NASA TM-83216

NASA Technical Memorandum 83216

NASA-TM-83216 19830020606

DECISION-MAKING AND PROBLEM SOLVING METHODS IN AUTOMATION TECHNOLOGY

W. W. Hankins, J. E. Pennington, and L. K. Barker

May 1983

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

TABLE OF CONTENTS

	Page
SUMMARY	1
INTRODUCTION	2
DETECTION AND RECOGNITION	3
Types of Sensors	
Computer Vision	
CONSIGHT-1 System	
SRI Vision Module	
Touch and Contact Sensors	
PLANNING AND SCHEDULING	7
Operations Research Methods	
Linear Systems	
Nonlinear Systems	
Network Models	
Queueing Theory	
Dynamic Programming	
Simulation	
Artificial Intelligence (AI) Methods	
Nets of Action Hierarchies (NOAH)	
System Monitoring	
Resource Allocation	
Robotics	
LEARNING	13
Expert Systems	
Learning Concept and Linguistic Constructions Through English Dialogue	
Natural Language Understanding	
Robot Learning	
The Computer as an Intelligent Partner	
Learning Control	
THEOREM PROVING	15
Logical Inference	
Disproof by Counterexample	
Proof by Contradiction	
Resolution Principle	
Heuristic Methods	
Nonstandard Techniques	
Mathematical Induction	
Analogies	
Question-Answering Systems	
Man-Machine Systems	
Other Comments	
DISTRIBUTED SYSTEMS	20
Decentralized Control	
Distributed Computer Systems	
Hierarchical Systems	

KNOWLEDGE BASES 22
Relational Data Bases
Frames
Common Sense Algorithms and the Bypassable Casual Network
Knowledge Representation Language (KRL)
Expert Systems
Hierarchically Ordered Knowledge Bases
SEARCH 28
Graph Searches
Breadth First Algorithm
Uniform Cost Search
Ordered Search
Depth First Search
Heuristic Search
Minimax Search
N Step Look Ahead
Alpha-beta Procedure
Means-end Analysis
Common Sense Algorithm
Dynamic Programming
Control Formulation of Artificial Intelligence Problems
HEURISTICS 30
EVOLUTIONARY PROGRAMMING 33
CONCLUDING REMARKS 34
REFERENCES 35

DECISION-MAKING AND PROBLEM-SOLVING METHODS IN AUTOMATION TECHNOLOGY

By

Walter W. Hankins, Jack E. Pennington, and L. Keith Barker
Langley Research Center

SUMMARY

It is a never-ending process to automate to a higher level than that which presently exists. Current thinking on automation is to design machines to have more decision-making and problem-solving capability, especially in tasks for which they have not been explicitly programmed. In this respect, technology is needed to allow machines to use "human-like" reasoning in their operations. This will free man from routine mental operations so that he can dedicate his full potential to other unsolved problems (in which the reasoning machines may be of further assistance). This technology will enhance man's control over large and complex problems and will aid him in making quick, real-time decisions which are mission critical.

The state-of-the-art in decision making and problem solving is reflected in the problems being solved; that is, the pertinent concepts are problem specific. At this point, no taxonomy of generic concepts exists; therefore, this report presents brief synopses on some major topics related to decision making and problem solving. These topics are: (1) detection and recognition; (2) planning and scheduling; (3) learning; (4) theorem proving; (5) distributed systems; (6) knowledge bases; (7) search; (8) heuristics; and (9) evolutionary programming.

For space missions, computers represent a tool which can be directed by man to achieve greater goals. In many cases (for example in the first Shuttle flight into Earth orbit and back), the response times for certain actions are too short to be handled by man and, therefore, must be accomplished by computer directed machines. As computers are given more reasoning ability, they will be of greater assistance in problem solving. There are computer programs which search over databases, logically putting facts together, in formulating a response to an interrogator's questions. If its program is of a medical nature, the computer may ask the interrogator to measure another of the patient's functions in order to rule out or converge on a prognosis.

For advanced automation by computer, basic questions arise, such as: How should knowledge be structured in a computer? How can information be accessed promptly and efficiently? What is the hierarchical structure for machine decision making? How can a machine learn? What are generic concepts associated with automated decision making and problem solving? How does man go about solving a problem? What is the best man-machine interface? How can a machine generate its own goals?

