Report 84-32
Stanford -- KSL

Scientific DataLink

Artificial Intelligence: Toward Machines that Think.
Bruce G. Buchanan,
Jul 1984

card 1 of 1

Heuristic Programming Project July 1984

HPP 84-32

Artificial Intelligence:

Toward Machines that Think

Bruce G. Buchanan
Department of Computer Science
Stanford University

ARTIFICIAL INTELLIGENCE

by Bruce G. Buchanan

Computer scientists are striving to endow their electronic creations with abilities similar to human reasoning and learning. "Expert" consultation systems that can diagnose disease, analyze locomotive problems, or do theoretical chemistry are recent payoffs.

Artificial intelligence (AI) is the branch of computer science that deals with ways of representing knowledge using symbols rather than numbers and with rules-of-thumb or heuristic methods for processing information. The major goal of AI is to understand intelligence—loosely, the processes that humans and other animals use to learn, solve problems, and understand language—by constructing or programming machines that behave intelligently. Artificial intelligence differs from other areas of computing in its attention to both symbolic information and heuristic methods for solving problems.

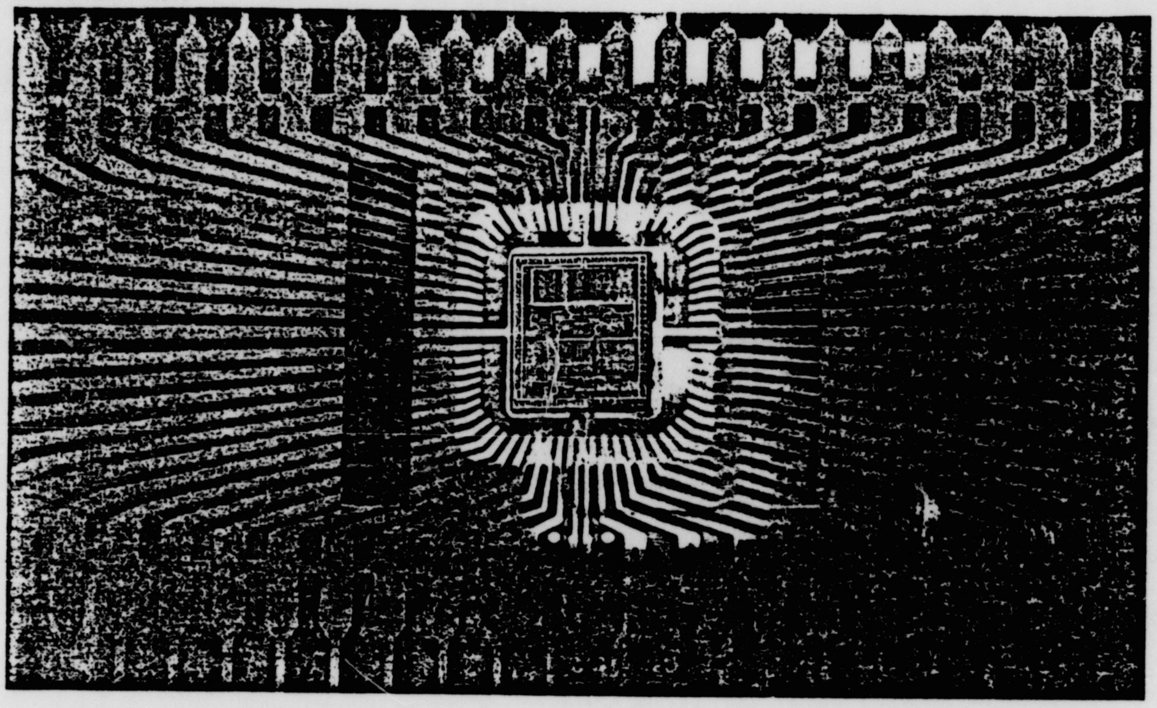
In general, the kinds of problems that AI methods address are not well structured. That is, one does not already know in advance from a description of the problem alone what is the best method for solving it. In short, there are no algorithms, *i.e.*, no straightforward step-by-step procedures for solution. Broadly speaking, AI substitutes exploratory search for precise algorithmic methods.

Fundamental themes in AI

Although the intellectual roots of artificial intelligence research stretch back to antiquity, its modern history began with work during the 1950s. Early, influential products that emerged from these investigations were a computer program that proved theorems of propositional logic, a program for the game of checkers that learned from its mistakes, a theorem prover for plane geometry, a program that recognized handwritten characters, and programming languages for general manipulation of symbols.

Alan Turing, Norbert Wiener, and other pioneers in computing foresaw the use of electronic computers for intelligent problem solving. In the 1940s and early 1950s chess and other games were considered to be prototypical of many reasoning problems because information about these games is not naturally represented mathematically, if at all, and because solving these problems requires heuristics as it is far too time-consuming to look exhaustively at all possible sequences of moves. Today, games and game playing remain a popular and convenient focus of much research in AI because of the tidy, well-defined nature of the subject.

Erich Hartmann—Magnum



Motorola 68000 microprocessor is one of the new 16-bit integrated circuits that have allowed computers of small size and considerable power to be developed.

the end of the century is not clear. Many paths are open, and surely more trails will be discovered in ensuing years. Consideration of the phenomenal progress of the past 30 years leaves one with a feeling of anticipation for what is yet to come. The only certainty in sight is that scientists, engineers, and consumers alike will continue to be amazed by the continuing "microelectronics revolution."

FOR ADDITIONAL READING

F. L. Carter, *Molecular Electronic Devices* (Marcel Dekker, Inc., 1982).

D. A. Doane, D. B. Fraser, and D. W. Hess (eds.), *Semiconductor Technology* (The Electrochemical Society, Inc., Pennington, 1982).

N. G. Einspruch (ed.), *VLSI Electronics: Microstructure Science*, vol. 1–7 (Academic Press, 1981–83).

T. Forester (ed.), *The Microelectronic Revolution* (The MIT Press, 1981).

R. E. Howard, P. F. Liao, W. J. Skocpol, L. D. Jackel, and H. G. Craighead, "Microfabrication as a Scientific Tool," *Science* (July 8, 1983, pp. 117–121).

E. L. Hu, "New Approaches to the Fabrication of Submicron Switches," *American Scientist* (September-October 1981, pp. 517–521).

