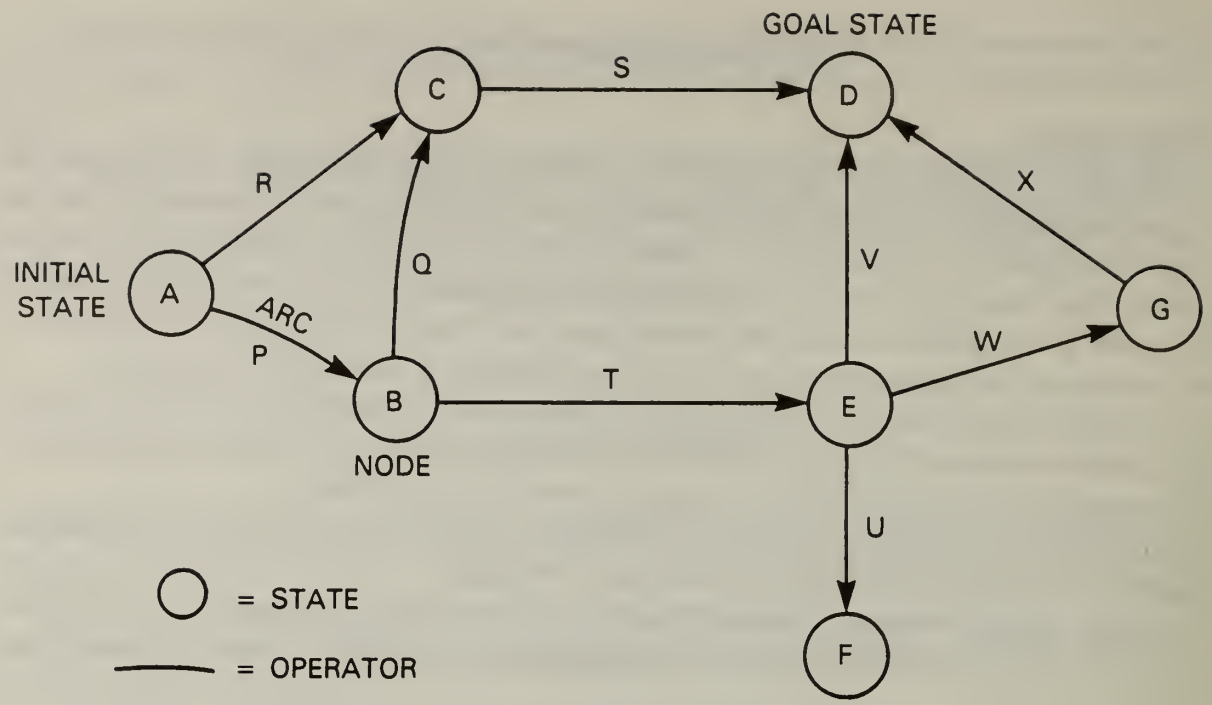


Figure III-2. State Graph for a Simple Problem.

are two common blind search procedures, breadth-first search and depth-first search. In breadth-first search, the nodes of the search tree are generated and examined level by level starting with the root node. In a depth-first search, a new node (at the next level) is generated from the one currently being examined, the search continuing in this way deeper and deeper until forced to backtrack.

Blind search does not make any use of knowledge about the problem to guide the search. In complex problems, such searches often fail, being overwhelmed by the combinatorial explosion of possible paths. If on the average there are n possible operators that can be applied to a node, and the search space is searched to a depth of d, then the size of the search space tends to grow in relation to n^d. Heuristic methods have been designed to limit the search space by using information about the nature and structure of the problem domain. Heuristics are rules of thumb, techniques or knowledge that can be used to help guide search. Heuristic search is one of the key contributions of AI to efficient problem solving. It often operates by generating and testing intermediate states along a potential solution path.

One straightforward method for choosing paths by this approach is to apply an evaluation function to each node generated and then pursue those paths that have the least total expected cost. Typically, the evaluation function calculates the cost from the root to the particular node that we are examining and, using heuristics, estimates the cost from that node to the goal. Adding the two produces the total estimated cost along the path, and therefore serves as a guide as to whether to proceed along that path or to continue along another, more promising, path among those thus far examined. However, this may not be an efficient approach to minimize the search effort in complex problems.

Search techniques are now relatively mature and are codified in the *AI Handbook* (Barr and Feigenbaum, 1981) and various AI texts. More detailed information on search techniques are given in Chapter II of Part C of this report.

Knowledge Representation*

The purpose of knowledge representation (KR) is to organize required information into a form such that the AI program can readily access it for making decisions, planning, recognizing objects and situations, analyzing scenes, drawing conclusions, and other cognitive functions. Thus knowledge representation is especially central to expert systems, computational vision, and natural language understanding.

Representation schemes are classically classified into declarative and procedural ones. Declarative refers to representation of facts and assertions, while procedural refers to actions, or what to do. A further subdivision for declarative ("object oriented") schemes includes relational (semantic network) schemes and logical schemes.

The principal KR schemes are briefly discussed in the following paragraphs.

Logical Representation Schemes

The principal method for representing a knowledge base logically is to employ First Order Predicate Logic. In this approach, a knowledge base (KB) can be viewed as a collection of logical formulas which provide a partial description of the world. Modifications to the KB result from additions or deletions of logical formulas.

An example of a logical representation is:

IN (SHUTTLE ORBIT) = The shuttle is in orbit

Logical representations are easy to understand and have available sets of inference rules needed to operate upon them. A drawback of logical representation is its tendency to consume large amounts of memory.

Semantic Networks

A semantic network is an approach to describing the properties and relations of objects, events, concepts, situations or actions by a directed graph consisting of nodes and labelled edges (arcs connecting nodes). A simple example is given in Figure III-3. Because of their naturalness, semantic networks are very popular in AI.

Procedural Representations and Production Systems

In procedural representations, knowledge about the world is contained in procedures—small programs that know how to do specific things (how to proceed in well specified situations). Classification of procedural representation approaches are based on the choice of activation mechanisms for the procedures, and the forms used for the control structures.

The two common approaches consist of procedures representing major chunks of knowledge — subroutines—and more modular procedures, such as the currently popular "production rules." The common activation mechanism for procedures is matching the system state to the preconditions needed for the procedure to be invoked.

*A more detailed presentation of knowledge representation is given in Chapter III of Part C.

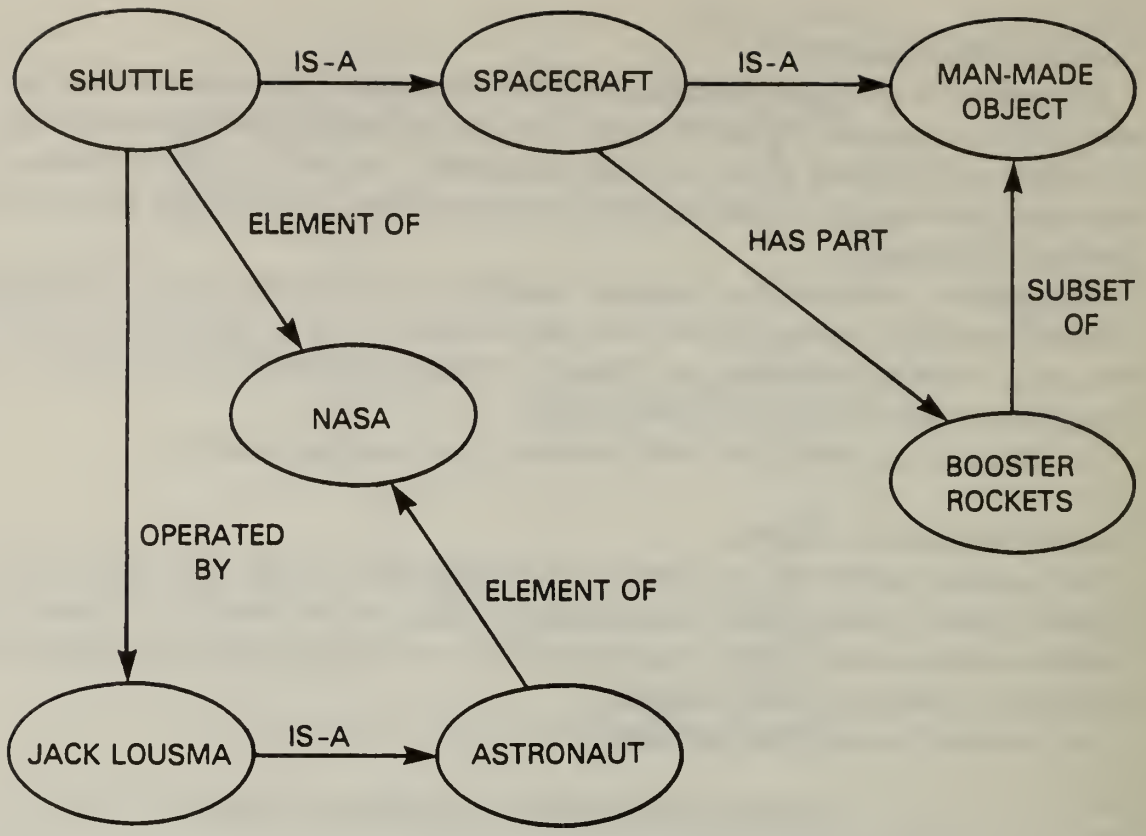


Figure III-3. Simple Example of a Semantic Network.

Production rules (PR) are characterized by a format of the type:

Pattern, Action
If, Then
Antecedent, Consequent
Situation, Procedure

e.g., If the shuttle power supply fails,
and a working power supply is available,
and the situation causing the problem no longer exists,
Then switch to the backup.

Because of their modular representation of knowledge and their easy expansion and modifiability, PR's are now probably the most popular AI knowledge representation, being chosen for most expert systems.

Analogical or Direct Representations

In many instances it is appropriate to use natural representations such as an array of brightness values for an image, or a further reduced "sketch map" of the scene delineations in a computer vision system. These natural representations are useful in computational vision, spatial planning, geometric reasoning and navigation.

This form of representation has the advantages of being easy to understand, simple to update, and often allows important properties to be directly observed, so that they don't have to be inferred.

Property Lists

One approach to describe the state of the world is to associate with each object a property list, that is a list of all those properties of the object pertinent to the state description. The state, and therefore the object properties, can be updated when a situation is changed.

Frames and Scripts

A large proportion of our day-to-day activities are concerned with stereotyped situations such as going to work, eating, shopping, etc. Minsky (1975) conceived of "frames," which are complex data structures for representing stereotyped objects, events or situations. A frame has slots for objects and relations that would be appropriate to the situation. Attached to each frame is information such as:

—how to use the frame

—what to do if something unexpected happens

—default values for slots.

Frames can also include procedural as well as declarative information. Frames facilitate expectation-driven processing—reasoning based on seeking confirmation of expectations by filling in the slots. Frames organize knowledge in a way that directs attention and facilitates recall and inference.

An example of a frame is:

Airplane Frame:
Type:
range: (fighter, transport, trainer, bomber, light plane, observation)
Manufacturer:
range: (McDonnell-Douglas, Boeing . . .)
Empty Weight:
range: (500 lbs to 250,000 lbs)
Gross Weight:
range: (500 lbs to 500,000 lbs)
if needed: (1.6 × empty weight)
Max Cruising Range:
if needed: (look up in table cruising range appropriate to type and gross weight)
Number of Cockpit Crew:
range: (1 to 3)
default: 2

Scripts are frame-like structures designed for representing stereotyped sequences of events such as eating at a restaurant or a newspaper report of an apartment fire.

Semantic Primitives:

For any knowledge representation scheme, it is necessary to define an associated vocabulary. For semantic nets, there has been a real attempt to reduce the relations to a minimum number of terms (semantic primitives) that are non-overlapping. A similar effort has emerged for natural language understanding, for which several attempts have been made to describe all of the world's aspects in terms of primitives that are unique, unambiguous representations into which natural language statements can be converted for later translation into another language or for other cognitive actions.

Schank (see, e.g., Schank and Riesbeck, 1981) has developed a "conceptual dependency" theory for natural language, in an attempt to provide a representation of all actions in terms of a small number of primitives. The system relies on 11 primitive physical, instrumental and mental ACT's (propel, grasp, speak, attend, etc.), plus several other categories, or concept types. There are two basic kinds of combinations or conceptualizations. One involves an actor doing a primitive ACT; the other involves an object and a description of its state. Attached to each primitive act is a set of inferences that could be associated with it.

An example of a representation in conceptual dependency is:

"Armstrong flew to the moon."

Actor:	Armstrong
Action:	flew
Direction to:	the moon
From:	unknown

Computational Logic*

Logic is a formal method of reasoning. Graham (1979, p. 163) observes that:

> Computational logic—doing logical reasoning with a computer—is based on what is traditionally known as symbolic logic, or mathematical logic. This, in turn, is divided into two (principal) parts, the simpler *"propositional* logic" and the more complex *"predicate logic"*.

Propositional Logic

In logic, a "proposition" is simply a statement that can be true or false. Rules used to deduce the truth (T) or falsehood (F) of new propositions from known propositions are referred to as "argument forms."

To make more interesting statements we can join propositions together with the following connectives:

Connective	Symbol	Meaning
And	$\wedge$ or $\cap$	both
Or	$\vee$ or $\cup$	either or both
Not	$\neg$ or $\sim$	the opposite
Implies	$\supset$ or $\rightarrow$	If the preceding term is true, then the term following will also be true
Equivalent	$\equiv$	has the same truth value

The simplest argument form is the "conjunction," which utilizes the connective AND. It states that if proposition p is true and proposition q is true, then the conjunction "p AND q" are true. In symbolic form we have

$$\begin{array}{ll} p & \text{(premise)} \\ q & \text{(premise)} \\ \hline p \wedge q & \text{(conclusion).} \end{array}$$

That is, the conclusion is true if the premises are true.

Deduction involves deriving answers to problems based on a given set of premises. In mathematical logic, deductive procedures are sometimes referred to as "formal inference."

*A more complete discussion of computational logic is presented in Chapter IV of Part C.

A simple form of deduction is the "syllogism." For example:

All Greeks are mortal.
Socrates is a Greek.
Therefore, Socrates is mortal.

This type of reasoning can be represented as a mathematical form of argument called "Modus Ponens" (MP):

p	(premise)
p implies q	(premise)
q	(conclusion)

or in logical notation as:

$$(p \wedge \neg(p \rightarrow q)) \rightarrow q$$

Predicate Logic:

Propositional logic is limited in that it deals only with the T or F of complete statements. Predicate logic remedies this situation by allowing you to deal with assertions about items in statements, and allows the use of variables and functions of variables.

Propositions make assertions about items (individuals). A "Predicate" is the part of the proposition that makes an assertion about the individuals, and is written as:

$$\underbrace{\text{Predicate (individual, individual, . . .)}}_{\text{arguments of the predicate}}$$

For example,

"The box is on the table," (proposition) is denoted as:
ON (BOX, TABLE)

The predicate, together with its arguments, is a proposition. Any of the operations of propositional logic may be applied to it.

By including variables for individuals, Predicate Logic enables us to make statements that would be impossible in Propositional Logic. This can be further extended by the use of functions of variables. Finally, by use of the universal and existential quantifiers $\forall$ (for all) and $\exists$ (there exists), we arrive at First Order Predicate Logic (FOPL). FOPL permits rather general statements to be made, e.g.