This paper is not a compendium on automated decision making and problem solving, but is rather a presentation of some of the interesting topics being examined in this exciting and difficult area.

INTRODUCTION

In the last 20 years, with the rapid advances in electronics and computer science, it has become possible to automate, partially or completely, many manual tasks. Industrial automation has taken over many repetitious tasks, freeing the worker for more challenging tasks (refs 1 to 6). Automation promises to improve the efficiency and productivity of man-intensive activities. Also, large-scale, small-size, high-speed computing capability makes it possible to attack very large, complex problems which otherwise would be beyond man's capabilities to solve in a reasonable manner and time. Along with the rapid increase in computer capabilities has been a growth in the discipline of "Artificial Intelligence" which involves enabling a computer to perform tasks, (including decision-making and problem solving) which, if performed by a human would be considered intelligent (refs 7 to 9).

NASA's automation program is directed towards supporting research to develop technology applicable to future missions requiring advanced automation. As part of this program, the authors were tasked to determine the "state of the art" in automated problem solving and decision making. The goal was to develop a taxonomy of methods or techniques, including the problems to which they are applicable, and their availability and maturity. Such scientific classifications of the field of automation is complicated by several factors:

1. Several different disciplines, including Artificial Intelligence, Operations Research, and Controls Theory, are involved in work related to problem solving and decision making.

2. New or improved methods are constantly being developed and applied. The volume of reports and articles related to problem solving and decision making could be overwhelming.

3. Publications often lack sufficient detail on methods employed.

4. Many current problems are so complex that they need to be attacked by multi-disciplinary application of several techniques.

Thus, this report is more realistically scoped to discuss decision-making and problem-solving methods which are related to automation technology. The intent is to discuss specific problems or programs as well as the applicable methods, with the assumption that the state of the art in problem solving is reflected in the problems being solved.

Sources of information included a survey of available literature, visits to universities, visits to government agencies performing and funding related work, and results of a conference on "Automated Decision Making and Problem Solving" (ref 10) held at Langley in May 1980. As a result of the survey and review, problem-solving and decision-making methods have been identified in the following areas:

1. Detection and Recognition

2. Planning and Scheduling

3. Learning

4. Theorem Proving

5. Distributed Systems

6. Knowledge Bases

7. Search

8. Heuristics

9. Evolutionary Programming

These subject areas are broad enough so that many problem-solving methods are included, but narrow enough to be useful to those interested in an overview of automation.

DETECTION AND RECOGNITION

In order to learn, adapt, or accomplish tasks in a changing environment, a robot must (1) have sensors to obtain information about its environment; and (2) be able to process this information in a meaningful way.

Types of Sensors

There are numerous types of sensors, for example: (See ref 11)

1. Light sensors such as photodiodes, phototransistors, or photocells for computer vision or proximity detection

2. Strain gages in which the electrical resistance changes with stress to indicate, for example, pressure or force exerted

3. Piezoelectric crystals which sense pressure on themselves by changes in frequency

4. Capacitors for pressure feedback

5. Light or radio beams for "homing in" on target

6. Wheel rotations or conveyer belts for distance or location measurement

7. Vidicon cameras, and charge coupled devices (CCD) for sensing shades of gray for computer vision

8. Gyroscopes for measuring orientations and accelorometers for accelerations

9. Color sensors for vision

10. Voice recognition systems for receiving commands

11. Fiber optics for proximity sensors in the robot's hand

12. Infrared radiation sensors for heat detection

13. Potentiometers and optical encoders to measure angular orientation

Reference 12 is a tutorial overview of sensors in manipulation control. In general, industrial robots represent "hard automation"; that is, they are pre-programmed to follow a pattern of sequenced operations. These robots are extremely useful; however, sensors are needed to provide the robot with the capability of adapting to its environment. One of the most important sensors available for a robot to perceive its environment is vision.

As a mathematician would say, having a sensor is necessary but not sufficient to have an intelligent robot. Just having sensor information is not enough. The robot must be able to process this information, draw conclusions about the environment, and deduce corresponding actions to be taken to accomplish some task or purpose. For example, a TV camera pointed at an object conceptually provides a checkerboard type of image in which the blocks are various shades of gray. The robot's task is to uncover the object from these shades of gray.