S. M. Sze (ed.), *VLSI Technology* (McGraw-Hill, 1983).

TOWARD MACHINES THAT THINK

__BRUCE G. BUCHANAN__ is Professor of Computer Science Research at Stanford University and has worked in the field of artificial intelligence since 1966.

(Overleaf) Illustration by Fred Nelson

In addition to game playing early AI work focused on techniques for solving small symbolic reasoning problems, on robotics and associated vision and manipulation systems, and on understanding instructions given in natural human language. Researchers continue to ponder these problems as well. There was also considerable work on self-improving machines like the learning program for checkers, on pattern recognition like the character-recognition program, and machine translation of text from one language to another.

Another important dimension of work in AI was, and still is, psychological modeling. Not only have studies of the ways humans solve problems provided clues for how to program machines to behave intelligently but computer programs have helped test psychological models of human intelligence. Early work in AI gave rise to a branch of psychology known as information-processing psychology, and the intersection of AI and psychology (also overlapping philosophy, linguistics, and other related disciplines) is now known as cognitive science.

Two research themes, representation and search, dominated the first decade or so of AI work. How one represents descriptions of objects, processes, properties, and relations greatly influences the efficiency of the problem-solving system and the quality of the solutions. How one searches for plausible solutions and uses heuristics to guide the search also influences the efficiency and effectiveness of the system.

Search is the fundamental problem-solving paradigm for AI programs. Conceptually, if not practically, one can consider most problem-solving activity as the generation and selection of alternative solutions. Even random trial and error, which is not usually considered intelligent, can be viewed as generating a solution and then testing to see if the solution is correct. The section below on control discusses variations on the generate-and-test method that lead to more intelligent behavior.

Heuristic methods, rules that in the human sense could be said to be based on experience or judgment or even "intuition," are central to research in artificial intelligence. They frequently lead to plausible solutions to problems or increase the efficiency of a problem-solving procedure. Whereas algorithms guarantee a correct solution (if there is one) in a finite time, heuristics only increase the likelihood of finding a plausible solution or increase the plausibility of the solutions. In safecracking, to use a classic example, a time-consuming but unfailing algorithm for opening a combination safe is to try all possible combinations of numbers on the dials. One heuristic that may work, and that will speed up the process if it does, is to listen for the tumblers in the dials to drop into place. In almost all human intellectual endeavors, there are numerous heuristics that guide our problem-solving activity. By contrast, algorithms are rare and tend to be specialized to tasks that are highly mathematical.

The fundamental difficulty with the search paradigm is that, for most interesting problems, the number of possible solutions grows exponentially with the size of the problem. Computer programs for generating solutions (or partial solutions) often are made to combine every element of the problem in all possible ways, as a legal-move generator for a chess-

In the late 1940s U.S. mathematician Norbert Wiener established the science of cybernetics, which is concerned with the common factors of control and communication in living organisms and automatically controlled machines. At a time when the electronic computer was still in its infancy, Wiener and other pioneers in computing foresaw its use for intelligent problem solving.

Massachusetts Institute of Technology

playing program combines all possible opening moves by White with all possible opening responses by Black—for several exchanges into the game. This exponential growth of possibilities is called the combinatorial explosion. It can be controlled by heuristics when sufficiently powerful judgmental knowledge can be found to constrain and focus the search.

Like other disciplines AI encompasses different approaches to research. These are broadly classed as theoretical and experimental. Theoreticians in AI prefer to look at problem solving and representation questions from a formal point of view, often using predicate logic as a lingua franca for discussing methods and results. Experimentalists, on the other hand, prefer to build programs that demonstrate the competence of an idea and then to investigate empirically the strengths and limits of the idea as implemented. Just as AI programs can reason "top down," from a model to concrete instances of it, or "bottom up," from data to generalizations, AI research itself follows these two approaches.

How AI systems represent knowledge

The problem of representation is one of finding an appropriate set of conventions for describing objects, relations, events, and other information. Different programming languages offer different features that make it easier or harder to represent and manipulate facts and relations about the world, and the choice is further complicated by the need to overlay higher level constructs on top of the programming language in order to form a complete AI system. The issue is finding or inventing constructs within a computer language that allow knowledge to be manipulated easily. List-processing languages, discussed below, were invented to facilitate the representation of facts and relations symbolically in linked lists.

It is widely accepted that people have difficulty solving problems when they represent the facts inappropriately. The "trick" for solving algebra word problems, for example, is to map the problem from English into a more suitable representation as a set of equations with correctly conceived variables, constants, and ways of linking them together. Similarly, computer programs must have appropriate constructs to work on.

Formal logic has been investigated as a way to represent knowledge for AI programs because it allows a program to maintain true and consistent beliefs about the world, provided that the starting axioms are true and consistent. Various logical formalisms are widely known and used; most are variations on a formal language known as first-order predicate calculus. For example, "All men are mortal" would be written in the form of a universally quantified statement whose main connective is implication: For all x, man(x) implies is-mortal(x), meaning "Whatever is man is mortal." Standard rules of inference allow an AI program to make deductively valid inferences from starting axioms and previously proved theorems in order to deduce new facts. A program cannot, however, keep all valid conclusions around, nor can it afford the time to make all valid inferences. (This is another instance of the combinatorial explosion.) Instead, once a program has a number of facts about the world represented as logical statements, one can ask about the truth of a new statement by posing the question "Is this statement a theorem?" For example, given the fact that all men are mortal, can one prove as a theorem "Socrates is mortal"? Considerable attention has thus been given to efficient means of proving theorems.

There are many extensions to first-order predicate calculus that make logic a more expressive language. For example, certain techniques allow a program to keep track of what it believes to be true, what might possibly be true, and so forth. These additions, however, substantially complicate the theorem-proving procedures. In recent years specialized programming languages such as PROLOG, which are based on logic, have been created to assist this approach to problem solving. MRS and FOL also use logic as the fundamental framework for representing knowledge.

Production rules, another way to describe facts and relations, were introduced into AI for use in psychological modeling of human problem solving. A production rule is a conditional sentence consisting of an "if" part and a "then" part, also called a condition and an action. The "if" part lists a set of conditions in some logical combination. If the "if" part of the rule is satisfied, the piece of knowledge represented by the production rule is relevant to the line of reasoning being developed. As a consequence, the "then" part can be added to the data base or acted upon.