For all Earth satellites, there exists a point y on the satellite that is closest to Earth:

$$\forall(x)\ \text{SATELLITE}(x) \rightarrow \exists(y)\ (\text{CLOSEST}(y, \text{Earth}) \wedge \text{ON}(y, x))$$

Various inference rules exist for the manipulation of quantifiers, the substitution of connectives, and other syntactic operations that assist in performing logical reasoning.

Logical Inference*

Resolution Method

Logical inference—reaching conclusions using logic—is normally done by "theorem proving." The most popular method for automatic theorem proving is the resolution procedure developed by Robinson (1965). This procedure is a general automatic method for determining if a hypothesized-conclusion (theorem) follows from a given a set of premises (axioms). First, using standard identities, the original premises and the conclusion to be proved are put into clause form. The conclusion to be proved is then negated. New clauses are then automatically derived using resolution and other procedures. If a contradiction is reached, then the theorem is proved.

Basically, resolution is the cancellation between clauses of a proposition in one clause with the negation of the same proposition in another clause.

Unfortunately, resolution has been unable to handle complex problems, as the search space generated by the resolution method grows exponentially with the number of formulas used to describe a problem. Thus for complex problems, resolution derives so many clauses not relevant to reaching the final contradiction, that it tends to use up the available time or memory before reaching a conclusion. Several domain-independent heuristics have been tried to constrain the search, but have proved to be too weak.

Factors Which Affect the Efficiency of Deductive Reasoning

Cohen and Feigenbaum (1982, pp. 80-81) state that "One kind of guidance that is often critical to efficient system performance is information about whether to use facts in a *forward-chaining* or *backward-chaining* manner . . . Early theorem-proving systems used every fact both ways leading to highly redundant searches . . ."

Another factor that can greatly affect the efficiency of the deductive reasoning is the way in which a body of knowledge is formalized. "That is, logically equivalent formalizations can have radically different behavior when used with standard deduction techniques."

Non-Resolution Theorem Proving

Cohen and Feigenbaum (1982, p. 94) observe that "In *non-resolution* or *natural deduction* theorem-proving systems, a proof is derived in a goal-directed manner that is natural for humans using the theorem prover. Natural-deduction systems represent proofs in a way that maintains a distinction between goals and antecedents, and they use inference rules that mimic the reasoning of human theorem-proving." They also tend to use domain-specific heuristics that help guide the search, and many proof rules to reduce goals to subgoals. The result is much more complex than the simpler, but less effective, resolution procedure.

Though requiring help from the programmer, the non-resolution Boyer and Moore (1979) Theorem Prover is one of the most powerful theorem provers available.

Special higher order languages (such as PROLOG, that helps structure the deduction problem and provides various built-in aids), coupled with domain-specific formulation and heuristic guidance rules, appears to be the direction that computational logic is proceeding in an attempt to handle complex real world problems.

**A more detailed discussion* is given in Chapter IV of Part C.

Common Sense Reasoning

Common sense reasoning is low level reasoning based on a vast amount of experiential knowledge. An example is reasoning about falling objects, based upon experience rather than upon Newton's Laws. The same sort of reasoning tells us what is the appropriate thing to do in everyday social situations. While it is a simple matter for humans, it is very difficult to achieve in present AI systems with current techniques.

Nilsson (1980, p. 154) states:

> . . . many common sense reasoning tasks that one would not ordinarily formalize can, in fact, be handled by predicate calculus theorem-proving systems. The general strategy is to represent specialized knowledge about the domain as predicate calculus expressions and to represent the problem or query as a theorem to be proved.

Nilsson (1980, p. 423) also observes that, "Much common sense reasoning (and even technical reasoning) is inexact in the sense that the conclusions and the facts and rules on which it is based are only approximately true. Yet people are able to use uncertain facts and rules to arrive at useful conclusions about everyday subjects such as medicine. A basic characteristic of such approximate reasoning seems to be that a conclusion carries more conviction if it is independently supported by two or more separate arguments." Several of the AI expert systems, such as Mycin and Prospector, make use of this approach.

AI approaches to approximate and plausible reasoning such as fuzzy set theory and default reasoning, and non-monotonic logic are given in Nilsson (1980) and Cohen and Feigenbaum (1982).

Non-Deductive Problem Solving Approaches*

Elements of a Problem Solver

All problems have certain common aspects: an initial situation, a goal (desired situation) and certain operators (procedures or generalized actions) that can be used for changing situations. In solving the problem, a control strategy is used to apply the operators to the situations to try to achieve the goal. This is illustrated in Figure III-4 where we observe a control strategy operating on the procedures to generate a sequence of actions (called a plan) to transform the initial conditions in the situation into the goal conditions. Normally, there are also constraints (specifying the conditions necessary for a specific procedure to be applied) which must be satisfied in generating a solution. In the process of trying to generate a plan, it is necessary for the problem solver to keep track of the actions tried and the effects of these actions on the system state. Figure III-5 is a restatement of Figure III-4 in which we can view the operators as impacting the data base to change the current situation (system state).

Problem Reduction

One simple form of problem solving is "divide and conquer," usually referred to as "problem reduction". Very often several subproblems (conjuncts) must be satisfied simultaneously in order to achieve a goal.

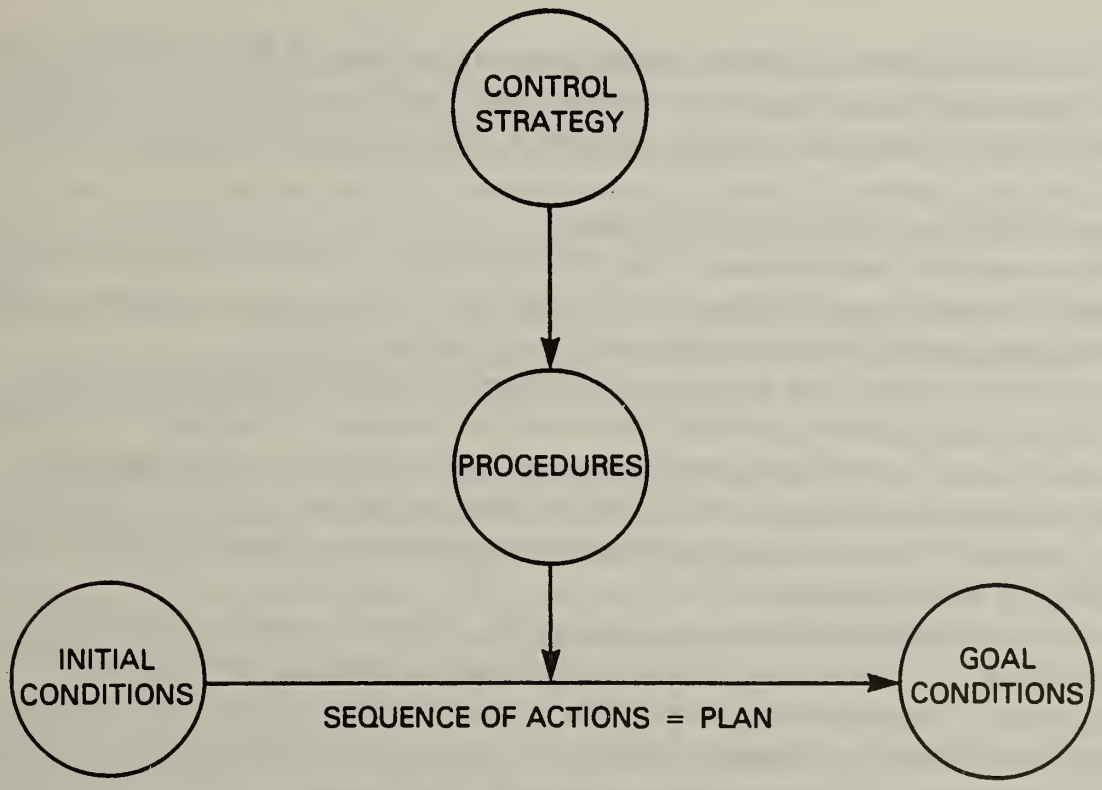


Figure III-4. Problem Solving.

*A more complete presentation is given in Chapter II of Part C.

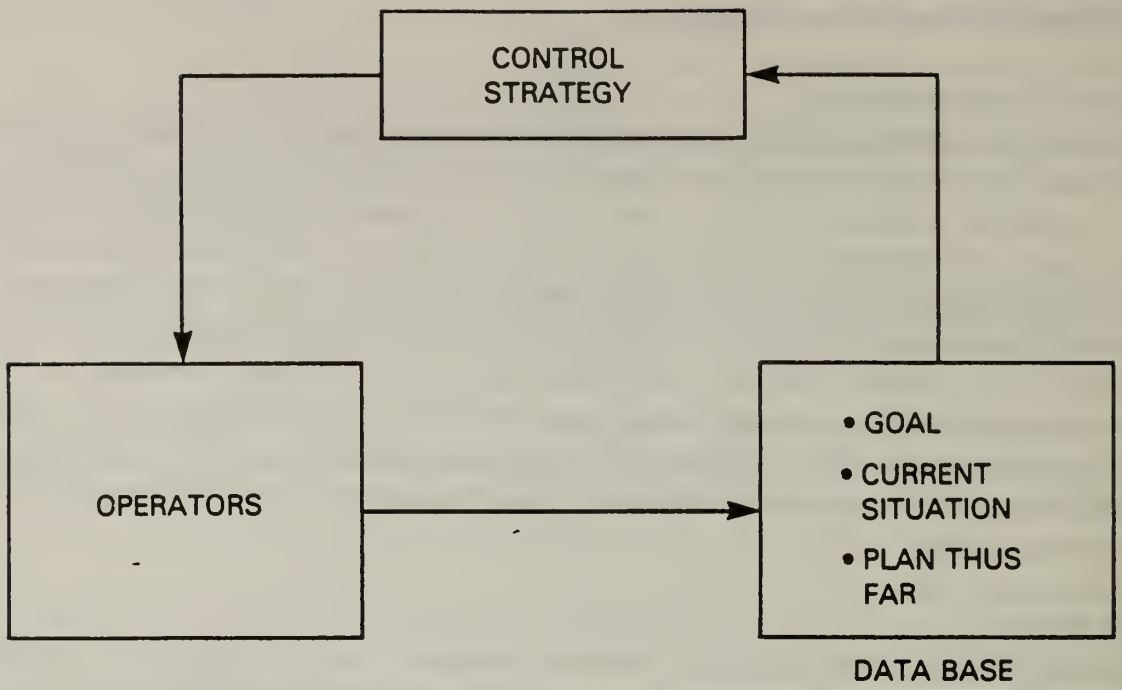


Figure III-5. Automatic Problem Solving Relationships

Problem reduction often fails without specific problem knowledge, as there is otherwise no good reason to attack one interacting conjunct before another. Lack of such knowledge may lead to an extensive search for a sequence of actions that tries to achieve subgoals in an unachievable order.

Difference Reduction ("Means-Ends" Analysis)

Difference reduction was introduced by the General Problem Solver (GPS) Program developed by Newell, Shaw and Simon, beginning in 1957. This was the first program to separate its general problem-solving method from knowledge specific to the current problem.

The means-ends analysis first determines the difference between the initial and goal states and selects the particular operator that would most reduce the difference. If this operator is applicable in the initial state, it is applied and a new intermediate state is created. The difference between this new intermediate state and the goal state is then calculated and the best operator to reduce this difference is selected. The process proceeds until a sequence of operators is determined that transforms the initial state into the goal state.

The difference reduction approach assumes that the differences between a current state and a desired state can be defined and the operators can be classified according to the kinds of differences they can reduce. If the initial and goal states differ by a small number of features, and operators are available for individually manipulating each feature, then difference reduction works. However, there is no inherent way in this approach to generate the ideas necessary to plan complex solutions to difficult problems.

More Efficient Tactics for Problem Solving

For more efficient problem solving it is necessary to devise techniques to guide the search by making better use of initial knowledge about the problem or of the information that can be discovered or learned about the problem as the problem solver proceeds through the search.

Sacerdoti (1979) indicates that information relevant to planning that can be learned during the exploration process includes:

- order relationships among actions
- hierarchical links between actions at various levels of abstraction
- the purpose of the actions in the plan
- the dependence among objects (or states) being manipulated

There are two opposing ways to improve the efficiency (solution time) of a problem solver:

- use a cheap evaluation function and explore lots of paths that might not work out, but in the process acquire information about the interrelationships of the actions and the states as an aid in efficiently guiding a subsequent search.
- use a relatively expensive evaluation function and try hard to avoid generating states not on the eventual solution path.

The following methods are attempts to achieve more efficient problem solving through employing various ratios of exploration and evaluation.

a. Hierarchical Planning and Repair

As in planning by humans, one can start by devising a general plan and refine it several times into a detailed plan. The general plan can be used as a skeleton for the more detailed plan. Using this approach, generating rather complex plans can be reduced to a hierarchy of much shorter, simpler subproblems. As the detailed plans are generated, the results should be checked to see that the intended general plan is being realized. If not, various methods for patching up the failed plan can be applied.

Another approach is to observe that some aspects of a problem are significantly more important than others. By utilizing this hierarchical ranking, a problem solver can concentrate most of its efforts on the critical decisions or more important subgoals first.

b. Problem Solving by Creating and then Debugging Almost-Right Plans

This approach deliberately oversimplifies the problem so it can be more readily solved and then corrects the solution using special debugging techniques (associated with errors due to the simplification). An everyday example is the general tactic by which people use road maps: find a simple way to get to the vicinity of your destination and then refine the plan from there.

c. Special Purpose Subplanners

This approach uses built-in subroutines to plan frequently occurring portions of a problem, such as certain moves or subgoals in robotics.

d. Constraint Satisfaction

This technique provides special purpose subplanners to help insure that the action sequences that are generated will satisfy constraints.

e. Relevant Backtracking (Dependency-Directed or Non-Chronological Backtracking)

The focus here is on sophisticated post-mortem analysis gained from several attempts that failed. The problem solver then uses this information to backtrack, not to the most recent choice point, but to the most relevant choice point.

f. Disproving

In this approach, attempts are made to prove the impossibility of the goal, both to avoid further pursuing an intractable problem, and to employ the resultant information generated to help suggest an action sequence to achieve the goal for a feasible problem.

g. Pseudo-Reduction

For the difficult case where multiple goals (conjuncts) must be satisfied simultaneously, one approach is to find a plan to achieve each conjunct independently. The resultant solutions to these simpler problems are then integrated using knowledge of how plan segments can be intertwined without destroying their important effects. By avoiding premature commitments to particular orderings of subgoals, this tactic eliminates much of the backtracking typical of problem solving systems.

h. Goal-Regression

This tactic regresses the current goal to an earlier position in the list of goals to be satisfied. This approach can be useful in cases where conjunctive subgoals must be satisfied, but where the action that satisfies one goal tends to interfere with the satisfaction of the others.

Table III-1 (derived from Sacerdoti, 1979, p. 15) indicates where the emphasis lies in the various problem-solving techniques discussed—either in the computational effort employed in evaluating the information gained thus far from the searched region, or in the effort expended in choosing the next move based only on local information.

Production Systems

Production rules (PR's), such as:

If the Shuttle Power Supply fails
  and a backup is available,
  and the cause of failure no longer exists,
Then switch to the backup.

have proved such a convenient modular way to represent knowledge, that they now form the basis of most Expert Systems.