Computer Vision

The amount of literature on the various aspects of computer vision is quite extensive. For example, reference 13 is the ninth in a series of bibliographies on computer processing of pictorial information, covering primarily items published during 1978, and lists over 800 references alone. Hence, only a few highlights of current research on computer vision are practical for presentation in this brief synopsis.

To have a robot view the three-dimensional world as man views it is, at present, only a dream; however, much headway has been made in this direction and current research efforts are attacking this problem. In 1977, a survey (ref 14) was published on current techniques for computer vision with particular emphasis on the description, analysis, and reconstruction of scenes from single or multiple views. In 1974, a Russian survey article (ref 15) was published on visual information processing by robots. Current status reports appear in references 16 and 7, for example. Commercially, the state-of-the-art in computer vision with respect to manipulator systems is represented by systems such as CONSIGHT - 1, which is a General Motors' vision-controlled robot system for transferring parts from conveyer belts (ref 18) and the SRI Vision Module (ref 17).

CONSIGHT - 1 system - In this system a light source and cylindrical lens are used to project a narrow and intense line of light across a conveyer belt. This line is viewed from above by a linear array camera. As an object on the conveyer belt crosses this line of light, the part of the line which hits the object is deflected from the remaining parts of the line which are not intersected. The deflected part of the line is not visible to the camera. This break in the light is used to obtain an outline of the object for recognition and orientation.

SRI Vision Module - This system uses a solid state TV camera to view an object. A light source and the camera are angled for illumination of the object and image reception, or the camera views the object placed on a back-lighted surface to highlight the object's silhouette. Dark image areas in the latter case correspond to the object; whereas, light image points correspond to the background. Thresholding is used to classify all areas as either black or white. Dark areas are treated as "blobs" and numerous identifying characteristics are computed from this data, such as centroid, perimeter length, center of bounding rectangle, number of holes, moments of

inertia, and angle of the maximum and minimum radius vector from the center of gravity to points on the perimeter. The various descriptors are used to identify the object as the one stored in memory which gives the "closest" fit to the unknown object.

A computer vision system which can operate in the three-dimensional world is required for general applications, such as space construction. "Almost all the manipulator systems in present use in industry and also those under development extract the required information about object recognition from the analysis of two dimensional pictures (ref 19)."

A stereo pair of TV cameras has been used to extract three dimensional measurements (refs 20 and 21). Correlation is one of the basic drawbacks of the stereo system, that is, identifying corresponding points in the two images. This operation requires complex and lengthy computations which hinders real-time operation. After the correlation process, the object must still be identified by some pattern recognition method. Another problem is missing data because some points can be seen by only one camera.

Shirai and Suwa (ref 22) developed the idea of using a single camera and a plane of light to recognize polyhedrons in three-dimensional space. A series of light lines are cast across an object. By changes in slope and displacements, these light lines indicate the planes of the polyhedra. To reduce processing time only the end points of lines are determined. Then, lines are classified into planes. Thereafter, the three-dimensional position of each plane is calculated and the object is recognized by the relationship between the planes. This type of system has been investigated by others (refs 23 and 24) and has been used in a real-time robot arm experiment at the National Bureau of Standards (G.J. Vanderburg, J.S. Albus, and E. Barkmeyer) to acquire an object randomly placed on a table.

Agin and Binford (ref 23) consider more complex objects which are considered to be a composite of a number of cylinders. It is difficult to determine the relationships of parts of an object to each other.

Will and Pennington (ref 25) describe some experiments in which a rectangular type of grid is projected toward an object. By the distortions in the grid that falls upon the object, planer areas are extracted so that the object can be identified.

In general it appears that some type of parallel processing will be needed to speed up these methods for real-time operation.

Work at the Rensselaer Polytechnic Institute has dealt with computer vision by reconstructing the surface of an object from noisy measurements obtained from a laser ranging system. Range measurements are used to compute surface gradients. A recursive algorithm for smoothing data and reducing the computational complexity has recently been developed.

Marking parts to aid robot vision has been considered in reference 26. This work is only preliminary and further analysis is needed to determine under what conditions the procedure is feasible.

It has been suggested that the solutions to the difficult problem of computer vision may reside in hardware rather than software, or a combination of both (ref

16). Furthermore, rather than repeatedly reconstructing a visual scene, it appears more economical to perform a detailed analysis of a visual pattern only when it has changed in some important way from the previously seen pattern (ref 27). In other words, examine only that portion of the scene which is changing.