MYCIN, an AI program developed to diagnose certain infections and select appropriate antibiotic treatment, is an example of a system based on production rules. During a diagnosis MYCIN will request information about a particular set of symptoms, findings, or test results—the "if" part of a rule. If these conditions are satisfied by the responses, the program presents a hypothesis—the "then" part. Further rules are applied until the program reaches the diagnosis that most plausibly explains the findings.

Semantic nets and frames are a third type of representational formalism. They are structures similar to each other that associate definitions of concepts by means of semantic links. A frame is a collection of associated symbolic knowledge about an entity. Typically a frame lists

An example of a production rule from MYCIN, an artificial intelligence program developed to diagnose and select antibiotic treatment for severe infections, is outlined below. If the "if" part of the rule can be satisfied from findings about the patient, the "then" part becomes a relevant piece of knowledge about the infectious agent and the appropriate treatment. The production rule is augmented with a certainty factor, (.7), that indicates the evidential strength of the premise and the importance of the conclusion. (Below right) Another medically oriented AI program, CADUCEUS, whose diagnoses range over the whole field of internal medicine, engages in a consultation session with one of its creators, physician Jack D. Myers of the University of Pittsburgh. CADUCEUS uses a frame-based system to represent knowledge and as of early 1984 contained the largest knowledge base of any expert system, encompassing definitions, symptoms, and diagnostic tests for more than 600 diseases.

IF (1) the infection is primary bacteremia, and
(2) the site of the culture is one of the sterile sites, and
(3) the suspected portal of entry of the organism is the gastrointestinal tract

THEN there is suggestive evidence (.7) that the identity of the organism is bacteroides

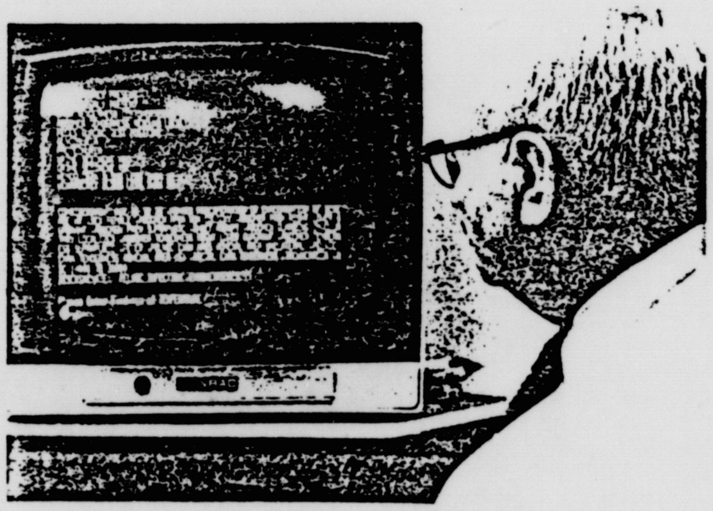
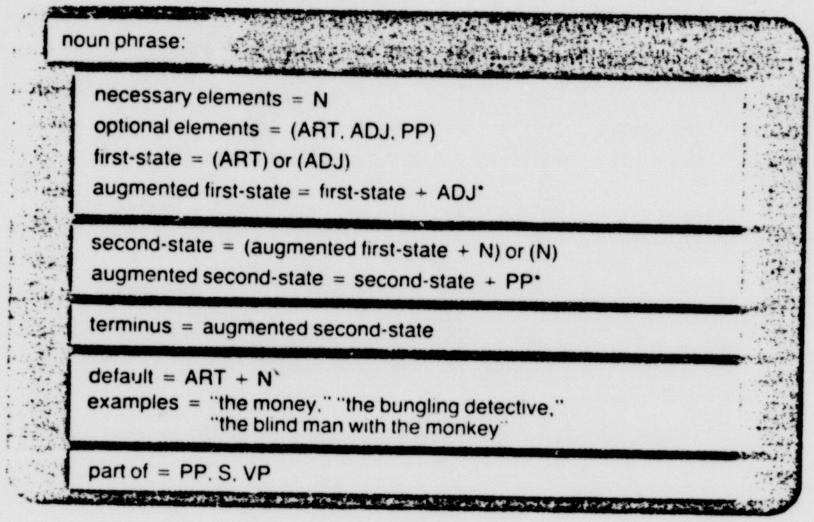


Photo courtesy University of Pittsburgh



noun phrase:

necessary elements = N
optional elements = (ART, ADJ, PP)
first-state = (ART) or (ADJ)
augmented first-state = first-state + ADJ*

second-state = (augmented first-state + N) or (N)
augmented second-state = second-state + PP*

terminus = augmented second-state

default = ART + N
examples = "the money," "the bungling detective," "the blind man with the monkey"

part of = PP, S, VP

key to codes:
N = noun
ART = article
ADJ = adjective
PP = prepositional phrase
S = sentence
VP = verb phrase
* = any number of: *e.g.*, ADJ* means "any number of adjectives"
( ) = alternative element: a parenthetically enclosed element is a member of a group of two or more alternatives

Shown at left is a frame-based representation of a rule of grammar defining a noun phrase. The parts of speech mentioned in this frame are themselves frames, while the entity being represented exists as part of other frames than define a sentence, a prepositional phrase, and a verb phrase. In this kind of system any number of attributes may be attached to the concept being represented.

the properties of the entity, including necessary and optional defining properties and properties that indicate more general and more specific concepts. Because every problem-solving task involves many entities that stand in various relations to each other, the properties can be used to specify those relations. Furthermore, because the properties of an entity are entities themselves that are linked according to those same relations, one frame can give knowledge that is a "special case" of another frame, while some frames can be "part of" another frame. For example, CADUCEUS is a medical diagnosis program that represents knowledge in frames. Each disease and each finding occupies a frame, with obvious two-way links between them and with links within families of diseases. The entire aggregation of frames and linking structure is sometimes called a node-link network, or semantic network, in which the frames are the nodes and the relations are the links.