The basic automatic problem solving relationships of Figure III-5 can be recast as a production system as shown in Figure III-6. A production system consists of a knowledge base of production rules (consisting of domain facts and heuristics), a global data base (GDB) which represents the

TABLE III-1. Primary Emphasis of Problem Solving Tactics.

Relationship	Learn and Evaluate	Choose New Move Based on Local Information
Sequencing Order	• Pseudo Reduction (Plan Generation Portion) • Relevant Backtracking • Disproving	
Hierarchy	• Plan and Repair	• Special Purpose Subplanner
Purpose of Actions	• Creating Almost-Right Plans • Pseudo Reduction (Plan Repair Portion)	• Goal Regression
Dependency Among Objects	• Relevant Backtracking	• Constraint Satisfaction

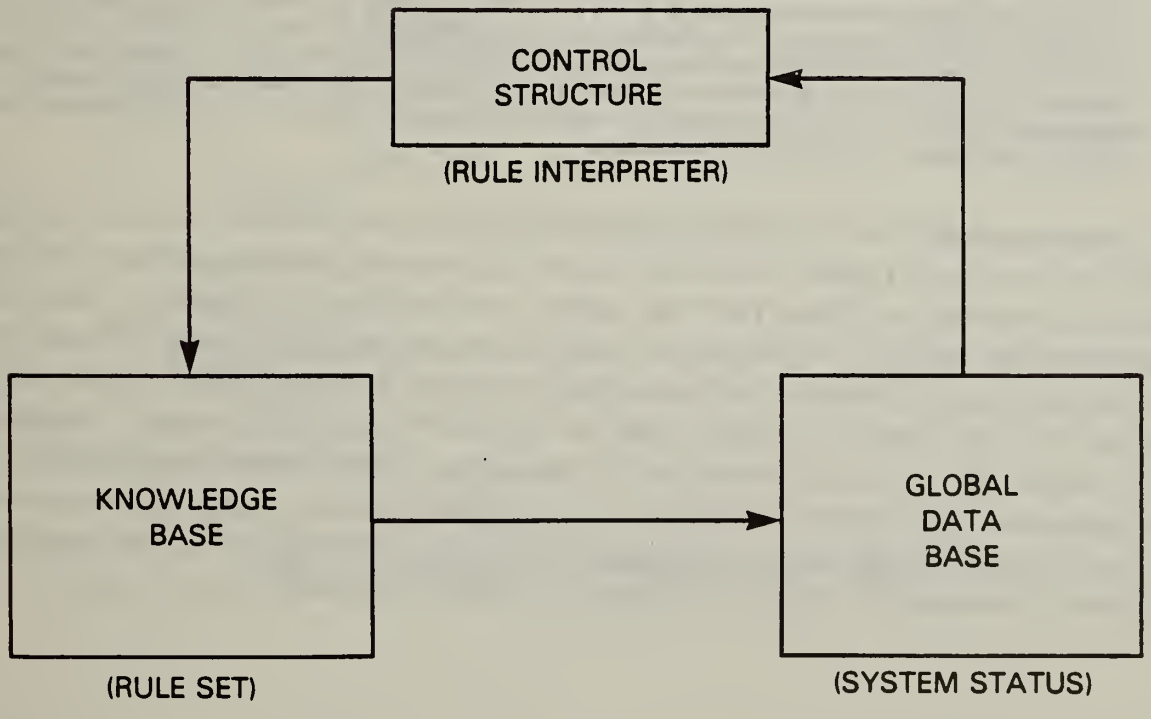


Figure III-6. A Production System.

system status, and a rule interpreter (control structure) for choosing the rules to execute. In a simple production rule system, the rules are tried in order and executed if they match a pattern in the GDB.

However, in more complex systems, such as used in Expert Systems, a very complex control structure may be used to decide which group of PR's to examine, and which to execute from the PR's in the group that match patterns in the GDB. In general, these control structures work in a repetitive cycle of the form:

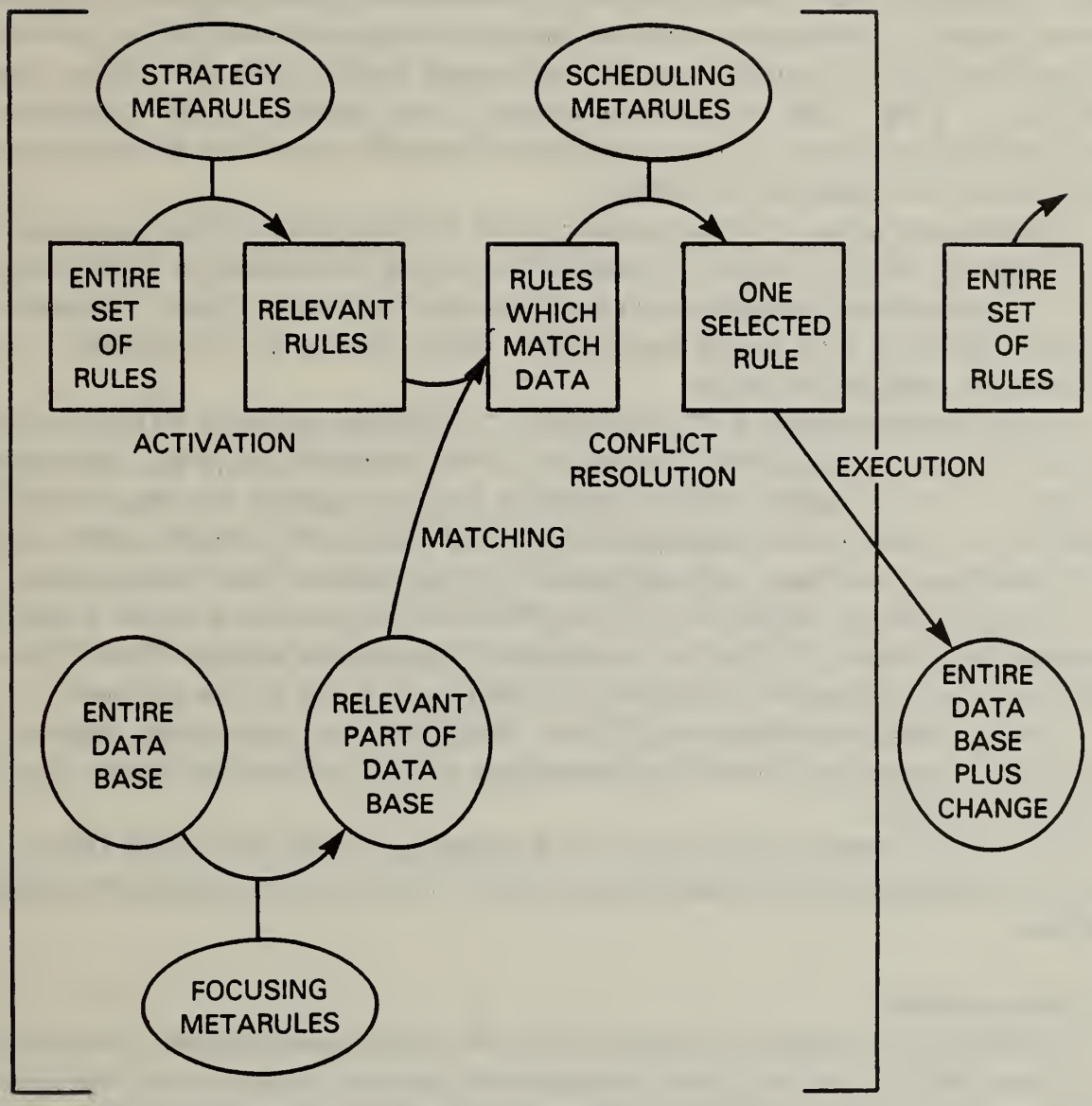
1. Find the "conflict set" (the set of competing rules which match some data in the GDB).
2. Choose a rule from among the conflict set.
3. Execute the rule, modifying the GDB.

Production rule systems can be implemented for any of the problem-solving approaches discussed earlier. Thus, we may use a "top-down" approach, employing the rules to chain backwards from the goal to search for a complete supportive or causal set of rules and data ("goal-driven," or "model-driven" control structure). Or, we can use a "bottom-up" approach employing forwarding-chaining of rules to search for the goal ("event-driven" or "data-driven" control structure).

In complex systems (employing many rules) the control structure may contain meta-rules which select the relevant rules from the entire set of rules, and also focuses attention on the relevant part of the data base. This reduces the search space to be considered. The control structure then employs further heuristics to select the most appropriate rule from the conflicting rules which match the preconditions in the global data base. Johnson (1980, p. 7) describes this approach as follows:

> . . . event-driven logic operates in the forward direction, comparing the left-hand sides of the rules in the rule-set with the data in the data base. The "best" of the matching rules found is selected and fired, causing the righthand side of that rule to make some modification in the global data base. This process is repeated until a goal rule matches the data and terminates the process . . . a goal rule is one which tests whether the problem is done.
>
> In a "pure" production rule interpreter, the generalized repeating process takes the form shown in Figure III-7. That idealization may be considered as a four-cycle logical process with activation, matching, conflict-resolution, and execution subcycles. For generality, the subcycle machinery in Figure III-7 is shown to be controlled by additional sets of higher-order rules about rules, which are called meta-rules. In practice, one often finds part or all of the meta-rule machinery of Figure III-7 replaced by simpler mechanisms. In practical programs many variations of this basic scheme exist because of efficiency considerations, the characteristics of the particular applications, and programmer preferences.

A simple example of an event-driven production system can be visualized for a Shuttle Flight in which the power supply status is observed to be out of limits in the Global Data Base. The strategy meta-rules indicated in Figure III-7 then select, from the tens of thousands of rules in the Knowledge Base having to do with Shuttle Flight Operations, those rules having to do with power and the use of power. Similarly, the focusing meta-rules select from the GDB the relevant part having to do with the status of the power supply, and the Shuttle's and the experiments' use of power. The relevant rules are then compared with the relevant part of the GDB to determine which rules are appropriate for the current system status. The scheduling metarules (using priorities) then select the most appropriate rule (such as switching in the backup, or turning off the less important experiments). Executing the selected rule changes the system status, and the cycle repeats.



SOURCE: JOHNSON, 1980.

Figure III-7. Idealized Event-Driven Control Scheme.

Programming Needs of AI

AI research has been an experimental science trying to develop computer programs that exhibit intelligent behavior. This has proved to be a difficult endeavor requiring the best programming tools. AI programs tend to develop iteratively and incrementally. As the programs are thus evolutionary, creating AI programs requires an interactive environment with built-in aids such as dynamic allocation of computer memory as the program evolves, rather than advance memory allocation as in most other programming domains. More importantly, the unpredictable intermediate forms of the data (as the program evolves) also influence the form of the programming languages and the management of memory.

Another unusual aspect of AI programming is that AI researchers found that expressing functions recursively (defined in terms of themselves) was a great simplification in writing programs. Thus, AI programming languages tend to support recursive processing. Finally, AI programs are primarily concerned with symbol manipulation, rather than numeric computation. All AI languages thus support this feature.

Barr and Feigenbaum (1982, p. 32) observe that, "AI programs are among the largest and most complex computer programs ever developed and present formidable design and implementation problems . . . AI researchers in their capacity as language designers and programmers have pioneered an interactive mode of programming in environments with extensive support: editors, trace and debugging packages, and other aids for the construction of large complex systems.

Two basic general AI languages—LISP and PROLOG—have evolved in answer to these programming requirements. LISP has been the primary AI programming language. PROLOG, a logic-based language, has appeared more recently and has gained favor in Europe and Japan.

Various derivatives and dialects of LISP exist. Special high level programming languages, for such purposes as assisting in knowledge representation and constructing expert systems, have been built on top of LISP.

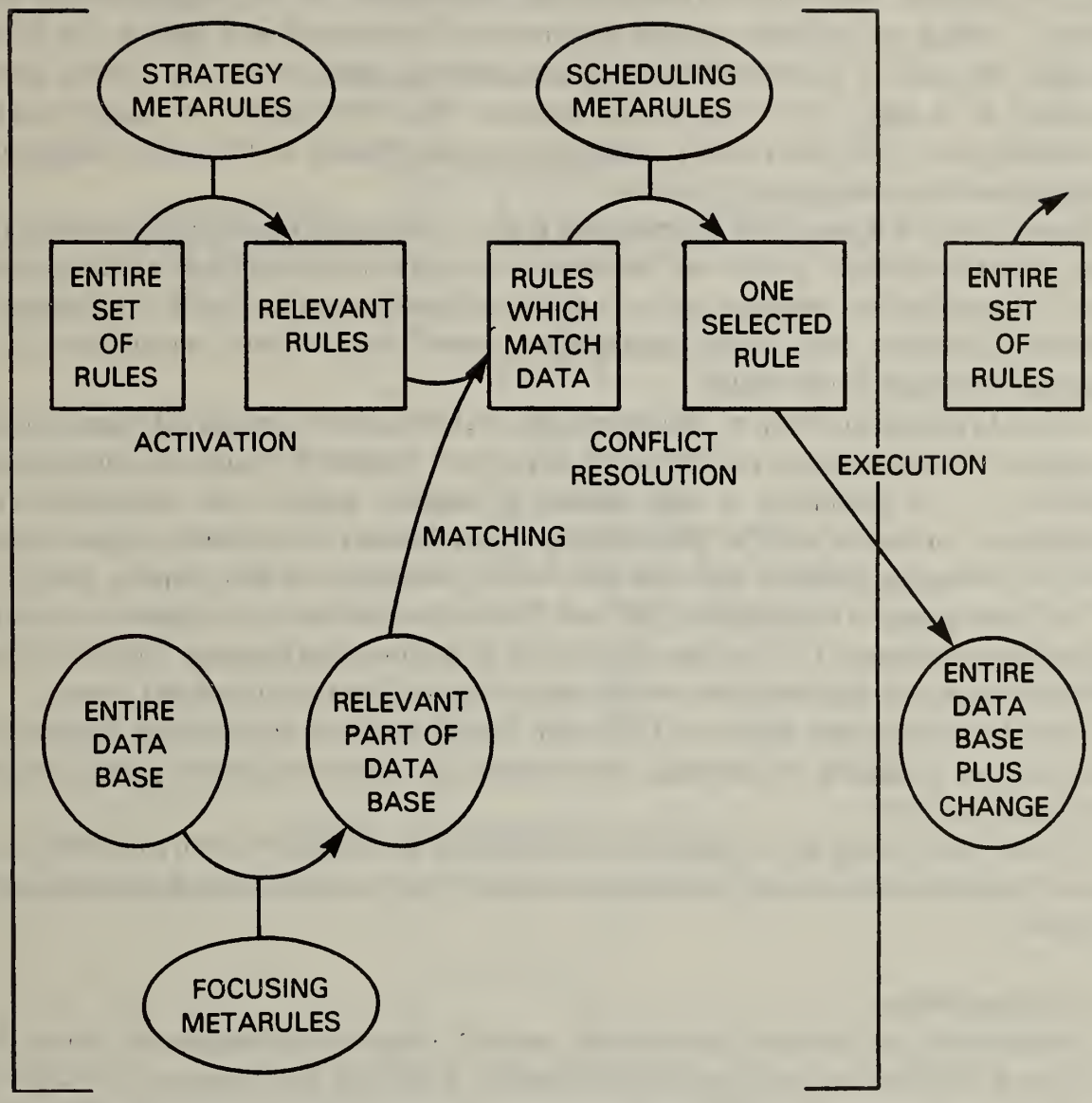
In recent years, nearly all AI programs were developed on the DEC PDP-10 and PDP-11 computers. AI programming is now transitioning to the DEC VAX computers and the new personal AI machines.