Humans have the ability to infer three-dimensional surface structure from a two-dimensional line drawing. Yet, there is an infinitude of space curves which can be projected onto the same image boundary. A conjecture (supported by psychological experiments at MIT) is that humans perceive that curve having the most uniform curvature and least torsion (planar) consistent with the boundary conditions. To capitalize on this, investigators at SRI International are considering a cost function which, when minimized, tends to produce this type of interpretation. A computer model is being developed to interpret two-dimensional line drawings (assumed to be obtained from gray-level imagery) as three-dimensional space curves and then to compute the three-dimensional surfaces bounded by these space curves (refs 28 and 29), where the final interpretation should correspond to that inferred by humans.

Touch and Contact Sensors

There are numerous types of touch and contact sensors. A few examples are mentioned in this section. Examples of proximity sensors, electro-optical imaging sensors, and range-imaging sensors that are used in robot systems are presented in reference 30.

The logic strategy associated with a sensor varies with the task to be accomplished. At the Naval Research Lab, the use of a touch sensor to essentially "feel" the three-dimensional shape of an object for recognition has been investigated in connection with an underwater robot.

An intelligent industrial arm, designed and built at the University of Virginia, used four point sensors located on the inner surfaces of each of two jaw-like fingers to indicate any slippage of a grasped object (ref 30). The hand, which could exert 11 levels of force, tightened when slippage was detected. Very quickly responding needles (slippage sensors) were thrust forward to make contact with the object grasped by the hand. If the object slipped, causing one of the needles to be deflected, then a signal was sent to the hand to tighten its grip. Meanwhile, the deflected needle quickly withdrew and was thrust forward again in the undeflected position. This process was automatically repeated until the grasp was sufficient to disallow slippage of the object from the hand.

As noted in reference 6, the functions of contact sensors in controlling manipulation may be classified into searching, recognition, grasping, and moving. Furthermore, there are three basic methods for sensing forces and torques for controlling a manipulator. In the first method, the force and torque on each manipulator joint can be obtained in a direct manner. In the case of an electrically driven joint, these parameters are proportional to the armature current. Force sensing has been used in inserting a peg into a hole and in assembly tasks. In the second method, the three components of force or torque between the hand and the terminal link of a manipulator can be measured by a wrist force sensor. A typical strain gage wrist force and torque sensor is illustrated in reference 28, along with other examples. In the third method, a pedestal force sensor measures the three components of force and torque applied to a workpiece mounted on a pedestal.

Reference 12 describes joint torque sensing and wrist force sensing as a means of detecting contact force at a manipulator's hand. A device is described which "... performs chambered pin-hole insertions by rocking the pin back and forth within the geometrically determined wobble angle range, detecting when the limit of rocking has been reached by means of limit switches that detect compression of springs in the device's wrist." In another example, torques in an arm's shoulder joint are used to sense contact in the process of "poking" for a hole. Sometimes sensitive elements attached to a structure at various points (for example, strain gages) detect deformations and are used to obtain a vector output.

Touch sensors, which can be mounted on the outer and inner surfaces of each finger of a manipulator hand to indicate contact between the fingers and objects in the workspace, are described in reference 12. The outer sensors can be used (1) to avoid damaging movements caused by forcing the mechanical hand against an obstacle in its search for an object, (2) to possibly identify the object, and (3) to determine the position and orientation of the object. The inner sensors provide information about an object before it is grasped and also slippage information when it is grasped. A combination of computer vision and "feel" sensors allows the manipulator to handle more complex tasks.

PLANNING AND SCHEDULING

In general, planning can be defined as a process of preparing and selecting a set of actions or decisions, from alternative options, in order to achieve a goal. Methods used to develop a plan will depend upon the nature of the problem, namely:

1. Well-defined or ill-structured
2. The goal and possible subgoals
3. Number of independent variables
4. Qualitative and/or quantitative variables
5. Constraints
6. Uncertainties

A schedule is a plan in which time is an independent variable. Planning or scheduling is closely related to search. Search can be described as the exploration of a tree of possible action sequences with the goal of finding a path between an initial and goal state. A number of search methods are discussed in a separate section. Scheduling problems are often studied in operations research, but recently AI techniques have been applied to planning and scheduling problems. The following sections explore some of the work in both areas.