Controlling the problem-solving process

One way to think of problem-solving behavior is to envision a search through a treelike space of alternatives, called a search space. To explore this space, an artificial intelligence program might be given rules for legal (or sometimes only plausible) moves that tell it what element of the space to consider next. In chess, for example, it is simple to construct rules for legal moves according to the rules of the game. The space of all possible move sequences (complete games) is thus a tree of alternating moves by opposing players. In principle, it is possible to construct an algorithm that explores the consequences of each possible move, that is, all the paths to win, lose, or draw, because the tree is finite. The space is so large, however, that exhaustive searching would not finish in human lifetimes. Thus, ways must be found to control the search so that implausible branches of the tree will not be explored and the most promising branches will be explored first. By introducing heuristics that embody criteria of plausibility and implausibility, one transforms an exhaustive search algorithm into heuristic search.

One well-known heuristic-search program is DENDRAL, which searches a space of graphical descriptions of chemical structures to determine the molecular structure of an unknown chemical compound, given analytical data about it. The number of alternative molecular descriptions that can be constructed from a fixed number of chemical atoms of known types (*e.g.*, $C_8H_{16}O$) grows exponentially with the number of atoms, another example of the combinatorial explosion. DENDRAL successfully uses the same kinds of heuristics that chemists use to limit the number of alternative molecular structures actually considered and thus to avoid searching the space exhaustively. At the heart of the program is a "legal-move" generator that can, in principle, generate all possible alternatives. (This is very different from the ad hoc generation methods used by chemists.) But it can be constrained—using heuristic knowledge to infer the constraints—in order to focus on plausible molecular structures.

For many complex tasks there is no such legal-move generator that is a reasonable starting point for implementing a search program. In understanding speech—transforming acoustic patterns from a spoken English sentence to a sequence of words, for example—AI programs such as HEARSAY-II do not systematically generate successive legal sequences of English words. Instead, they use a variety of cues to trigger the generation of plausible additions to an emerging "best guess" at what the sentence is. While a legal-move generator will always be capable of generating the correct solution if the heuristics allow it to do so, a plausible-move generator may not have a rich enough set of generating principles to include the correct solution in every case. Thus it is a high-level assumption that plausible-move generators will eventually produce the correct solution.

The way a search space is explored depends on the choice of control strategy. The major strategies, all of which have variations, include depth-first search, which explores only the first branch at each level;

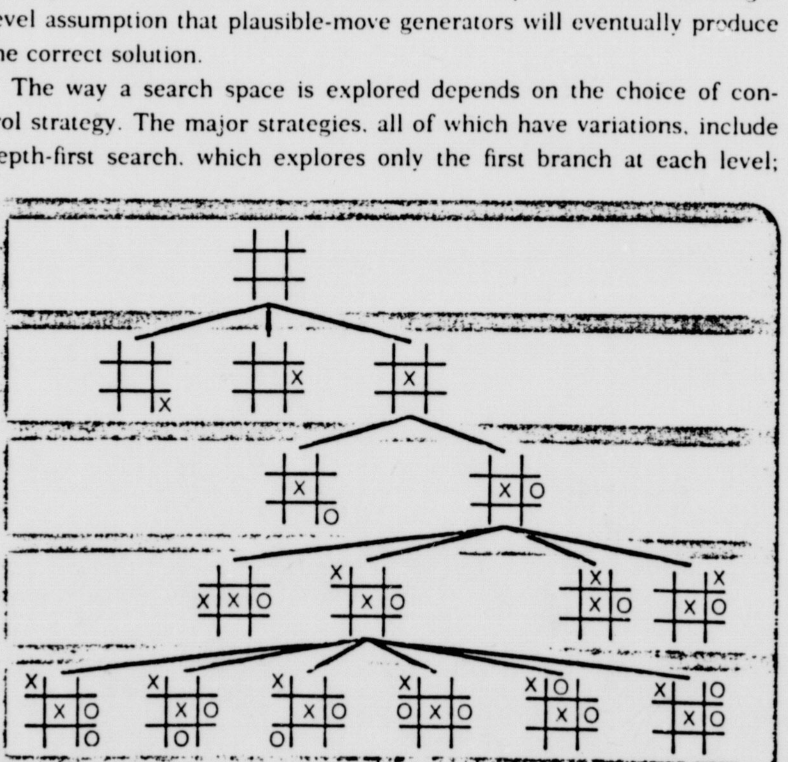
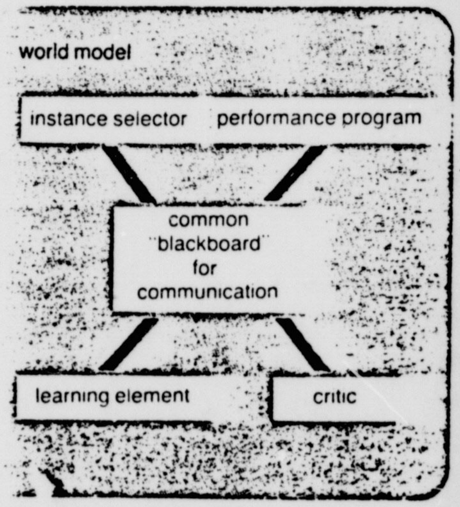


Diagram shows part of the treelike space of alternatives, or search space, for the game of ticktacktoe. Note that the other moves open to the players at each level are symmetric, and thus equivalent, to the ones shown. Only one move at each level is expanded to the next level; if these particular moves were being chosen by a heuristic that judged each of them best among the alternatives, this exploration would be an example of the strategy of best-first search.

breadth-first search, which explores all branches at each level (progressive deepening); and best-first search, which explores at each level only the branch most likely to lead to a solution.

Researchers in the field of AI are beginning to recognize the desirability of endowing their programs with the knowledge that would allow them to choose their own problem-solving strategy. Just as it deals with problem-solving knowledge itself, a program should be able to reason about strategy, change it, and explain it. Knowledge about how to use the contents of a knowledge base is one form of "meta-knowledge," a current topic of research.

A recent development in research on control is dependency-directed backtracking. The cost of backtracking through a search space from a path that no longer appears fruitful can be substantial because the program must redetermine its state of knowledge at a previous branch point in the search tree in order to continue the search along a different branch from that point. In dependency-directed backtracking the program keeps track of the origin of each piece of knowledge in its current state of belief. Then when backtracking is necessary, the program can look at the dependency links to deduce previous states of knowledge without having to regenerate them.