List Representations

List processing was originally introduced in their IPL programming language by Newell, Shaw and Simon (1957) to deal with symbol manipulation. Lists form associations of symbols which allow computer programs to build data structures of unpredictable shape and size. To handle such unpredictably shaped data structures IPL used primitive data elements (called cells).

The same idea is used in LISP in the form of CONS cells. Each CONS cell is an address (a computer word) that contains a pair of pointers to other locations in computer memory.* The left portion of the cell points to the first element (the "CAR") of the list. The right portion points to

*One can thus view the basic data object in LISP to be pointers, with lists as one interpretation placed upon the resultant pair structure.



SOURCE: JOHNSON, 1980.

Figure III-7. Idealized Event-Driven Control Scheme.

Programming Needs of AI

AI research has been an experimental science trying to develop computer programs that exhibit intelligent behavior. This has proved to be a difficult endeavor requiring the best programming tools. AI programs tend to develop iteratively and incrementally. As the programs are thus evolutionary, creating AI programs requires an interactive environment with built-in aids such as dynamic allocation of computer memory as the program evolves, rather than advance memory allocation as in most other programming domains. More importantly, the unpredictable intermediate forms of the data (as the program evolves) also influence the form of the programming languages and the management of memory.

Another unusual aspect of AI programming is that AI researchers found that expressing functions recursively (defined in terms of themselves) was a great simplification in writing programs. Thus, AI programming languages tend to support recursive processing. Finally, AI programs are primarily concerned with symbol manipulation, rather than numeric computation. All AI languages thus support this feature.

Barr and Feigenbaum (1982, p. 32) observe that, "AI programs are among the largest and most complex computer programs ever developed and present formidable design and implementation problems . . . AI researchers in their capacity as language designers and programmers have pioneered an interactive mode of programming in environments with extensive support: editors, trace and debugging packages, and other aids for the construction of large complex systems.

Two basic general AI languages—LISP and PROLOG—have evolved in answer to these programming requirements. LISP has been the primary AI programming language. PROLOG, a logic-based language, has appeared more recently and has gained favor in Europe and Japan.

Various derivatives and dialects of LISP exist. Special high level programming languages, for such purposes as assisting in knowledge representation and constructing expert systems, have been built on top of LISP.

In recent years, nearly all AI programs were developed on the DEC PDP-10 and PDP-11 computers. AI programming is now transitioning to the DEC VAX computers and the new personal AI machines.

List Representations

List processing was originally introduced in their IPL programming language by Newell, Shaw and Simon (1957) to deal with symbol manipulation. Lists form associations of symbols which allow computer programs to build data structures of unpredictable shape and size. To handle such unpredictably shaped data structures IPL used primitive data elements (called cells).

The same idea is used in LISP in the form of CONS cells. Each CONS cell is an address (a computer word) that contains a pair of pointers to other locations in computer memory.* The left portion of the cell points to the first element (the "CAR") of the list. The right portion points to

*One can thus view the basic data object in LISP to be pointers, with lists as one interpretation placed upon the resultant pair structure.

another CONS cell representing the remainder (the "CDR") of the list. Thus, as indicated in Figure III-8, representing a sequence of words or symbols in memory can be visualized as a binary tree structure using these memory cells.

The problem of unpredictable size of data structures was solved by having a free list of memory cells that could be dynamically allocated as required.

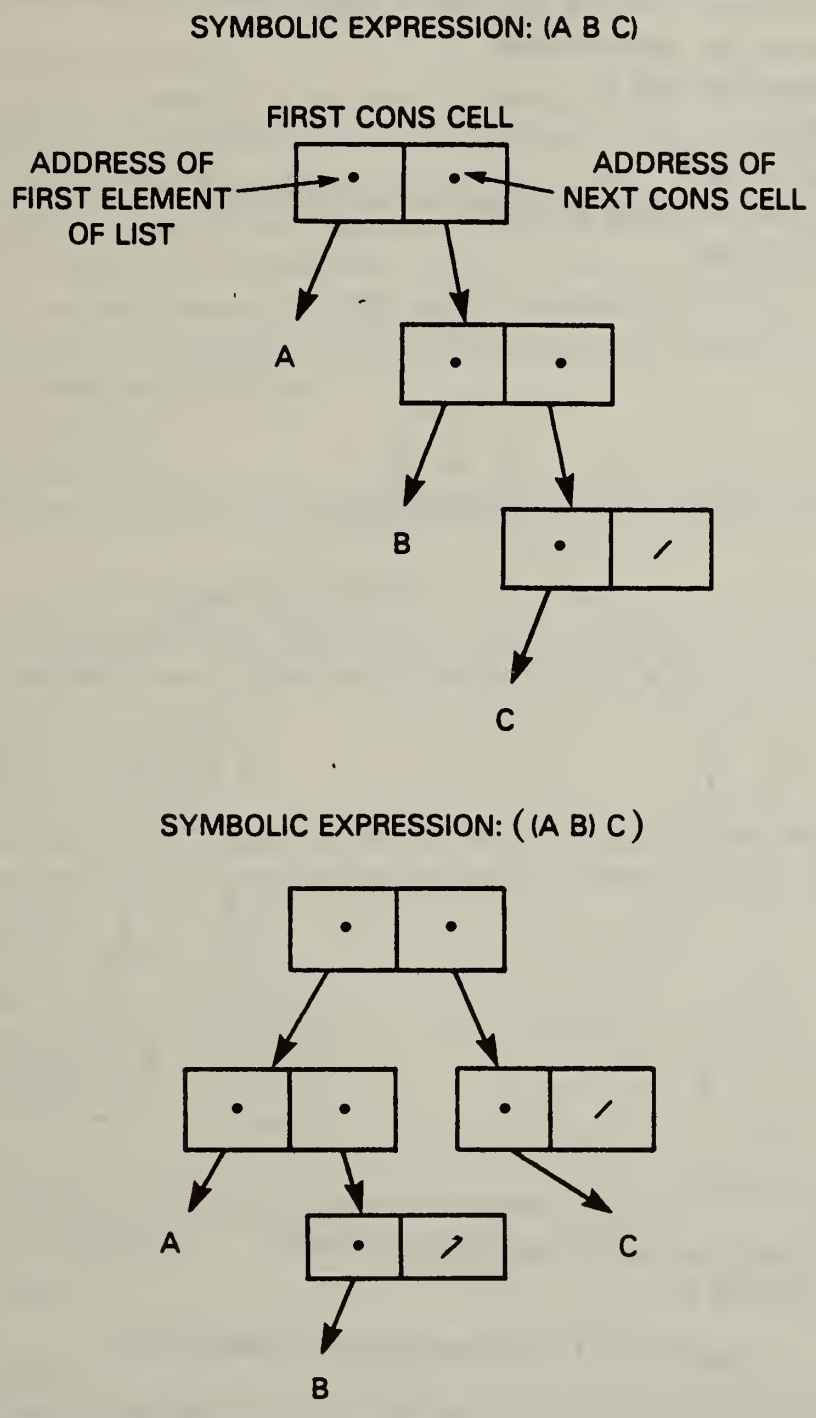


Figure III-8. Representation of List Structures in Memory.

A list is a sequence of zero or more elements enclosed in parentheses, where each element is either an atom (an indivisible element) or a list. Lists can be used to represent virtually any type of data. Lists are therefore useful for representations in such AI areas as language understanding, computer vision, and problem solving and planning.

Tree structures (used to represent search spaces) are ubiquitous in AI. A list representation for a tree structure is shown in Figure III-9. It will be observed that the resultant representation is a list (as indicated by parentheses) consisting of elements, some of which are also lists. These nested structures are common in list representations.

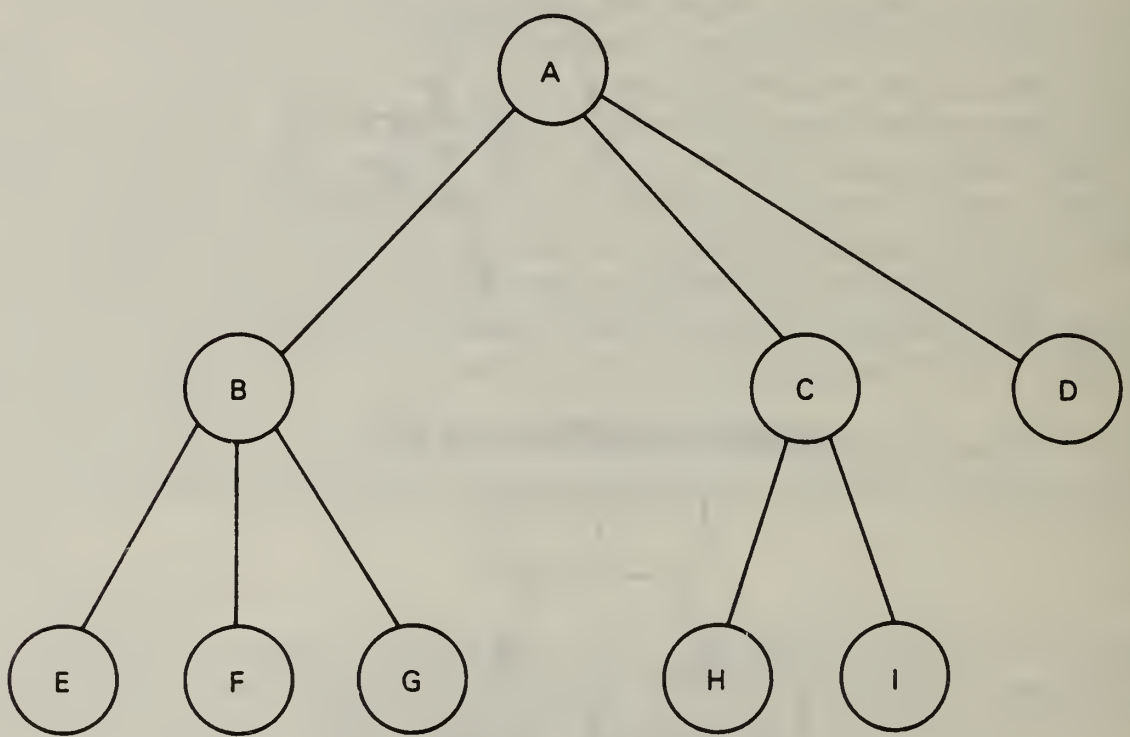
Predicate logic expressions such as

IN(x,A) OR IN(x,B)

meaning x is in A or in B

can be conveniently expressed, using prefix notation, in list form as

(OR (IN x A) (IN x B))



(A (B E F G) (C H I) D)

WHERE (B E F G), (C H I) AND D ARE THE SUBTREES
OF THE ROOT NODE A.

Figure III-9. List Representation of a Search Tree.

LISP

a. Background

Around 1960, John McCarthy at M.I.T. developed LISP as a practical list processing language with recursive-function capability for describing processes and problems. Since then, LISP has been the primary AI programming language—the one most used in AI research.

b. Basic Elements of LISP

All LISP programs and data are in the form of symbolic expressions (S-expressions) which are stored as list structures. LISP deals with two kinds of objects: atoms and lists. Atoms are symbols (constants or variables) used as identifiers to name objects which may be numeric (numbers) or non-numeric (people, things, robots, ideas, etc.). A list is a sequence of zero or more elements enclosed in parentheses, where each element is either an atom or a list.

Graham (1979, p. 226) observes, "A LISP system is a function-evaluating machine. The user types in a function and its arguments. LISP types back the result of applying the function and its arguments." For example for addition:

User input: (PLUS 6 2)

LISP response: 8

To manipulate lists, LISP has three basic functions (related to the memory cell structure storage for lists):

CONS, to join a new first member to a list.

CAR, to retrieve the first member of a list.

CDR, (pronounced coud-er) to retrieve the list consisting of all but the first member of a list. Thus:

User: (CONS 'Z '(C D E))

LISP (Z C D E)

where the quote symbol, ', is used to indicate that the expression following is not to be evaluated. Normally, LISP evaluates all expressions (starting with the innermost parentheses) before carrying out other operations.

User: (CAR '(John Mary X Y))

LISP: John

User: (CDR '(John Mary X Y))

LISP: (Mary X Y)

c. Variables

In LISP, the SET function assigns a value to a variable. Thus:

User: (SET 'Z George)

LISP: George

User: Z

LISP: George

Atoms are used for variables in LISP. When quoted, an atom stands for itself, otherwise LISP automatically substitutes its value during processing.

d. Defining New Functions

Programming in LISP involves defining new functions. Thus, we could define SECOND (the second atom of a list) by:

User: (DEFUN SECOND (Y) (CAR(CDR Y)))

where Y is a dummy variable.

LISP: SECOND

User: (SECOND '(JOHN FRANK MARY JANE))

LISP: FRANK

e. Predicates

A predicate is a function which returns either NIL (false) or T (true). As a programming convenience, any none-NIL value is also considered to be true. (NIL is actually the name of the empty list.)

Thus, the predicate GREATERP returns T if the items in the series are in descending order.

User: (GREATERP 6 5 2)

LISP: T

f. Conditional Branching

It is often necessary in AI to use conditional branching. For example, if so and so is true, then do X, if not, if thus and so is true, then do Y, if not do Z.

The COND function in LISP has this role. Its form is:

(COND (condition 1 expression 1)
(condition 2 expression 2)
•
•
•
•
•
(condition m expression m))

where each condition is an expression that will evaluate to NIL or something else. The COND function evaluates the conditions in order until one of them evaluates to other than NIL. It then returns the value of the corresponding expression.

g. Recursive Functions

It is often much easier to define a function recursively—in terms of itself—than to define it as an explicit series of steps. This recursive feature is an important characteristic of LISP. A simple illustration is the factorial example (Barr and Feigenbaum, 1982, P. 6):

$$N! = \begin{cases} 1 & \text{if } N = 1 \\ N \times (N-1)! & \text{if } N > 1 \end{cases}$$

This factorical function (FACTORIAL) can be written as:

```
(DEFUN FACTORIAL (N)
  (COND (EQUAL N 1) 1)
    (T (TIMES N (FACTORIAL (DIFFERENCE N 1)))))
```

h. Review of Program Features of LISP

It should be noted that LISP programs and data are both in the same form - lists. Thus, AI programs can manipulate other AI programs. This allows programs to create or modify other programs, an important feature in intelligent applications. It also allows programming aids for debugging and editing to be written in LISP, providing great interactive flexibility for the LISP programmer, who can thus tailor things to suit his needs.

Other notable aspects of LISP are (Barr and Feigenbaum, 1982, p. 17)

—LISP is an interactive interpreted language and therefore relatively slow. (However, it can be compiled, often resulting in an order of magnitude improvement.)

—Memory allocation is automatic.

—LISP expressions are very simple and regular. All expressions are made up of atoms and compositions of atoms.

—Control is normally applicative—the flow of control being guided by the application of functions to arguments, in contrast to the sequential control structure of most programming languages.

—Dynamic Scoping of Variables—usually a non-local variable will have its value locally assigned by the function evaluating it, unless it was assigned a value by the function calling the evaluating function.

—For real-time operation, LISP requires a sophisticated garbage collection system to recycle memory cells no longer being used.

—LISP is a huge package and until the advent of the special personal LISP machines, the full capabilities of LISP could only be implemented on large computers.