Operations Research Methods

Operations Research involves the construction of mathematical descriptions of economic and statistical models of decision and control problems to treat situations of complexity and uncertainty. It also involves analyzing the relationships that determine the probable future consequences of decision choices, and devising

appropriate measures of effectiveness in order to evaluate the relative merit of alternative actions. Operations research offers many tools including linear and nonlinear optimization, dynamic programming, queueing theory, combinatorial theory, network theory, and scheduling theory.

Linear Systems

Linear optimization is concerned with finding an extremal of a linear performance index

$$Z = C1*X1 + C2*X2 + \ldots + Cn*Xn \text{ subject to constraints}$$

$$A11*X1 + A12*X2 + \ldots + A1n*Xn \leq B1$$

$$\vdots$$

$$Am1*X1 + Am2*X2 + \ldots + Amn*Xn \leq Bm$$

Such problems occur in product-mix or blending schedules, resource scheduling and allocation, transportation, distribution, and production levels (ref 30). Techniques for solving linear optimization problems include the Simplex method , networks and decision trees, sensitivity analysis, integer programming, decomposition, and duality. Markov chains can be used to model a physical or economic process when the set of possible outcomes (states) is finite, and the probabilities of state transition are constant over time. In problems consisting of discrete outcomes which depend on a prior result, Markov chains can be used in both modelling and analysis.

Nonlinear Systems

Nonlinear optimization is concerned with finding an extremal of a performance index $Z = F(X1,X2, \ldots,Xn)$ subject to constraints, such as $Ai(X1,X2,\ldots,Xn) \leq Bi$ for $i=1,2,\ldots,n$ where F and Ai may be nonlinear functions.

Such problems arise in real-time operations, when a specific cost (objective) function to be maximized cannot be defined, or when a linear model approximation is not accurate enough. Techniques or methods for solving nonlinear optimization problems include steepest ascent (or descent), quadratic programming, Newton-Raphson method, modified gradient techniques, subrelaxation, and dynamic programming. Bellman (refs 33 and 34) notes that dynamic programming is actually the study of multistage decision processes. This interpretation yields an approach which is useful for both analysis and computational purposes.

Network Models

Network modelling techniques are used in many problems, including transportation, resource allocation (assignment), and manpower planning. An example is a transportation network in which modes represent sources (factories), sinks (markets), or transhipment point (warehouses) and; arcs represent shipping routes subject to constraints such as availability, cost, or capacity. These problems are among the easiest, since they are polynomially bounded; that is, in the worst case, the maximum number of steps required to solve such a problem can be constrained by a polynomial function of the amount of input data needed (ref 35). Despite an upper bound on the number of steps, the number can be, and usually is, very large. Even with the speed

of modern computers, exhaustive search can be prohibitively costly and time consuming. Reference 35 also discusses the single (sole) source transportation problem, having the constraint that all the demand at a market must be supplied from a single warehouse. This problem is not polynomially bounded. Two methods are discussed for avoiding an exhaustive traversal of the search tree. The first is the use of a heuristic, a methodology or rule which will produce a feasible solution, although not necessarily an optimal solution. An example is the "regret" heuristic which at each step selects the market having the largest regret - the largest difference between the smallest and second smallest cost. In many cases heuristics ε e combined and perform better than any single heuristic. The second method is the use of an algorithm, a method or procedure, which will produce an optimal solution if given sufficient time. An example is the branch and bound algorithm which can "prune" the search tree by prohibiting search below a node at which the lower bound is greater than a known upper bound. Heuristic search is a major part of AI problem solving methods as well as Operations Research.

Queueing Theory

Reference 36 notes that of all concepts dealt with by basic operations research techniques, queueing theory appears to have the widest potential application and yet is perhaps the most difficult to apply. The theory of queues is the study of phenomena of waiting lines, which occur as customers needing service arrive at a service facility. Classically, customers arriving at the queueing system for service wait in line until service is provided or are served immediately if there is no line. Examples of queueing phenomena can be seen at banks, machine shops, a doctor's office, supermarkets, and in computer processing. A queueing system can be completely characterized by:

1. The number of customers
2. The arrival pattern of the customers
3. The service mechanism
4. The queue discipline

Customers may come from a finite or infinite population and may arrive singly or in groups. If the customer arrivals are completely random, then the number of arrivals per unit time can be modeled by a position distribution.