A conceptual view of learning that has emerged from AI research portrays the process as a knowledge-based problem-solving activity. In an AI system this activity can be assigned to a learning program whose task is to modify another program, the performance program, so that its performance improves. In such a system, diagrammed above, the problem-solving vocabulary, assumptions, and procedures are defined for all of the components of the system within a world model. One component, the instance selector, chooses training instances to present to the performance program. Performance is critiqued by the critic, whose advice is implemented by the learning element. These steps are not always separate or all automatic.

Learning as a problem-solving activity

Because a characteristic feature of intelligent living things is their ability to adapt their behavior to a changing environment, considerable work in AI has focused on methods for learning. Learning takes many different forms depending on the type and amount of data available, the nature of the feedback, the knowledge structures that are being learned, and the structure of the performance program that is being improved.

Out of early work on stimulus-response and statistical approaches to learning has come a strong competing concept founded in AI research—that learning is a knowledge-based problem-solving activity. The problem that the learning program is supposed to solve is to modify another program, called the performance program (*e.g.*, a program that plays checkers) so that the performance program improves. With respect to games, improvement is generally measured in win/loss ratios. For other tasks improvement may mean increasing the correctness of predictions or advice, or it may mean decreasing the cost (in speed or memory space) of reaching conclusions.

Learning programs also take different forms depending on the extent to which a person is involved in selecting training instances, providing feedback, giving advice, and integrating new knowledge with old. New knowledge may come from any of several primary sources:

1. Experience: Rote learning is storing results of previously encountered situations (*e.g.*, a board position in a game) without generalizing. Because it saves time whenever the same situation is encountered, a program using saved results can also search further to determine if a better response to the situation can be found.

2. Example: Learning by inductive reasoning can be thought of as

heuristic search through a space of possible knowledge structures (*e.g.*, predictive rules or concept definitions) to find new knowledge that "covers" positive examples presented in the training data and excludes counterexamples. As such, it requires a definition of the search space as well as heuristics that allow it to consider plausible elements in the space efficiently.

3. Discovery: Learning by discovery involves a program's considering many different knowledge structures and assessing heuristically their inherent value. Some heuristics, but not all, may examine how well the new concept applies to specific cases (as in induction) whereas others may examine how the concept or rule was derived in order to assess its value.

4. Advice: Learning by taking advice can range from a passive set of text-editing instructions that can accept and integrate new knowledge to an active program that assists in defining new knowledge and keeping the total knowledge base consistent.

5. Experts: Human experts may be the source of new knowledge, and the transfer of their expertise to a program generally is the duty of a programmer, in this context called a knowledge engineer, who helps structure new knowledge appropriately for a program. This activity has come to be known as knowledge engineering and is the primary method for building expert systems, discussed below.

6. Analogy: Learning by analogy has been successful in a few prototype programs. It requires rich knowledge structures for two or more areas and requires solving two difficult problems: finding relevant and useful analogies among all possible correspondences between areas of knowledge and using the correspondences in plausible ways (and only plausible ways) to discover new knowledge.

7. Watching: Learning may come from observing a person solve problems or perform a task correctly. Prototype programs exist that first examine the instructions that an expert gives to a computer in order to accomplish a task and then generalize those instructions to cover similar tasks.

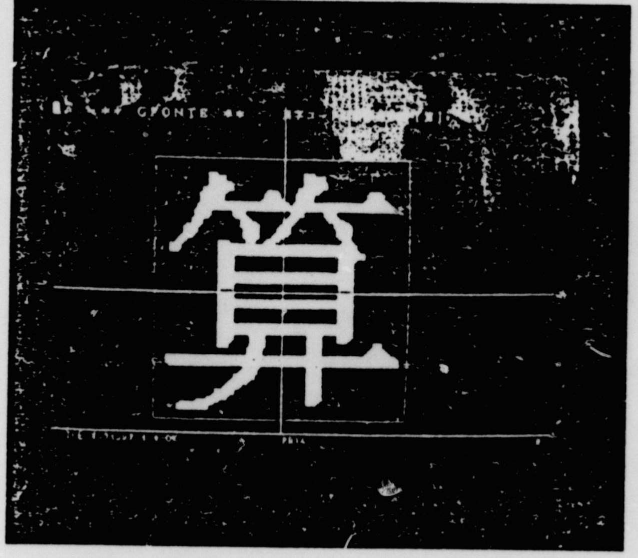
8. Text: Learning by reading text requires broad, powerful programs that understand natural language.

Languages and AI systems

Because AI works more with symbolic information than numeric information, languages designed to support symbolic processing have been central. The two most influential languages have been IPL-V and LISP, and most AI work is presently done in various dialects of LISP. One of LISP's key advantages is the ease with which it can use programs as data in order to edit or explain parts of the program and can treat data structures as programs in order to execute newly created or edited parts of the program.

A recent development in AI has been the construction of personal computers specifically designed to use LISP efficiently. Many of these, often called LISP machines, are now commercially available. They have large main memories and disk storage capacities and come with high-

Photos Milton and Joan Mann—Cameramann International



Display of a kanji (character) conveying meanings of "number" and "calculation" (above left) is drawn from a massive computer dictionary, part of a natural-language system being developed for Japan's fifth-generation computer. At the Japanese government's Institute for New Generation Computer Technology (above) researchers conceive software for the advanced machine, in which AI approaches to problem solving and inference will form the core of the processing functions.

resolution visual displays and some pointing device for identifying and manipulating displayed information. Local communications networks allow these personal workstations to be connected together and to communicate with large central computers and their facilities.