—The use of nested parentheses in LISP can be confusing, but the confusion can be reduced somewhat by indenting expressions according to their level of nesting.

i. LISP Today

There are two major LISP dialects today:

MACLISP developed at M.I.T.

INTERLISP developed at BBN and XEROX-PARC.

Both offer very similar programming environments with editing and debugging facilities. Both offer many LISP functions and optional features. The emphasis in INTERLISP has been to provide the best possible programming environments, even at the expense of speed and memory space. MACLISP has had more emphasis on efficiency, conservation of address space and flexibility for building tools and embedding languages.

INTERLISP has been the much better-supported version, with complete documentation and many users. It runs on DEC and XEROX operating systems.

Out of the need to standardize the various MACLISP dialects has evolved "Common LISP" and "LISP Machine LISP" for personal AI computers. This new dialect—Common LISP—appears destined to be used on most of the new personal AI machines and operating systems. Common LISP is intended to be efficient and portable, with stability a major goal.

Because of the rapid development of LISP features by the user community, other more local LISP versions (such as FRANZLISP at U. of CA, Berkeley) exist at several university AI labs.

A good text on LISP programming is *LISP* by Winston and Horn (1981).

PROLOG (PROgramming in LOGic)

a. History

PROLOG is a logic-oriented language developed in 1973 at the University of Marseille AI Laboratory by A. Colmerauer and P. Roussel. Additional work on PROLOG has been done at the University of Edinburgh in Great Britain. Development of PROLOG in France has continued to the present, achieving a documented system that can be run on nearly all computers (Colmerauer et al., 1981).

b. Nature of PROLOG

PROLOG is a theorem-proving system. Thus, programs in PROLOG consist of "axioms" in First-Order Predicate Logic together with a goal (a theorem to be proved). The axioms are restricted to implications, the left- and right-hand sides of which are written in "Horn-clause" form.

A Horn clause consists of a set of statements joined by logical AND's. Thus, the form of a typical PROLOG axiom is:

$$A \cap B \cap C \cap X \longrightarrow Y \cap Z$$

That is, A AND B AND C AND X together IMPLY Y AND Z.

when read declaratively. It can also be read procedurally as:

To prove Y AND Z, try to prove A AND B AND C AND X.

Looked at this way, a PROLOG program consists of a group of procedures, where the left side of a procedure is a pattern to be instantiated* to achieve the goals on the right side of the procedure.

$$\text{PROCEDURE:} \quad \text{PATTERN} \longrightarrow \text{GOALS}$$

(Note the similarities of these modular rules to the IF, THEN production rules used in constructing Expert Systems. It is this modularity which promotes clear, accurate, rapid programming—that is one of the reasons for PROLOG's popularity.)

*An instance is found that satisfies it.

EXAMPLE: Find the geopolitical entities in Europe:
DATA: written as a relational data base in Horn clause form:

PARTOF (London, England)
$\cap$ PARTOF (England, Europe)
$\cap$ PARTOF (Boston, U.S.)
$\cap$ PARTOF (Tokyo, Japan)

PROCEDURES:

(1) $\text{PARTOF}(X,Y) \longrightarrow \text{IN}(X,Y)$

That is, to prove X is in Y, try to prove X is part of Y.

(2) $\text{PARTOF}(X,Y) \cap \text{IN}(Y,Z) \longrightarrow \text{IN}(X,Z)$

That is, to prove X is in Z, try to prove X is part of Y and Y is in Z.

GOAL (Theorem to be Proved):

IN(X, Europe)

That is, What X's are in Europe?

By matching the goal to the right-hand side of the first procedure we instantiate the procedure by letting

Europe = Y

Then matching the data to this procedure we find

X = England

Matching the goal to the right-hand side of the second procedure, we instantiate it by letting

Europe = Z

Now, matching the data to the two procedures, we instantiate them by letting

Y = England
X = London

Thus we have two instances:

X = England
X = London

that satisfy the goal.

As indicated by the example, PROLOG solves a problem by pattern-matching, which can be viewed as unification (the carrying out of instantiations) in the sense of First Order Predicate Logic. (PROLOG incorporates a very powerful pattern-matching mechanism.) If the pattern-matching fails as PROLOG searches through its procedures, then it automatically backtracks to its previous choice point, resulting in a depth-first type of search.

The solution process starts with the system searching for the first clause whose right side matches (unifies with) the goal. Thus, the search process can be guided by the programmer by choosing the order for the procedures, the data, and the goals in the clauses.

PROLOG can be considered as an extension of pure LISP coupled with a relational data base query language (as exemplified by the Horn clause form for expressing the basic data) which utilizes virtual relations (implicit relations defined by rules). Like LISP, PROLOG is interactive and uses dynamic allocation of memory.

c. PROLOG Today

PROLOG is a much smaller program than LISP and has now been implemented on a variety of computers (including microcomputers). A documented highly portable version of PROLOG has been written in France (Colmerauer, et al, 1981.) The execution of PROLOG is surprisingly efficient and in its compiled version it is claimed to be faster than compiled LISP. PROLOG has proved very popular in Europe and is now targeted as the language for Japan's Fifth Generation Computer project. PROLOG's design (and its powerful pattern matcher) is well suited to parallel search and therefore an excellent candidate for such powerful future computers incorporating parallel processing. Substantial interest in PROLOG is now arising in the U.S., with some of PROLOG's features being implemented in LISP.

PROLOG's principal drawback appears to be its depth-first search approach which could be a concern in certain complex problems that tend toward combinatorial explosions in the size of the search space.

PROLOG was originally developed for natural language understanding applications, but has since found use in virtually all AI application areas.

Other AI Languages

A number of AI languages have been developed as extensions of, improvements upon (e.g., special features for knowledge organization and search), or alternatives to, LISP. These include (see Barr and Feigenbaum, 1982):

- *System Programming Languages* (LISP-level)
  - —SAIL (Stanford AI Language) (1969)
  - —QA4 and QLISP (at SRI 1968, 1972)
  - —POP-2 (at U. of Edinburgh 1967)
- *Deduction/Theorem-Proving Languages*
  - —PLANNER and MICROPLANNER (M.I.T., 1971)
  - —CONNIVER (M.I.T., 1972)
  - —POPLER (U. of Edinburgh, 1972)
  - —PROLOG (U. of Marseille, 1973)
  - —AMORD (M.I.T., 1977)

LISP and POP-2 are designed to simplify the building of new languages within or on top of them—QLISP being embedded in INTERLISP, POPLER embedded in POP-2. POP-2 remains popular in England, but has not caught on elsewhere.

Most of the above AI languages are no longer supported and have fallen into disuse. However, they were experiments that helped pave the way for the modern AI languages now in use, such as the LISP dialects, PROLOG, and POP-2. For instance, PROLOG's style of programming is similar to that demonstrated in QA3 and PLANNER.

Other special languages have been built for Knowledge Representation, Knowledge Base Management, writing rule-based systems (such as Expert Systems), and for special application areas. These languages are treated in the appropriate sections of this report.

AI Computational Facilities

a. Requirements

Good AI people are still scarce, expensive and dedicated. It therefore behooves an organization supporting an AI group to provide the best facilities available (commensurate with cost) so as to maximize the productivity of their AI people.

Fahlman and Steele (1982) state that desirable features of an AI Programming Environment are:

- Powerful, well-maintained, standardized AI languages.
- Extensive libraries of code and domain knowledge (A facility should support the exchange of code with other AI research facilities).
- Excellent graphic displays: high resolution, color, multiple windows, quick update, and software to use all of these easily.
- Good input devices.
- Flexible, standardized inter-process communication.
- Graceful, uniform user-interface software.
- A good editor that can deal with a program based on the individual program structure.

They suggest that:

- Sticking with the hardware and systems that major AI centers are using is important, so that the time can be spent getting work accomplished, not reinventing the wheel.
- $50K-$100K per researcher for computing facilities is appropriate.
- Your AI product can be developed in the best available environment. Once developed, it can be ported to other languages and machines, as appropriate.
- Isolated machines are nearly useless. Good network interfaces, internal and external are critical.*
- AI people spend roughly 80% of their time editing, reading and communicating. Thus, facilities for this must be excellent.

b. AI Machines

The computers used for AI research for the past several years have been primarily the DEC system-10 and DEC system-20 family of time-shared machines. These are now being superseded by the more economical DEC VAX time-shared computers and the newer personal AI machines. However, Brown (1981) sees this as a mixed blessing, as the newer machines are still deficient in software, as compared to the older DEC 10's and 20's with their rich software libraries.

The newer machines tend to have 32-bit words, sorely needed for address space, as most AI programs are huge.

Fahlman and Steele (1982) see the DEC VAX, with a UNIX operating system, as the best time-sharing machine today for AI purposes. Several LISP dialects are available. The VAX is the current choice of many universities.

M.I.T. designed a personal machine specially microcoded for LISP. This M.I.T. LISP Machine has been licensed to LMI and Symbolics. Table III-2 lists some of the current Personal AI

*Brown (1981) sees an ARPANET link as essential for a NASA AI lab. "The ARPANET links most U.S. AI research centers and provides an electronic bulletin board, electronic mail, file transfer, and access to remote data bases, software tools and computing power."

TABLE III-2. Some Personal AI Computers Now Available.

Machine/Company	Approx. Price	Characteristics
3600 Symbolics, Inc. Cambridge, MA	$75K	A new complete redesign of MIT LISP Machine. Very fast and flexible. Extensive software.
Lambda LISP Machines, Inc. Los Angeles, CA	$73K	A recent redesign of MIT LISP Machine. Software from MIT.
PERQ1a 3 Rivers Computer Corp. Pittsburgh, PA	$30K	SPICE Software from CMU, including Common LISP. Much less powerful than LISP machines.
XEROX 1100 Series XEROX Electro-Optical Systems Pasadena, CA	$25K*/up	A mature INTERLISP system available in several versions having different memory capacities.

*Mainframe computer support desirable.

Machines. Several other personal machines of lesser capacity are also being offered for AI applications.

These new AI personal machines represent unusually powerful interactive exploratory programming environments in which system design and program develop together (Sheil, 1983). This is in sharp contrast to the more traditional structured programming approach in which software program specifications are first written, with the software development following in rigid adherence to the specifications.

To further enhance the exploratory programming approach, the user-friendly object-oriented programming languages have been devised. An object (such as an airplane, or a window on a computer screen) can be encoded as a package of information with attached descriptions of procedures for manipulation of that information. Objects communicate by sending and receiving messages that activate their procedures. A class is a description of one or more similar objects. An object is an instance of a class and inherits the characteristics of its class. The programmer developing a new system creates the classes that describe the objects that make up the system and implements the system by describing messages to be sent. Use of object-oriented programming reduces the complexity of large systems. The notion of class provides a uniform framework for defining system objects and encourages a modular, hierarchical (top-down) program structure (Robinson, 1981). "Smalltalk" is an object-oriented language available on the Xerox machines. "Flavors" is available on the M.I.T.-based LISP machines. LOOPS being developed at Xerox PARC is a further extension of the Smalltalk system. ROSS at RAND Corp. is an object-oriented programming language for use in symbolic simulation of activities such as air combat.

The new AI personal machines tend to come with interactive facilities for program development, editing, debugging, etc. For key portions of AI programs, microcode allows key inner loops to be run very fast. This is especially important in graphics and vision programs. The efficiency of computers microcoded for AI applications and suppporting large memories make these personal computers especially attractive.

ZETALISP derived from MACLISP is an integrated software environment for program development and program execution as the Symbolics 3600. ZETALISP has available nearly 10,000 compiled functions making it an exceptionally powerful and functional form of the LISP programming language. Similar capabilities are available on the LMI Lambda machines.

Interlisp-D, used on the Xerox 1100 machines, provides a comprehensive programming environment particularly suited for the development and delivery of expert and other knowledge-based systems.

c. Future

It is expected that the price of good AI personal computers that run LISP will rapidly drop below $50K as competition heats up and demand escalates. It is thus anticipated that one personal AI machine per AI person will be the standard of the future.

Parallel architectures are now being considered for future AI machines. This is especially attractive for PROLOG because its structure facilitates parallel search. Japan intends to build sequential PROLOG personal computers by 1985, featuring 10K logical inferences per second. In the 1990 time frame, Japan's Fifth Generation Computer project is projected to yield an enormously powerful AI parallel processing machine running PROLOG at one billion logical inferences per second (about 10,000 times more powerful than the DEC-KL-10 on which the AI community grew up).

Summary and Forecast

It now appears that LISP dialects designed specifically for personal computers will become commonplace. It is also expected that software portability will improve substantially. PROLOG and its derivatives, now prevalent throughout Europe, will become incorporated with LISP in the U.S.

Powerful Personal AI Computers under $50K will rapidly appear and become the AI standard for the next several years. In the longer term, powerful parallel computers, such as the Japanese Fifth Generation Computers, will probably become the standard as the number of AI practitioners expands and more difficult problems are addressed.

The rapidly increasing capability and ease of development of VLSI chips, promises to move AI computing power for developed applications out of the laboratory and into the field and products as needed.

An emerging trend is the increased use of object-oriented programming to ease the creation of large exploratory programs. The use of objects is also a good way to program dynamic symbolic simulations, which will become more important as the quest for utilizing deeper knowledge accelerates and the demand for increased reliability of knowledged-based systems is pursued. Object-oriented programming also holds promise for distributed processing, as each object could be implemented on a separate processor in a linked network of processors.

Finally, it is anticipated that the AI exploratory software development approach will slowly infuse conventional software practices.

CHAPTER III REFERENCES

- Barr, A. and Feigenbaum, E.A. (Eds.), *The Handbook of Artificial Intelligence,* Vol. I, Los Altos, CA, W. Kaufmann, 1981.
- Barr, A., and Feigenbaum, E.A. (Eds.), *The Handbook of Artificial Intelligence,* Vol. II, Los Altos, CA, W. Kaufmann, 1982.
- Boyer, R.S., and Moore, J.S., *A Computational Logic,* New York, Academic Press, 1979.
- Brown, D.R., "Recommendations for an AI Research Facility at NASA/GSFC," SRI Project 2203, SRI Inter., Menlo Park, CA, October 1981.
- Cohen, P.R. and Feigenbaum, E.A. (Eds.), *The Handbook of Artificial Intelligence,* Vol. III, Los Altos, CA, W. Kaufmann, 1982.
- Colmerauer, A., Kanoui, H., Van Canegham, M., "Last Steps Toward an Ultimate PROLOG," *IJCAI-81,* Vancouver, Canada, Aug. 1981, pp. 947-948.
- Fahlman, S.E., and Steele, G.L., "Tutorial on AI Programming Technology: Language and Machines," AAAI-82, Pittsburgh, PA, August 16, 1982.
- Graham, N., *Artificial Intelligence,* Blue Ridge Summit, PA, Tab Books, 1979.
- Johnson, C.K., "Programming Methodology of Artificial Intelligence," *Computing in Crystallography,* Diamond, R., and Kenkatsan, S.R. (Eds.), Bangladore, India, Indian Academy of Sciences, 1980, pp. 28.01-28.16.
- Minsky, M., "A Framework for the Representing Knowledge," In P. Winston (Ed.), *The Psychology of Computer Vision.* New York, New York, McGraw-Hill, 1975, pp. 211-277.
- Newell, A., Shaw, J.C. and Simon, H.A., "Programming the Logic Theory Machine," *Proc of the Western Joint Computer Conf.,* 1957, pp. 230-240.
- Newell, A., Shaw, J.C. and Simon, H.A., "A Variety of Intelligent Learning in a General Problem Solver," In M.C. Yovits and S. Cameron (Eds.), *Self Organizing Systems,* New York, Pergamon Press, 1960, pp. 153-189.
- Nilsson, N.J. *Principles of Artificial Intelligence,* Palo Alto, CA, Tioga Pr., 1980.
- Robinson, D., "Object-Oriented Software Systems," *BYTE,* Aug. 1981, pp. 74-86.
- Robinson, J.A., "A Machine-Oriented Logic Based on the Resolution Principle," *J. ACM,* Vol. 12, 1965, pp. 33-41.
- Sacerdoti, E.D., "Problem Solving Tactics," TN189, SRI Inter., Menlo Park, CA, July 1979.
- Schank, R.C. and Riesbeck, *Inside Computer Understanding,* Hillsdale, N.J.: Lawrence Erlbaum, 1981.
- Sheil, B., "Power Tools for Programmers," *Datamation,* Feb. 1983, pp. 131-144.
- Winston, P.H. and Horn, B.K.P., *LISP,* Reading, Mass., Addison-Wesley, 1981.