Reference 37 distinguishes three aspects of the servicing mechanism: (1) the availability of service (2) capacity (the number of customers that can be serviced simultaneously) (3) duration of service which can be constant or a random variable. Queue discipline describes the priority given customers in receiving service. First-in first-out is common, but other priorities are possible, such as last-in first-out or highest priority for shortest required service time.

Frequently, the cost of providing service must be balanced against the costs incurred by delay. Reference 38 cites a number of cases where queueing theory has been used to model physical processes and to predict the effects of altering system characteristics.

Dynamic Programming

The basic idea of dynamic programming is usually introduced by considering a discrete multistage decision process. For example, consider a square partitioned into nine smaller squares with hypothetical towns located at each corner of the small squares. (Common corners are associated with the same town). Let the lower left-hand corner of the original square represent town A and the upper right-hand corner represent town P. Furthermore, let each line connecting two sequential towns be labeled with a cost (perhaps, distance or travel time). The problem is to find the optimal path (minimal cost) from town A to town P. A brute force solution would be to simply list all the possible paths from A to P, along with the total costs, and then select the path with minimal cost. A more intelligent approach is to use dynamic programming which can reduce the required computational effort tremendously as the number of towns increase. The dynamic programming strategy is to proceed sequentially away from the end point P. At each sequential point (town) away from P, a decision is made about which path to take to reach P in an optimum manner from that sequential point, taking advantage of the previously computed costs from the preceding sequential points. This process is continued until point A is reached. The forward optimal solution is then obtained by moving forward and noting the costs computed at each point in the backward solution process.

Practical applications of discrete dynamic programming occur in transportation and scheduling problems. Bellman applied dynamic programming to the solution of continuous optimization problems in control.

Unlike linear programming, which refers to a specific mathematical model that can be solved by a variety of techniques, dynamic programming (ref 34) is an analytical technique which can be applied to a variety of mathematical models (ref 32). It is suited for multistage decision processes, where decisions must be made in a sequence and can influence future decisions in the sequence.

Dynamic programming attacks the optimization problem by splitting the problem into a sequence of stages in which lower order optimization takes place, rather than attempting to consider all constraints simultaneously. Reference 39 presents a dynamic programming computer program and applies it to a space program scheduling example. The major problem with dynamic programming is computer storage requirements. With a large number of stages, control options and states, the storage requirements can be more restrictive than the computation requirements.

Simulation

Despite the large number of analysis techniques offered by Operations Research, they are not sufficient or powerful enough to encompass all decision problems requiring analysis. Uncertainty, nonlinearity, nonstationarity, and a large number of states, possibly with interactions is characteristic of many actual problems. Frequently, simulation is used to describe current systems, to determine the effect of modifications, or to examine hypothetical systems. With the advances in electronics and high speed computers, large systems (such as power distribution networks, spacecraft, and industrial processes) are being simulated. Through time scaling, very fast processes can be "slowed down" for scrutiny, or slow processes can be speeded up. Other simulations are run in "real time" to obtain human participation and interaction. Real-time, man-in-the-loop simulation is used to train the operator of complex systems and to involve man as part of the total

system. Large scale, high fidelity simulation is used extensively by airlines, NASA, and in the power industry.

As automation technology advances, man is being moved away from direct control into the role of systems supervisor with many decisions being made by computer. Simulation will be needed to define the role and the interface of man and machine in the decision-making process.

Reference 40 notes that planning is one of nine core topics in Artificial Intelligence. Specific accomplishments cited include the development of hierarchical planning systems to allow the top-down generation of plans at various levels of detail, and the development of ways to represent plans so the plans themselves can be manipulated. Hart (ref 41) distinguishes between planning using AI techniques and conventional methods, noting that the planning function in conventional computer programs ordinarily amounts to following a rigid, inflexible algorithm specified by the programmer; standard numerical computations are an example of this, and are not usually thought of as "planning" at all. By contrast, AI systems can generate plans to meet situations that have been foreseen in only broad outline by the system designer; planning a sequence of database accesses to answer an unanticipated query from a user is an example of this. The following sections describe some areas in which AI planning techniques have been used.