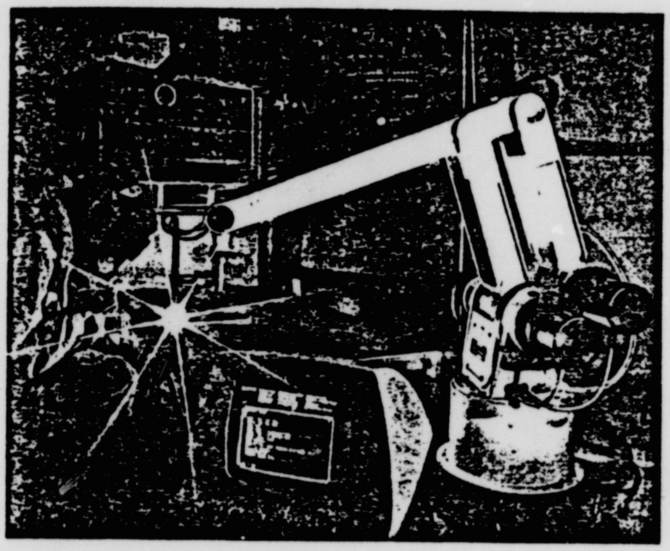
In the early 1980s the Japanese industrial and research communities announced their intention to design and build a new generation of computer specifically for symbolic reasoning, the fifth generation in the evolution of computers based on technological advances. In the Japanese design problem solving and inference will be the core of the processing functions. Power is measured in logical inferences per second (LIPS), where one LIPS is one such inference per second. Currently, one inference operation is estimated to require 100–1,000 steps; thus one LIPS is equivalent to 100–1,000 instructions per second. Machines of the present generation are said to work at 10,000–100,000 LIPS. The final target for fifth-generation machines is performance of about a billion LIPS, four or five orders of magnitude faster than current machines.

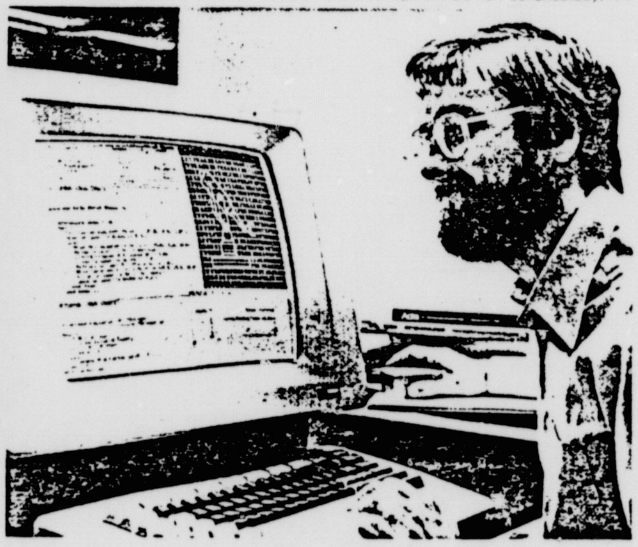
A natural-language understanding system, which will be an integral part of successful AI programs, will require large dictionaries of words and phrases. In the design for the Japanese fifth-generation computer, the design requirement for the natural-language system is a vocabulary of 100,000 words, each requiring hundreds to thousands of computer words of storage.

Applications

Two early AI systems, DENDRAL and MACSYMA, demonstrated that AI methods could be applied to practical problems; namely, identifying the molecular structure of chemical compounds and simplifying complex mathematical expressions. Other experimental researchers in AI selected similarly interesting problems as vehicles for investigating AI issues. Still others have recently begun commercial applications of AI methods to industrial problems.

Photos General Electric Research and Development Center, Schenectady, N Y





Robot welder (above) is basically a large computer-controlled reprogrammable mechanical arm. Whereas common industrial robots depend on numerical algorithms for their control, AI methods have begun to be used to program complex sequences of robotic movements. (Above right) Using a LISP machine, i.e., a computer designed to use the AI language LISP, a researcher develops programming for a robot-based factory workstation.

(Facing page, top) A major emphasis of AI work in robotics has been the development of programs that will allow computers to understand sensory data, particularly visual images. Recognition of an aerial view of a tank, for example (lower left), begins with the conversion of its video image to a digital form (upper left) made up of thousands of dots (pixels), representing light-intensity values, that are stored as arrays of numbers in the computer. "Intelligent" manipulation of these numbers can identify image areas containing sharp changes of intensity, which correspond to the edges of the object (upper right). Once the computer isolates the edge outline from the background (lower right), it then can apply a more refined degree of AI to find distinctive features, such as the gun barrel, that will help its recognition of the object as a tank.

Photos Lockheed Electronics Company Inc

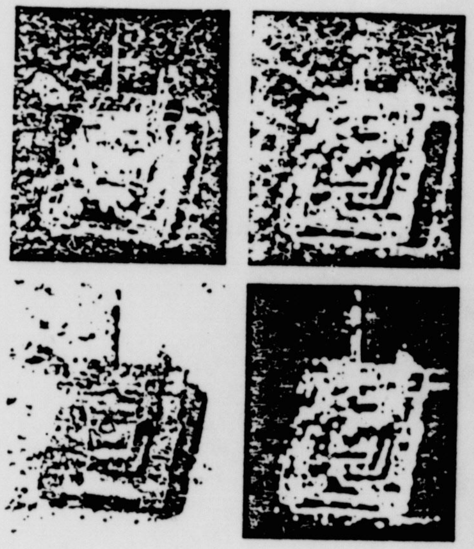
Industrial applications fall roughly into four classes: robotics, natural language, expert systems, and tools for developing software. One of the important developmental milestones for AI has been the commercialization of AI tools and methods in these four areas. Many industrial research laboratories have established groups to work on applications of AI, and new companies have been formed to provide AI tools and customized software to companies in nearly every segment of the economy.

Building intelligent robots has been an ambitious goal of researchers for centuries. Within AI this work has centered on computer-controlled mechanical manipulators and vehicles and on computer understanding of sensor data, mostly in the form of television signals. AI efforts in problem solving, speech recognition, and other general topics are also clearly relevant to building intelligent robots although not specialized to it.

Industrial computer-controlled manipulators are currently used in thousands of manufacturing operations, such as spray painting and welding. Most use little AI but are based on engineering control algorithms. AI-based languages for describing the movements of a manipulator have begun to be used for programming complex operations.

Without feedback from sensors, a manipulator must rely on precise alignment of parts, on correct positioning and orientation—in short, on everything being set up ahead of time in a foolproof, blind operation. In dynamically changing situations, this requirement is impossible. Therefore, an intelligent robot must be able to sense (see, feel, hear, or otherwise determine) what is in its environment. Apart from robots, understanding data from various sensors (*e.g.*, television, X-ray, sonar, infrared, or ultrasound) is also important in contexts ranging from medical diagnosis to military intelligence and surveillance activities.