IV. APPLICATIONS

The potential range of AI applications is so vast that it covers virtually the entire breadth of human intelligent activity. Detailed listings of focused applications are given in Part B of this volume, in each of the sections on applications: expert systems, computer vision, natural language processing, and problem solving and planning. This section just summarizes some of the key applications. Generic applications are listed in Table IV-1. Examples of specific applications of AI are listed in Table IV-2. Potential functional applications for NASA are indicated in Table IV-3. The opportunities this opens up for NASA are listed in Table IV-4. Similar opportunities are available in many other public and private domains.

TABLE IV-1. Generic Applications of AI.

Knowledge Management
- Intelligent data base access
- Knowledge acquisition
- Text understanding
- Text generation
- Machine translation
- Explanation
- Logical operations on data bases

Human Interaction
- Speech understanding
- Speech generation

Learning and Teaching
- Intelligent computer-aided instruction
- Learning from experience
- Concept generation

Fault Diagnosis and Repair
- Humans
- Machines
- Systems

Computation
- Symbolic mathematics
- "Fuzzy" operations
- Automatic programming

Communication
- Public access to large data bases via telephone and speech understanding
- Natural language interfaces to computer programs

Operation of Machines and Complex Systems

Autonomous Intelligent Systems

Management
- Planning
- Scheduling
- Monitoring

TABLE IV-1. (cont'd)

Sensor Interpretation and Integration
- Developing meaning from sensor data
- Sensor fusion (integrating multiple sensor inputs to develop high level interpretations)

Design
- Systems
- Equipment
- Intelligent Design Aids
- Inventing

Visual Perception and Guidance
- Inspection
- Identification
- Verification
- Guidance
- Screening
- Monitoring

Intelligent Assistants
- Medical Diagnosis, Maintenance Aids and Other Interactive Expert Systems
- Expert System Building Tools

TABLE IV-2. Examples of Domain-Specific Applications of AI.

Medical
- Diagnosis and Treatment
- Patient Monitoring
- Prosthetics
  —Artificial Sight and Hearing
  —Reading Machines for the Blind
- Medical Knowledge Automation

Science and Engineering
- Discovering
  —physical and mathematical laws
  —determination of regularities and aspects of interest
- Chemical and Biological Synthesis Planning
- Test Management
- Data Interpretation
- Intelligent Design Aids

Industrial
- Factory Management
- Production Planning and Scheduling
- Intelligent Robots
- Process Planning
- Intelligent Machines
- Computer-Aided Inspection

TABLE IV-2. (cont'd)

Military
- Expert Advisors
- Sensor Synthesis and Interpretation
- Battle and Threat Assessment
- Automatic Photo Interpretation
- Tactical Planning
- Military Surveillance
- Weapon-Target Assignment
- Autonomous Vehicles
- Intelligent Robots
- Diagnosis and Maintenance Aids
- Target Location and Tracking
- Map Development Aids
- Intelligent Interactions with Knowledge Bases

International
- Aids to Understanding and Interpretation
  —goals, aspirations and motives of different countries and cultures
  —cultural models for interpreting how others perceive
- Natural Language Translation

Services
- Intelligent Knowledge Base Access
  —airline reservations
- Air Traffic Control
- Ground Traffic Control

Financial
- Tax Preparation
- Financial Expert Systems
- Intelligent Consultants

Executive Assistance
- Read Mail and Spot Items of Importance
- Planning Aids

Natural Resources
- Prospecting Aids
- Resource Operations
  —Drilling Procedures
  —Resource Recovery Guidance
- Resource Management Using Remote Sensing Data

Space
- Ground Operations Aids
- Planning and Scheduling Aids
- Diagnostic and Reconfiguration Aids
- Remote Operations of Spacecraft and Space Vehicles
- Test Monitors
- Real-time Replanning as Required by Failures, Changed Conditions or New Opportunities
- Automatic Subsystem Operations

TABLE IV-3. Potential Functional Applications of AI in NASA.

- Planning and Scheduling
- Test and Checkout
- Symbolic Computation
- Information Extraction
- Operations Management
  —Monitoring
  —Control
  —Sequencing
- System Autonomy
  —Subsystem Management
  —Fault Diagnosis
- Intelligent Assistants

TABLE IV-4. AI and NASA.

AI Opens up an Opportunity for NASA to

- Dramatically
  —Reduce Costs
  —Increase Productivity
  —Improve Quality
  —Raise Reliability
  —Utilize Facilities and People More Effectively
- Provide New Mission Capabilities
- Enable New Missions
- Improve Aerospace Science and Technology

By using AI techniques to increase human productivity and to help automate many activities previously requiring human intelligence.

V. THE PRINCIPAL PARTICIPANTS

Originally, AI was principally a research activity—the principal centers being Stanford U., M.I.T., Carnegie Mellon U. (CMU), SRI, and the U. of Edinburgh in Scotland. Research successes during the 1970's encouraged other universities to also become involved.

In the 1980's, it became apparent that AI had a large commercial and military potential. Thus, existing large computer, electronic, and multinational corporations, as well as some aerospace firms, started forming AI groups. Schlumberger, a multi-billion dollar oil exploration advisory firm, was the first to pursue AI in a big way. They now have AI groups in France, Connecticut, Texas and California.

In 1980, the Navy committed itself to building an AI Laboratory at Bolling AFB, Washington, DC to help transfer AI to Navy Applications from research at the universities. The lab now has some 20 people including Navy personnel and visiting scientists. By 1982, the Army and the Air Force also decided to form AI organizations and are now in the process of doing so.

In response to a perceived market in natural language processing, computer vision and expert systems, new small AI companies began to form, headed by former (and present) university researchers. Several dozen such companies now exist.

The computer science departments at major universities have also recently become involved, so that AI courses and beginning AI research now is evident at many universities.

Abroad, France and Great Britain have now joined Japan in evidencing major concern. The largest major commitment to AI has been by Japan, which has initiated a 10 year, one half billion dollar program to develop a "Fifth Generation Computer." This computer is to incorporate a parallel processing architecture, natural language interfacing, knowledge base management, automatic problem solving, and image understanding as the basis for a truly fast, intelligent computer. In the U.S., a new cooperative organization—Microelectronics Computer Technology Corporation (MCC)—made up of U.S. computer and electronics manufacturers, has recently been formed to be a sort of American version of the Japanese Fifth Generation Computer research project.

Thus, the AI research sponsored by DARPA, NIH, NSF, ONR and AFOSR for the past two decades has now spawned such a burgeoning AI community that it is no longer an easy task to list all those involved. (A list of participants in each of the application areas is given in the associated volumes in this series). However, Table V-1 provides an indication of the current principal players. These are given by application area, as even most research efforts initially have a specific application domain as a focus, with the results of the research usually being later generalized to cover a broader area.

TABLE V-1. The Principal Participants in AI.

1. **Universities**

Expert Systems	Computer Vision	Natural Language Processing
Stanford	CMU	Yale
MIT	U. of MD	U. of CA (Berkeley)
CMU	MIT	U. of IL
Rutgers	Stanford U.	Brown
	U. of Rochester	Stanford
	U. of MA	Rochester

2. **Non-Profit**

Expert Systems	Computer Vision	Natural Language Processing
SRI	JPL	SRI
RAND	SRI	
JPL	ERIM	
MITRE		

3. **U.S. Government**

Expert Systems	Computer Vision	Natural Language Processing
NRL AI Lab, Washington, DC	NBS, Wash., D.C.	NRL AI Lab
NOSC, San Diego		

4. **Diversified Industrial Corporations**

Expert Systems	Computer Vision	Natural Language Processing
Schlumberger	G.E.	BBN
Hewlett Packard	Hughes	IBM
Bell Labs	GM	TRW
Hughes	Westinghouse	Burroughs
IBM		SDC
DEC		Hewlett Packard
GM		Martin Marietta
Martin Marietta		Texas Instruments
Texas Instruments		Bell Labs
TRW		Sperry Univac
Xerox PARC		Lockheed Electronics Corp.
AMOCO		
United Technologies Corp.		
ATARI		
Grumman Aerospace Corp.		
Lockheed Palo Alto		
Westinghouse Electric Corp.		

TABLE V-1. (cont'd)

5. New AI Companies

Expert Systems	Computer Vision	Natural Language Processing
AIDS, Mt. View, CA	Automatix, Inc., Burlington, MA	AIC, Waltham, MA
Applied Expert Systems, Cambridge, MA	Machine Intelligence Corp., Sunnyvale, CA	Cognitive Systems Inc., New Haven, CT
Brattel Research Corp., Boston, MA	Octek, Burlington, MA	Symantec, Sunnyvale, CA
Daisy, Sunnyvale, CA		Computer Thought, Richardson, TX
Intelligent Software, Van Nuys, CA		Machine Intelligence Corp., Sunnyvale, CA
Jacor, Alexandria, VA		Weidner Communications Corp., Provo, UT
Kestrel Institute, Palo Alto, CA		
Smart Systems Technology Alexandria, VA		
Systems Control, Inc., Palo Alto, CA		
Teknowledge, Inc., Palo Alto, CA		
IntelliGenetics, Inc., Palo Alto, CA		

6. AI Computer Manufacturers

LISP Machines, Inc., Cambridge, MA, Culver City, CA
Symbolics, Cambridge, MA
Three Rivers Corp., Pittsburgh, PA
DEC, Hudson, MA
Xerox PARC, Palo Alto, CA
Daisy, Sunnyvale, CA
BBN, Cambridge, MA
MMC, Austin, TX (U.S. Fifth Generation Computer Research Consortium)

7. Major Foreign Participants

Japan

Electromechanical-Technology Lab, Tsukiba	Fifth Gen. Computer
Fujitsu-Fanuc, Ltd., Kawasaki	Fifth Gen. Computer
Hitachi, Ltd., Tokyo	Fifth Gen. Computer
Mitsubishi Elec. Corp., Tokyo	Fifth Gen. Computer
Nippon Electric Co., Ltd., Tokyo	Fifth Gen. Computer
Nippon Tele and Tele Corp., Tokyo	Fifth Gen. Computer

Great Britain
Imperial College, London
University of Edinburg, Scotland
University of Sussex, Sussex
Intelligent Terminals, Ltd.

France
University of Marseilles, Marseilles

Italy
University of Milan

VI. STATE-OF-THE-ART

General

The state-of-the-art of AI is moving rapidly as new companies enter the field, new applications are devised and existing techniques are formalized. The cutting edge of AI today is Expert Systems, with some one hundred demonstration systems having been built. With the advent of personal LISP machines and the general reduction in computing costs, development of commercial AI systems are now underway. A number of Natural Language Interfaces and Computer Vision Systems are already on the market.

Japan has focused on AI capabilities as the basis for its "Fifth Generation Computer" (Warren, 1982), and has already initiated research toward this one-half billion dollar, ten-year goal.

Britain's Industry Department has formed a study group to coordinate British AI efforts. The European Space Agency (ESA) has published a substantial survey of AI (Berger et al., 1981) from a point of view of space applications. In the U.S., DARPA has been spending in the order of $20 million annually on AI research and appears to be expanding its efforts. The U.S. Navy, Army and Air Force are all initiating substantial AI efforts. The Navy has established a major NRL AI applied research center at Bolling AFB. The Air Force is focusing their inhouse AI research efforts at Rome Air Development Center and Wright Patterson AFB.

Basic Core Topics

AI basic theory and techniques are now being codified. *The AI Handbook* (1981, 1982) (funded by DARPA) has been a major contribution in pulling together the basic theory and making it available at the graduate level for students and practitioners of AI.

Search theory is now relatively mature and well documented. A number of knowledge representation techniques have been devised and are now supported by representation languages. Basic programming languages have continued to evolve, with INTERLISP being the best supported, but Common LISP is beginning to emerge from MIT's MACLISP as the language of AI personal computers. PROLOG, a logic-based programming language, popular in Europe, appears to be the language of choice for Japan's Fifth Generation Computer, and is beginning to awaken interest in the U.S. PROLOG holds promise of reinvigorating First Order Predicate Logic as a major factor in AI for knowledge representation and problem-solving.

A large number of problem solving techniques developed during the last two decades are now forming the basis for the inference engines in Expert Systems.

Though much work remains to be done, the core topics of AI are now in a sufficient state of readiness for use in initial AI applications.

Expert Systems

Many prototype expert systems (ES) have now been built, so that ES's are no longer a rarity. However, only a few, such as MOLGEN, R1, ONCOCIN, DENDRAL, DIP-METER ADVISOR, and PUFF, are in actual commercial use on a regular basis. Expert systems are still restricted to a narrow domain of expertise and require laborious construction via interaction with human experts.

Further, these systems tend to have the characteristics of:

—Providing satisfactory, rather than optimum solutions

—Providing satisfactory answers only a percentage of the time

—Some of the time being unable to provide an answer

This is in contrast to normal engineering solutions that are algorithmic in nature and virtually always provide a satisfactory answer when supplied with appropriate inputs. This "sometimes the answer is wrong" characteristic of ES is also characteristic of human decision-making. Thus, at the moment, expert systems tend to be used as human assistants (with humans making the final decisions) rather than as "stand-alone" autonomous systems.

Natural Language

Natural language interfaces (NLI's) were the first commercial AI product. ROBOT, (now INTELLECT, by the Artificial Intelligence Corp., Waltham, MA), in 1980 was the first in the market and now exists in over 200 installations. Some half dozen other commercial NLI systems are now available. All these systems are restricted to limited sets of natural language and exhibit occasional failures in understanding or processing a user's input. However, with a little training of the users, NLI's have proved very useful.

Several commercial Machine Translation Systems are also now available. These are not used as completely automatic systems, as in many cases their translation is very rough or even incorrect. However, as an aid to a human translator, they can improve productivity by a factor of 2 to 10, depending on the system and the material being translated.

Text Understanding and Text Generation are still in the research stage.

Computer Vision

Computer Vision has entered the commercial market, with some dozen companies offering sophisticated commercial vision systems. These systems are operating successfully in specialized environments on low level problems of verification, inspection, recognition, and determination of object location and/or orientation. Current commercial vision systems deal primarily with two dimensional images—they can't handle three-dimensional analysis needed to recognize objects from arbitrary viewpoints.

Though quite a number of high level research vision systems have been explored, no general vision system is available today or is imminent. Major current efforts in this area are ACRONYM at Stanford U., VISIONS at the U. of Mass, and the robotic vision effort at NBS.

Conclusions

Figure VI-1 is a list of overall conclusions on the current state of AI. Summarizing, it appears that technology is now ready for early applications. However, the fact that current AI systems are prone to error, suggests that current AI applications should be focused on intelligent aids for humans, rather than on truly autonomous systems.

- Basic principles and techniques devised and demonstrated
- Initial languages, programs and tools developed
- Software portability a problem
- A few initial applications already in use
- Technology is now ready for early applications
- Current technology more appropriate for intelligent assistants than for autonomous systems
- Customizing, adapting and usually writing own programs necessary
- Because of huge potential benefits, utilization will be explosive as technology is further rationalized

Figure VI-1. Conclusions on the Current State-of-the-Art in AI.

CHAPTER VI REFERENCES

- Berger, G., Havas, R. and Prajoux, R., "Survey of the State-of-the-Art in Robotics and Artificial Intelligence," MATRA Report 0361/DX60, European Space Agency, August 1981.
- Warren, D.H.D., "A View of the Fifth Generation Computer and its Impact," *AI Magazine,* Vol. 3, No. 4, Fall 1982, pp. 34-39.
- Barr, A. and Feigenbaum, E.A. (Eds.), *The Handbook of Artificial Intelligence,* Vols. I and II, Los Altos, CA, W. Kaufmann, 1981, 1982.
- Cohen, P.R. and Feigenbaum, E.A. (Eds.), *The Handbook of Artificial Intelligence,* Vol. III, Los Altos, CA, W. Kaufmann, 1982.

VII. TOWARDS THE FUTURE

General

Today's initial AI systems can primarily be regarded as intelligent assistants. These are taking the form of expert systems, natural language interfaces, computer vision systems and intelligent computer-aided instruction systems. They—like humans—are all prone to failures, but unlike humans, they are not capable of drawing on deep knowledge when needed to achieve graceful degradation, so that their failures are more abrupt. Thus, researchers are currently engaged in developing a new set of advanced systems, based on deep knowledge—which includes such aspects as causal models and scientific knowledge.

Expert Systems

Utilizing emerging expert-system building tools, AI developers are expected to eventually put expert medical, financial and legal advice at the fingertips of anyone with access to a personal computer (though this will probably have to await the arrival of a new generation using 32 bit microprocessors). Expert systems will also put expertise in the hands of less-trained, lower-salaried workers.

Natural Language

Speech recognition appears to be emerging as a key man-machine interface. Researchers have found that the psychological problems inherent in talking to a machine are a barrier to the acceptance of speech interfaces. Overcoming the psychological problems may be even more important than reducing cost. To achieve practical continuous speech recognition, systems will have to expand today's vocabularies by an order of magnitude, increase speed by two orders of magnitude, and get costs below $1,000. Such systems are estimated to still be five years away.

Natural language interfaces appear to be the way to vastly increase the number of people who can interact with computers. Systems with near natural-language capabilities are available now, though it will be years before the systems can handle truly unrestricted dialogue. It is estimated that public access to large data bases via computer using restricted speech understanding may begin to appear within three years. This can be expected to open up a whole new industry of automated reservation, shopping and information services accessed by telephone.

Another emerging aspect of natural language processing are systems that understand text by utilizing world knowledge. Such systems could read and summarize news stories (as is now being done in research) but more likely would be applied to such tasks as reading mail and informing the recipient of important items, or in general, processing large amounts of information for humans trying to escape from overload.

Computer Vision

Computer vision will increasingly be used in industry for inspection, identification, verification, part location, and other purposes. Vision provides the most general purpose sensory input for intelligent robots. It is likely that roughly 25% of all robots will utilize vision by the end of the decade. Vision is also expected to play a large part in military automation, remote sensing, and as aids to the handicapped.

Intelligent Robots

The development of AI is making intelligent robots feasible. As intelligence is added to robots, they will not only be able to perform more flexibly in manufacturing, but will begin to be evident in tasks outside the industrial environment. Thus, robots in firefighting, underseas exploration, mining, and construction will appear. However, the big push may be in military applications with its actively hostile environments. In the 1990s, robots with intelligence and sensory capabilities will appear in the service industries — in everything from food service to household robots. It is also anticipated that in the 1990s, intelligent robots will enter the space arena, for such tasks as the construction and assembly of large space structures, space manufacture, extra-terrestrial mining and exploration, and operation and maintenance of space installations.

Industrial Applications

In addition to more intelligent robots, AI will influence virtually every aspect of the future industrial plant. Integrated plants that make usé of automated planning, scheduling, process control, warehousing, and the operation of automated robot carts, robots, and manufacturing machines, will appear in a few years and will become widespread within the next 10 years.

Computers for Future Automation

Computers and special purpose chips designed to incorporate parallel processing are being developed at several universities and computer organizations. MIT has been developing a parallel machine using VLSI techniques to break problems into subproblems and distribute them among its processors. Another chip will utilize parallel processing to rapidly search through the branches of a semantic network.

The most prominent future system is Japan's Fifth Generation Computer that could store and retrieve some 20,000 rules, incorporate a knowledge base of 100 million data items and help make Japan an AI leader before the end of this century. To help maintain U.S. competitiveness, a dozen of the largest U.S. electronics and computer companies have recently set up the Microelectronics and Computer Technology Corp. (MCC). This well-funded cooperative research venture is designed to develop a broad base of fundamental technologies. Among them is a 10-year program to develop advanced computer architectures and artificial intelligence.

Computer Aided Instruction (CAI)

CAI systems may produce one of the most dramatic changes of all. Education consumes some 10% of the U.S. gross national product today. Systems that will enable students to ask questions and receive insightful answers may begin to overcome the barriers of instruction by machines. Computer systems that model the student based on his or her response can gear instruction to the student's level of ability and interest, something not easily done in a conventional classroom.

To truly learn is to digest and make the material one's own by updating one's internal models and using them in new applications. Real time interaction with a computer providing immediate feedback and individual guidance is particularly appropriate to this goal.

Thus, as computer hardware costs continue to tumble, the nature of the entire present educational system may be radically changed. For adults and members of the armed forces, CAI will probably rapidly become the standard form of instruction.

Learning by Computers

The real breakthrough may come when machine learning is achieved. Already several learning systems, currently in the research stage, have been able to produce very interesting results. Someday machines will able to learn throughout their lifetime, building up the knowledge base needed for advanced reasoning. This will open up spectacular new applications in offices, factories and homes.

Machines may update their knowledge by reading natural language material, as well as learning by experience from the problems the computers are called upon to solve. Computers also may be able to form conclusions from examination of multiple data bases, thereby building new knowledge from existing knowledge.

The Social Impacts

The U.S. Defense Science Board has ranked AI as one of the top 10 military technologies of the 80's. Not only will human-level expertise and decision making capabilities show up in machines, but the task of achieving these results will help us understand how our minds work as well.

Combining expert systems and computer graphics will enable people to "see" the results of the computer actions. This will not only clarify and simplify the interaction, but will greatly speed human learning and decision making. The result may be to compress months of research and engineering experience gained the old way into insights gathered from just a few hours interaction with intelligent computer programs.

AI's effects on society may be slow at first, but by the end of the century the results should be revolutionary. The shift in employment away from manufacturing may be as dramatic as the shift away from agriculture. There will also be a revolution in white collar work—service, research, leisure. How to restructure society to take advantage of a potential abundance of goods and services, or to adapt to new work opportunities and leisure activities, may be the question of the century. This may give society another chance to pursue the social and mental goals so often deferred. It also may at last free us from the monetary and technical bonds to Earth. Perhaps, we can at last "reach for the stars."

SOURCES FOR FURTHER INFORMATION

1. Journals

- *SIGART Newsletter* — ACM (Association for Computing Machinery).
- *Artificial Intelligence*
- *Cognitive Science* — Cognitive Science Society
- *AI Magazine* — American Association for AI (AAAI)
- *Pattern Analysis and Machine Intelligence* — IEEE
- *International Journal on Robotics Research*
- *IEEE Transactions on Systems, Man and Cybernetics*

2. Conferences

- International Joint Conference on AI (IJCAI) - biannual. Current one: Aug. 1983 in Germany.
- AAAI Annual Conference
- IEEE Systems, Man & Cybernetics Annual Conference

3. Recent Books

- Barr, A. and Feigenbaum, E.A., *The Handbook of Artificial Intelligence,* Vols. I, II, Los Altos, CA, W. Kaufmann, 1981, 1982.
- Clocksin, W.F. and Mellish, C.S., *Programming in Prolog*, New York, Springer-Verlag, 1981.
- Cohen, P.R. and Feigenbaum, E.A., (Eds.), *The Handbook of Artificial Intelligence,* Vol. III, Los Altos, CA, W. Kaufmann, 1982.
- Davis, R., and Lenat, D.B., *Knowledge-Based Systems in Artificial Intelligence*, New York, McGraw-Hill, 1982.
- Feigenbaum, E.A. and McCorduck, P. *The Fifth Generation*, Reading, Mass, Addison-Wesley, 1983.
- Hayes-Roth, F. (Ed.), *Building Expert Systems,* Reading, Mass., Addison-Wesley, 1983.
- Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (Eds.), *Machine Learning — An Artificial Intelligence Approach,* Palo Alto, Tioga, 1983.
- Nilsson, N.J., *Principles of Artificial Intelligence,* Palo Alto, CA, Tioga, 1980.
- Rich, E., *Artificial Intelligence,* New York, McGraw Hill, 1983.
- Simon, H.A., *The Sciences of the Artificial, 2nd Ed.*, Cambridge, Mass, M.I.T. Press, 1981.
- Sowa, J.F., *Conceptual Structures, Information Processing in Mind and Machine,* Reading Mass, Addison-Wesley, 1983.
- Szolovits, P. (Ed.), *Artificial Intelligence in Medicine,* Boulder, CO, Westview Press, 1982.
- Wilensky, R. *Planning and Understanding,* Reading, MA, Addison-Wesley, 1982.
- Winston, P.H. and Horn, B.K.P., *LISP*, Reading, Mass., Addison-Wesley, 1981.

GLOSSARY

A

Activation Mechanism: The situation required to invoke a procedure—usually a match of the system state to the preconditions required to exercise a production rule.

AI Handbook: The Handbook of Artificial Intelligence, E. A. Feigenbaum, A. Barr and P. R. Cohen (Eds.). Published by W. Kaufmann, Los Altos, CA in 1981 and 1982. This important project was supported by DARPA and NIH.

Algorithm: A procedure for solving a problem in a finite number of steps.

AND/OR Graph: A generalized representation for problem reduction situations and two person games. A tree-like structure with two types of nodes. Those for which several successors of a node have to be accomplished (or considered) are AND nodes. Those for which only one of several of the node successors are necessary are OR nodes. (In about half the literature the labeling of AND and OR nodes is reversed from this definition.)

Antecedent: The lefthand side of a production rule. The pattern needed to make the rule applicable.

Argument Form: A reasoning procedure in logic.

ARPANET: A network of computers and computational resources used by the U.S. AI community and sponsored by DARPA (Defense Advanced Research Projects Agency).

Artificial Intelligence (AI): A discipline devoted to developing and applying computational approaches to intelligent behavior. Also referred to as machine intelligence or heuristic programming.

Artificial Intelligence (AI) Approach: An approach that has its emphasis on symbolic processes for representing and manipulating knowledge in a problem solving mode.

Atom: An individual. A proposition in logic that cannot be broken down into other propositions. An indivisible element.

Autonomous: A system capable of independent action.

B

Backtracking: Returning (usually due to depth-first search failure) to an earlier point in a search space. Also a name given to depth-first backward reasoning.

Backward Chaining: A form of reasoning starting with a goal and recursively chaining backwards to its antecedent goals or states by applying applicable operators until an appropriate earlier state is reached or the system backtracks. This is a form of depth-first search. When the application of operators changes a single goal or state into multiple goals or states, the approach is referred to as problem reduction.

Blackboard Approach: A problem-solving approach whereby the various system elements communicate with each other via a common working data storage called the blackboard.

Blind Search: An ordered approach that does not rely on knowledge for searching for a solution.

Blocks World: A small artificial world, consisting of blocks and pyramids, used to develop ideas in computer vision, robotics, and natural language interfaces.

Bottom-Up Control Structure: A problem-solving approach that employs forward reasoning from current or initial conditions. Also referred to as an event-driven or data-driven control structure.

Breadth-First Search: An approach in which, starting with the root node, the nodes in the search tree are generated and examined level by level (before proceeding deeper). This approach is guaranteed to find an optimal solution if it exists.

C

Clause: A syntactic construction containing a subject and a predicate and forming part of a statement in logic or part of a sentence in a grammar.

Cognition: An intellectual process by which knowledge is gained about perceptions or ideas.

Combinatorial Explosion: The rapid growth of possibilities as the search space expands. If each branch point (decision point) has an average of n branches, the search space tends to expand as n^d, as the depth of search, d, increases.

Common Sense: The ability to act appropriately in everyday situations based on one's lifetime accumulation of experiential knowledge.

Common Sense Reasoning: Low level reasoning based on a wealth of experience.

Compile: The act of translating a computer program written in a high level language (such as LISP) into the machine language which controls the basic operations of the computer.

Computational Logic: A science designed to make use of computers in logic calculus.

Computer Architecture: The manner in which various computational elements are interconnected to achieve a computational function.

Computer Graphics: Visual representations generated by a computer (usually observed on a monitoring screen).

Computer Network: An interconnected set of communicating computers.

Computer Vision (Computational or Machine Vision): Perception by a computer, based on visual sensory input, in which a symbolic description is developed of a scene depicted in an image. It is often a knowledge-based, expectation-guided process that uses models to interpret sensory data. Used somewhat synonymously with image understanding and scene analysis.

Conceptual Dependency: An approach to natural language understanding in which sentences are translated into basic concepts expressed as a small set of semantic primatives.

Conflict Resolution: Selecting a procedure from a conflict set of applicable competing procedures or rules.

Conflict Set: The set of rules which matches some data or pattern in the global data base.

Conjunct: One of several subproblems. Each of the component formulas in a logical conjunction.

Conjunction: A problem composed of several subproblems. A logical formula built by connecting other formulas by logical ANDs.

Connectives: Operators (e.g., AND, OR) connecting statements in logic so that the truth-value of the composite is determined by the truth-value of the components.

Consequent: The right side of a production rule. The result of applying a procedure.

Constraint Propagation: A method for limiting search by requiring that certain constraints be satisified. It can also be viewed as a mechanism for moving information between subproblems.

Context: The set of circumstances or facts that define a particular situation, event, etc. The portion of the situation that remains the same when an operator is applied in a problem-solving situation.

Control Structure: Reasoning strategy. The strategy for manipulating the domain knowledge to arrive at a problem solution.

D

Data Base: An organized collection of data about some subject.

Data Base Management System: A computer system for the storage and retrieval of information about some domain.

Data-Driven: A forward reasoning, bottom-up problem solving approach.

Data Structure: The form in which data is stored in a computer.

Debugging: Correcting errors in a plan.

DEC: Digital Equipment Company.

Declarative Knowledge Representation: Representation of facts and assertions.

Deduction: A process of reasoning in which the conclusion follows from the premises given.

Default Value: A value to be used when the actual value is unknown.