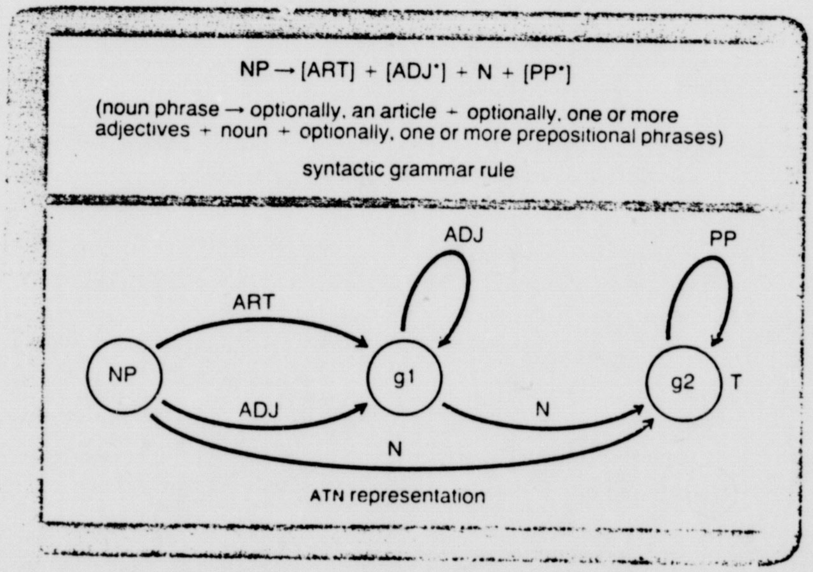
Work on computer vision is carried out at several different levels. At the low end, closer to the data-collection process are problems of digitizing, edge detection, line finding, region segmentation, and texture analysis. At the high end, closer to the interpretation process, are problems of recognizing meaningful shapes (*e.g.*, a house), and determining relationships among objects (*e.g.*, one object supporting another). Especially difficult problems include representing and recognizing three-dimensional objects and dealing with color, motion, and stereo images. (See *1983 Yearbook of Science and the Future* Feature Article: Robots "Man" the Assembly Line.)



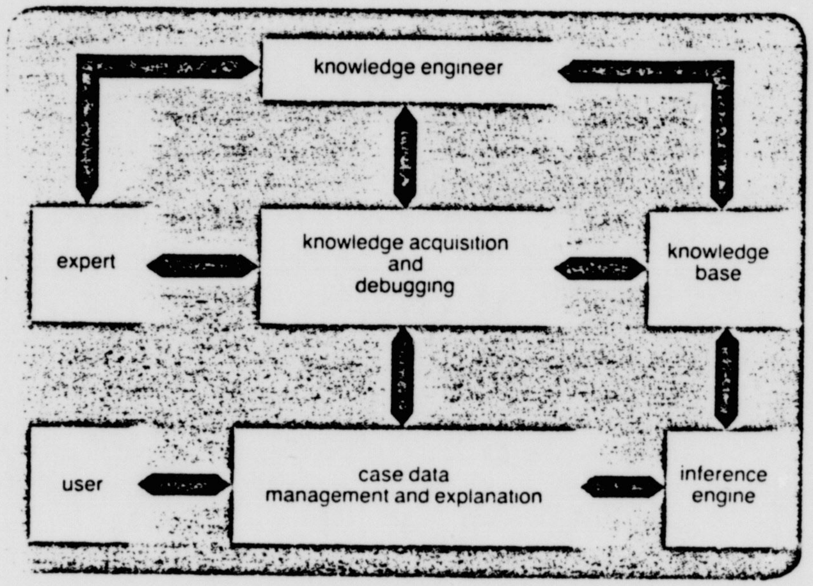
Human-computer interactions have always presented interesting and difficult problems and are becoming increasingly important as software systems become more complex and powerful. From the early days of artificial intelligence, researchers have worked on programs that are intelligent enough to carry on dialogues with people in English or another natural language. One grand goal, the focus of much early attention, was computer translation of text from one language into another, a task that requires both understanding and generating natural language.

Purely syntactic approaches to understanding language were shown to result in too many ambiguities and not enough understanding. A syntactic grammar is a general grammar for the language; it deals with the way words are strung together into larger units and may embody such rules as "a noun phrase is either a determiner [a word such as "his" in "his new idea"] followed by a noun or a determiner followed by an adjective followed by a noun." Semantic grammars specialize these rules to take account of specific meanings of words in particular contexts.

Augmented transition nets were a significant development as a convenient representation for grammars. They link different syntactic and semantic elements together in the form of labeled graphs to represent alternative ways of expressing meaningful sentences or parts of speech. Case frames are another important way to represent the information in sentences and groups of sentences. In this method framelike structures, each with predefined slots (some required, some optional, some forbidden in special cases) for various properties of the subject of the frame, define semantically meaningful cases.


NP → [ART] + [ADJ*] + N + [PP*]
(noun phrase → optionally, an article + optionally, one or more adjectives + noun + optionally, one or more prepositional phrases)
syntactic grammar rule
ADJ
PP
ART
NP
g1
g2
T
ADJ
N
N
ATN representation

A syntactic grammar rule for a noun phrase is compared with its augmented transition network (ATN) representation. In each case the noun phrase is defined as either (1) an article or an adjective, followed optionally by any number of adjectives, followed by a noun, followed optionally by any number of prepositional phrases, or (2) a simple noun. In the ATN representation, g1 and g2 are intermediate grammatical constructs, and T (terminus) indicates the point at which the representation is defined to be complete. ATN's, which link different syntactic and semantic elements in the form of labeled graphs, have proved to be a useful technique for representing grammar in AI research.


knowledge engineer
expert
knowledge acquisition and debugging
knowledge base
user
case data management and explanation
inference engine

The conceptual structure of an expert system including the elements necessary to build it is illustrated at the right. Separation of the knowledge base and the inference engine (the interpreter of the knowledge base plus auxiliary procedures for keeping track of inferences) is a central idea.