Depth-First Search: A search that proceeds from the root node to one of the successor nodes and then to one of that node's successor nodes, etc., until a solution is reached or the search is forced to backtrack.

Difference Reduction: "Means-Ends" analysis. An approach to problem solving that tries to solve a problem by iteratively applying operators that will reduce the difference between the current state and the goal state.

Directed Graph: A knowledge representation structure consisting of nodes (representing, e.g., objects) and directed connecting arcs (labeled edges, representing, e.g., relations).

Disproving: An attempt to prove the impossibility of a hypothesized conclusion (theorem) or goal.

Domain: The problem area of interest, e.g., bacterial infections, prospecting, VLSI design.

E

Editor: A software tool to aid in modifying a software program.

Embed: To write a computer language on top of (embedded in) another computer language (such as LISP).

Emulate: To perform like another system.

Equivalent: Has the same truth value (in logic).

Evaluation Function: A function (usually heuristic) used to evaluate the merit of the various paths emanating from a node in a search tree.

Event-Driven: A forward-chaining problem-solving approach based on the current problem status.

Expectation-Driven: Processing approaches that proceed by trying to confirm models, situations, states or concepts anticipated by the system.

Expert System: A computer program that uses knowledge and reasoning techniques to solve problems normally requiring the abilities of human experts.

F

Fault Diagnosis: Determining the trouble source in an electro-mechanical system.

Fifth Generation Computer: A non-Von Neumann, intelligent, parallel-processing form of computer now being pursued by Japan.

First Order Predicate Logic: A popular form of logic used by the AI community for representing knowledge and performing logical inference. First Order Predicate Logic permits assertions to be made about variables in a proposition.

Forward Chaining: Event-driven or data-driven reasoning.

Frame: A data structure for representing stereotyped objects or situations. A frame has slots to be filled for objects and relations appropriate to the situation.

FRANZLISP: The dialect of LISP developed at the U. of CA, Berkeley.

Functional Application: The generic task or function performed in an application.

Fuzzy Set: A generalization of set theory that allows for various degrees of set membership, rather than all or none.

G

Garbage Collection: A technique for recycling computer memory cells no longer in use.

General Problem Solver (GPS): The first problem solver (1957) to separate its problem-solving methods from knowledge of the specific task being considered. The GPS problem-solving approach employed was "means-ends analysis."

Generate and Test: A common form of state space search based on reasoning by elimination. The system generates possible solutions and the tester prunes those solutions that fail to meet appropriate criteria.

Global Data Base: Complete data base describing the specific problem, its status and that of the solution process.

Goal Driven: A problem-solving approach that works backward from the goal.

Goal Regression: A technique for constructing a plan by solving one conjunctive subgoal at a time, checking to see that each solution does not interfere with the other subgoals that have already been achieved. If interferences occur, the offending subgoal is moved to an earlier non-interfering point in the sequence of subgoal accomplishments.

Graph: A set of nodes connected by arcs.

H

Heuristics: Rules of thumb or empirical knowledge used to help guide a problem solution.

Heuristic Search Techniques: Graph searching methods that use heuristic knowledge about the domain to help focus the search. They operate by generating and testing intermediate states along potential solution paths.

Hierarchical Planning: A planning approach in which first a high level plan is formulated considering only the important (or major) aspects. Then the major steps of the plan are refined into more detailed subplans.

Hierarchy: A system of things ranked one above the other.

Higher Order Language (HOL): A computer language (such as FORTRAN or LISP) requiring fewer statements than machine language and usually substantially easier to use and read.

Horn Clause: A set of statements joined by logical AND's. Used in PROLOG.

I

Identity: Two propositions (in logic) that have the same truth value.

Image Understanding (IU): Visual perception by a computer employing geometric modeling and the AI techniques of knowledge representation and cognitive processing to develop scene interpretations from image data. IU has dealt extensively with 3D objects.

Implies: A connective in logic that indicates that if the first statement is true, the statement following is also true.

Individual: A non-variable element (or atom) in logic that cannot be broken down further.

Infer: To derive by reasoning. To conclude or judge from premises or evidence.

Inference: The process of reaching a conclusion based on an initial set of propositons, the truths of which are known or assumed.

Inference Engine: Another name given to the control structure of an AI problem solver in which the control is separate from the knowledge.

Instantiation: Replacing a variable by an instance (an individual) that satisfies the system (or satisfies the statement in which the variable appears).

Intelligence: The degree to which an individual can successfully respond to new situations or problems. It is based on the individual's knowledge level and the ability to appropriately manipulate and reformulate that knowledge (and incoming data) as required by the situation or problem.

Intelligent Assistant: An AI computer program (usually an expert system) that aids a person in the performance of a task.

Interactive Environment: A computational system in which the user interacts (dialogues) with the system (in real time) during the process of developing or running a computer program.

Interface: The system by which the user interacts with the computer. In general, the junction between two components.

INTERLISP: A dialect of LISP (used at Stanford U.) developed at BBN and XEROX-PARC.

Invoke: To place into action (usually by satisfying a precondition).

K

Knowledge Base: AI databases that are not merely files of uniform content, but are collections of facts, inferences and procedures, corresponding to the types of information needed for problem solution.

Knowledge Base Management: Management of a knowledge base in terms of storing, accessing and reasoning with the knowledge.

Knowledge Engineering: The AI approach focusing on the use of knowledge (e.g., as in expert systems) to solve problems.

Knowledge Representation (KR): The form of the data-structure used to organize the knowledge required for a problem.

Knowledge Source: An expert system component that deals with a specific area or activity.

L

Leaf: A terminal node in a tree representaton.

Least Commitment: A technique for coordinating decision making with the availability of information, so that problem-solving decisions are not made arbitrarily or prematurely, but are postponed until there is enough information.

List: A sequence of zero or more elements enclosed in a pair of parentheses, where each element is either an atom (an indivisible element) or a list.

List Processing Language (LISP): The basic AI programming language.

Logical Operation: Execution of a single computer instruction.

Logical Representation: Knowledge representation by a collection of logical formulas (usually in First Order Predicate Logic) that provide a partial description of the world.

M

MACLISP: A dialect of LISP developed at M.I.T.

Means-Ends Analysis: A problem-solving approach (used by GPS) in which problem-solving operators are chosen in an iterative fashion to reduce the difference between the current problem-solving state and the goal state.

Meta-Rule: A higher level rule used to reason about lower level rules.

Microcode: A computer program at the basic machine level.

Model Driven: A top-down approach to problem-solving in which the inferences to be verified are based on the domain model used by the problem-solver.

Modus Ponens: A mathematical form of argument in deductive logic. It has the form:

If A is true, then B is true.
A is true
Therefore B is true.

N

Natural Deduction: Informal reasoning.

Natural Language Interface (NLI): A system for communicating with a computer by using a natural language.

Natural Language Processing (NLP): Processing of natural language (e.g., English) by a computer to facilitate communication with the computer, or for other purposes such as language translation.

Natural Language Understanding (NLU): Response by a computer based on the meaning of a natural language input.

Negate: To change a proposition into its opposite.

Node: A point (representing such aspects as the system state or an object) in a graph connected to other points in the graph by arcs (usually representing relations).

Non-Monotonic Logic: A logic in which results are subject to revision as more information is gathered.

O

Object-Oriented Programming: A programming approach focused on objects which communicate by message passing. An object is considered to be a package of information and descriptions of procedures that can manipulate that information.

Operators: Procedures or generalized actions that can be used for changing situatons.

P

Parallel Processing: Simultaneous processing, as opposed to the sequential processing in a conventional (Von Neumann) type of computer architecture.

Path: A particular track through a state graph.

Pattern Directed Invocation: The activation of procedures by matching their antecedent parts to patterns present in the global data base (the system status).

Pattern Matching: Matching patterns in a statement or image against patterns in a global data base, templates or models.

Pattern Recognition: The process of classifying data into predetermined categories.

Perception: An active process in which hypotheses are formed about the nature of the environment, or sensory information is sought to confirm or refute hypotheses.

Personal AI Computer: New, small, interactive, stand-alone computers for use by an AI researcher in developing AI programs. Usually specifically designed to run an AI language such as LISP.

Plan: A sequence of actions to transform an initial situation into a situation satisfying the goal conditions.

Portability: The ease with which a computer program developed in one programming environment can be transferred to another.

Predicate: That part of a proposition that makes an assertion (e.g., states a relation or attribute) about individuals.

Predicate Logic: A modification of Propositional Logic to allow the use of variables and functions of variables.

Prefix Notation: A list representation (used in LISP programming) in which the connective, function or predicate is given before the arguments.

Premise: A first proposition on which subsequent reasoning rests.

Problem Reduction: A problem-solving approach in which operators are used to change a single problem into several subproblems (which are usually easier to solve).

Problem-Solving: A procedure using a control strategy to apply operators to a situation to try to achieve a goal.

Problem State: The condition of the problem at a particular instant.

Procedural Knowledge Representation: A representation of knowledge about the world by a set of procedures - small programs that know how do specific things (how to proceed in well-specified situations).

Production Rule: A modular knowledge structure representing a single chunk of knowledge, usually in If-Then or Antecedent-Consequent form. Popular in Expert Systems.

Programming Environment: The total programming set-up that includes the interface, the languages, the editors and other programming tools.

Programming in Logic (PROLOG): A logic-oriented AI language developed in France and popular in Europe and Japan.

Property List: A knowledge representation technique by which the state of the world is described by objects in the world via lists of their pertinent properties.

Proposition: A statement (in logic) that can be true or false.

Propositional Logic: An elementary logic that uses argument forms to deduce the truth or falsehood of a new proposition from known propositions.

Prototype: An initial model or system that is used as a base for constructing future models or systems.

Pseudo-Reduction: An approach to solving the difficult problem case where multiple goals must be satisified simultaneously. Plans are found to achieve each goal independently and then integrated using knowledge of how plan segments can be intertwined without destroying their important effects.

R

Recursive Operations: Operations defined in terms of themselves.

Relaxation Approach: An iterative problem-solving approach in which initial conditions are propagated utilizing constraints until all goal conditions are adequately satisfied.

Relevant Backtracking (Dependency-Directed or Non-Chronological Backtracking): Backtracking (during a search) not to the most recent choice point, but to the most relevant choice point.

Resolution: A general, automatic, syntactic method for determining if a hypothesized conclusion (theorem) follows from a given set of premises (axioms).

Root Node: The initial (apex) node in a tree representation.

Rule-Interpreter: The control structure for a production rule system.

S

Satisficing: Developing a satisfactory, but not necessarily optimum, solution.

Scheduling: Developing a time sequence of things to be done.

Scripts: Frame-like structures for representing sequences of events.

Search Space: The implicit graph representing all the possible states of the system which may have to be searched to find a solution. In many cases the search space is infinite. The term search space is also used for non-state-space representations.

Semantic: Of or relating to meaning.

Semantic Network: A knowledge representation for describing the properties and relations of objects, events, concepts, situations or actions, by a directed graph consisting of nodes and labeled edges (arcs connecting nodes).

Semantic Primitives: Basic conceptual units in which concepts, ideas or events can be represented.

S-Expression: A symbolic expression. In LISP, a sequence of zero or more atoms or S-expressions enclosed in parentheses.

Slot: An element in a frame representation to be filled with designated information about the particular situation.

Software: A computer program.

Solution Path: A successful path through a search space.

Speech Recognition: Recognition by a computer (primarily by pattern-matching) of spoken words or sentences.

Speech Synthesis: Developing spoken speech from text or other representations.

Speech Understanding: Speech perception by a computer.

SRI Vision Module: An important object recognition, inspection, orientation and location research vision system developed at SRI. This system converted the scene into a binary image and extracted the calculated needed vision parameters in real time, as it sequentially scanned the image line by line.

State Graph: A graph in which the nodes represent the system state and the connecting arcs represent the operators which can be used to transform the state from which the arcs emanate to the state at which they arrive.

Stereotyped Situation: A generic, recurrent situation such as "eating at a restaurant" or "driving to work."

Subgoals: Goals that must be achieved to achieve the original goal.

Subplan: A plan to solve a portion of the problem.

Subproblems: The set of secondary problems that must be solved to solve the original problem.

Syllogism: A deductive argument in logic whose conclusion is supported by two premises.

Symbolic: Relating to the substitution of abstract representations (symbols) for concrete objects.

Syntax: The order or arrangement (e.g., the grammar of a language).

T

Terminal Node (Leaf Node): The final node emanating from a branch in a tree or graph representation.

Theorem: A proposition, or statement, to be proved based on a given set of premises.

Theorem Proving: A problem-solving approach in which a hypothesized conclusion (theorem) is validated using deductive logic.

Time-Sharing: A computer environment in which multiple users can use the computer virtually simultaneously via a program that time-allocates the use of computer resources among the users in a near-optimum manner.

Top-Down Approach: An approach to problem-solving that is goal-directed or expectation-guided based on models or other knowledge. Sometimes referred to as "Hypothesize and Test."

Top-Down Logic: A problem-solving approach used in production systems, where production rules are employed to find a solution path by chaining backwards from the goal.

Tree Structure: A graph in which one node, the root, has no predecessor node, and all other nodes have exactly one predecessor. For a state space representation, the tree starts with a root node (representing the initial problem situation). Each of the new states that can be produced from this initial state by application of a single operator is represented by a successor node of the root node. Each successor node branches in a similar way until no further states can be generated or a solution is reached. Operators are represented by the directed arcs from the nodes to their successor nodes.

Truth-Maintenance: A method of keeping track of beliefs (and their justifications) developed during problem-solving, so that if contradictions occur, the incorrect beliefs or lines of reasoning, and all conclusions resulting from them, can be retracted.

Truth Value: One of the two possible values—True or False—associated with a proposition in logic.

U

Unification: The name for the procedure for carrying out instantiations. In unification, the attempt is to find substitutions for variables that will make two atoms identical.

V

Variable: A quantity or function that may assume any given value or set of values.

Von-Neuman Architecture: The current standard computer architecture that uses sequential processing.

W

World Knowledge: Knowledge about the world (or domain of interest).

World Model: A representation of the current situation.

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 83-~~2687~~ 2799	2. Performing Organ. Report No.	3. Publication Date February 1984

4. TITLE AND SUBTITLE

AN OVERVIEW OF ARTIFICIAL INTELLIGENCE AND ROBOTICS
Volume I - Artificial Intelligence
Part A - The Core Ingredients

5. AUTHOR(S)

William B. Gevarter

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

National Aeronautics and Space Administration
Washington, DC 20546

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

Artificial Intelligence (AI) is an emerging technology that has recently attracted considerable attention. Many applications are now under development. The goal of Artificial Intelligence is focused on developing computational approaches to intelligent behavior. This goal is so broad--covering virtually all aspects of human cognitive activity--that substantial confusion has arisen as to the actual nature of AI, its current status and its future capability. This volume, the first in a series of NBS/NASA reports on the subject, attempts to address these concerns. Thus, this report endeavors to clarify what AI is, the foundations on which it rests, the techniques utilized, applications, the participants and, finally, AI's state-of-the-art and future trends.

It is anticipated that this report will prove useful to government and private engineering and research managers, potential users, and others who will be affected by this field as it unfolds.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Artificial intelligence; machine intelligence, heuristic programming; computers; expert systems; computer vision, natural language processing; automation; overview

13. AVAILABILITY

☒ Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

74

15. Price

$10.00

USCOMM-DC 6043-P80