Considerable progress has been made in understanding text within restricted contexts, such as a conversation with a restaurant waiter, a newspaper story about terrorist attacks, or specification of computer programs. Good results have been obtained with machine translation of technical information as well. Commercial applications include systems that understand queries about large, specialized computer data bases, and can thus translate an English request for information into a query phrased efficiently in a formal computer language. As with other AI problem-solving programs, programs for understanding natural language have difficulty representing and reasoning with common-sense knowledge. Dealing with metaphors and contextual cues is especially hard in understanding natural language.

DELTA, an expert system developed for locating faults in diesel-electric locomotives and advising on their repair, is demonstrated by computer scientist Frank Lynch of General Electric Co. (seated). Originally set up in LISP, the system is based on more than 500 "if-then" production rules derived from interviews with Dave Smith (standing), a GE field engineer with four decades of experience in locomotive repair. DELTA also displays charts, drawings, and training-film sequences on videodisc to assist the user in locating components and carrying out repairs.

Expert systems are AI programs that embody considerable human expertise to solve important problems. They are flexible in the sense that they can be changed and extended easily, and they are understandable in the sense that they can explain the contents of their own knowledge bases and their own lines of reasoning. In the early 1980s expert systems captured the interest of the commercial world. Both new and established companies have begun building expert systems for commercial purposes.

Applications of expert systems range from medicine to electronics and from machinery to computer software; their tasks range from diagnosis and troubleshooting (analysis) to planning and configuration (synthesis). Some examples include programs that diagnose disease (*e.g.*, MYCIN), search for ore deposits, help solve oil-well drilling problems, troubleshoot locomotive breakdowns, and configure computer systems. Currently the major steps in building expert systems are selecting an appropriate problem (in terms of such attributes as size, difficulty, importance, and risk), selecting a representation and control structure (or framework system that supplies both), settling on an appropriate vocabulary and conceptualization for the problem, finding an available expert, transferring the

General Electric Research and Development Center
Schenectady, N.Y.

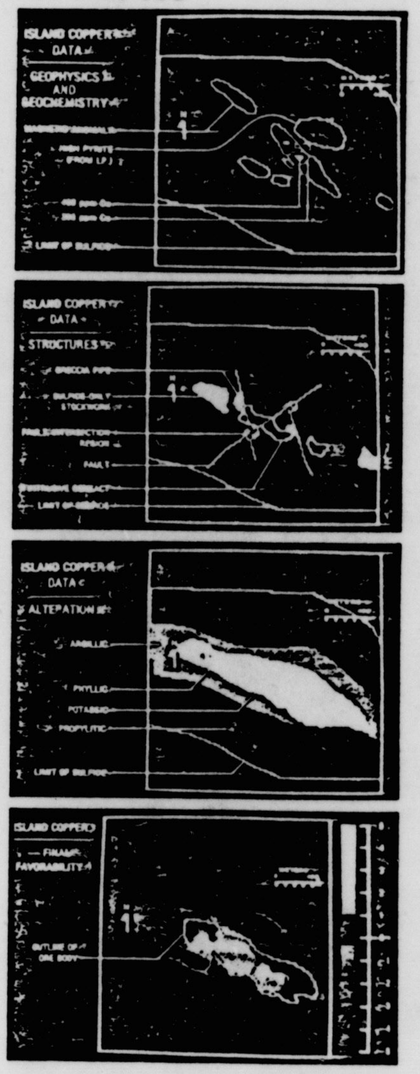
expert's knowledge into the program (knowledge engineering), refining the knowledge base with feedback from test cases, packaging the system in a form that is acceptable to users, and validating the quality of the program's advice.

Probably the most successful commercial expert system so far is the XCON system developed at Digital Equipment Corp. Its task is to configure orders for the company's VAX computer systems to meet customers' specifications on such things as speed and memory size while keeping track of additional considerations such as cooling requirements and lengths of cables. After five years of development the system contains more than 3,000 rules and has been used to configure more than 80,000 orders.

Developing complex AI programs is becoming ever more dependent on programming-support tools; *i.e.*, other programs, particularly ones based on LISP. The dominant trend is for smarter tools that take over more and more of the routine programming chores. According to the manual for one of these support systems, the INTERLISP Programmers' Apprentice, it is designed to "cooperate with the user in the development of his programs, and free him or her to concentrate more fully on the conceptual difficulties and creative aspects of the problem he is trying to solve." Present capabilities at the apprentice level include spelling correction, remembering commands and contexts, and bookkeeping.

Automatic programming techniques are currently powerful enough to construct simple programs from abstract specifications. These techniques are a logical extension of the "Do what I mean" concept in programming, which attempts to reduce the amount of detail and precision necessary in framing commands for them to be accurately understood. Future designers and implementers of large AI systems will be able to turn over substantial coding jobs to automated apprentices, thus freeing them to work more on design and less on implementation. Current research projects such as the Programmers' Apprentice, PIE system, and SAFE are pointing the way toward tools for producing efficient programs from specifications, plans, or very high-level programming languages.

Photos SRI International



The expert system PROSPECTOR, largely a rule-based system, incorporates the expertise of several geologists in mineral exploitation and resource evaluation in order to predict ore deposit locations from existing geologic data. Of the PROSPECTOR-generated maps above, the top three portray the geology of a region of Vancouver Island, British Columbia, as synthesized from information that the system requested of the user in evaluating the likelihood of a copper deposit. The bottom map shows PROSPECTOR's prediction of the location of the deposit (red, high probability; blue, low probability) as well as an outline of the actual ore body.

FOR ADDITIONAL READING

Avron Barr, Paul Cohen, and Edward Feigenbaum (eds.), *The Handbook of Artificial Intelligence*, 3 vol. (William Kaufmann, Inc., 1981 and 1982).

Bruce G. Buchanan and Edward H. Shortliffe, *Rule-Based Expert Systems* (Addison-Wesley, 1984).

Frederick Hayes-Roth, Donald Waterman, and Douglas Lenat (eds.), *Building Expert Systems* (Addison-Wesley, 1983).

Pamela McCorduck, *Machines Who Think* (W. H. Freeman and Co., 1979).

Patrick H. Winston, *Artificial Intelligence*, 2nd ed. (Addison-Wesley, 1984).

Periodicals focusing specifically on artificial intelligence include *The AI Magazine*, *Artificial Intelligence*, *Cognitive Science*, and the *Journal of Automated Reasoning*.