Networks of Action Hierarchies (NOAH)

An early successful application of AI was Sacerdoti's development of NOAH (ref 42). This set of programs used a "procedural net" to develop problem-solving strategies. A procedural net is a network of actions (nodes) at varying levels of detail, structured into a hierarchy of partially-ordered time sequences. Nodes are linked to form hierarchical descriptions of operations and to form plans of action. NOAH was built to serve as the problem solver and execution monitoring component of the computer based consultant. The key to NOAH's ability to intermix planning and execution is that both the planner and the execution monitor use the same data structure: the procedural net. Demonstrations of planning using NOAH included: (1) sequencing painting a ladder and ceiling; (2) removing and stacking blocks in requested order; and (3) assembling an air compressor.

System Monitoring

Chien and others have been developing the concept of an on-board intelligent system for monitoring and diagnosis of aircraft systems using AI concepts of problem solving, planning, and knowledge engineering (ref 43). The knowledge base model abstracts a functional description of an aircraft using a Common Sense Algorithm (ref 44). The Common Sense Algorithm is a semantic net adapted to describe cause and effect relationships inherent in physical mechanisms. The knowledge base is divided into four levels; the highest level being overall flight plans and goals, and the lowest level being the aircraft subsystems description. Top-down monitoring can detect error through indirect evidence, and generate corrective procedure recommendations. Bottom-up monitoring interprets consequences of abnormal events. The long range goal is a system which can detect abnormal operation, diagnose the problem, and plan possible courses of action.

A precursor to the systems monitoring effort was the recognized need for deduction mechanisms which can construct plans in a dynamic environment, where data may not be "known" to the planner or may be changing (ref 45). Such a mechanism should have the knowledge that certain relevant information may not be known at all stages of planning, and should be able to modify a plan when new information is received. Reference 45 discusses the consequence of missing data to some current AI planning programs, including PLANNER, QA4 (ref 46), CONNIVER (ref 47), and STRIPS (refs 48 and 49).

Resource Allocation

Work is in progress to develop a knowledge base and model driven system to support a manager or commander in planning and scheduling resources (ref 50). The system exhibits inferential capability, yet the user can intervene to interact or take over. The system is being developed in the context of mission and resource planning for a naval air squadron. Part of the problem that is addressed is possible interaction among and between plans. These interactions occur through competition for limited resources. There are two types of modules in the program: a planner, that generates and monitors plans; and a scheduler, that coordinates resources with competing demands. Instead of trying to minimize a cost function the program is concerned with the maintenance of plans as potentially disrupting events occur. Hart (ref 41) notes that the ability of a commander to interrogate, in ordinary English, a set of distributed, computerized databases to form an assessment of his own assets or to test the feasibility of a contingency plan is one of a number of areas in which AI can increase the effectiveness of defense systems.

Robotics

Planning and scheduling is a higher order process and one of the distinctive differences between industrial robots and autonomous robotic systems. Industrial robots can be preprogrammed or trained for a task, but the industrial robot does not recognize the goal or objective of the task. An autonomous system would accept general objectives and develop a plan or sequence of actions to accomplish the objective. Often this plan involves navigation, obstacle avoidance, and manipulation. A major consideration is the manner in which the "robot's world" is defined, and the method used to develop and execute a plan. One planning system, called STRIPS (ref 48), maintains a collection of statements in first order predicate calculus to model the world, then conducts a heuristic search for an acceptable plan. After SCRIPTS creates an acceptable plan, a program called PLANEX1 (ref 51) carries out the plan by executing the actions. PLANEX1 will initiate replanning if the plan is unable to succeed.

Another approach to robot planning is a model called RECOGNIZER (ref 52) which was developed for the JPL rover. It is programmed using Common Sense Algorithms (CSA). RECOGNIZER uses CSA in several nets: (1) a causal (action) net for describing each action in the repertoire; (2) an outcome net with a measurable prediction of the result of an action; and (3) a decision net in which pattern matching (expected outcomes versus actual) takes place. Two strategies are used for correcting errors: (1) "failure reason analysis," a knowledge base of possible operational precondition, constraint, and information errors; and (2) "multiple outcome analysis," perturbing an action to get additional information about the error.

Reference 53 discusses three advantages of utilizing a distributed planning system for sophisticated robot systems: (1) a software structure similar to the hardware structure provides efficiency and robustness; (2) task-specific (domain-dependent) planning systems have less generality but can be tailored to the task; and (3) program size is directly dependent on problem complexity. Task-specific planning has been demonstrated in a blocks-world experiment (ref 54) which combined automated planning with hardware implementation to accomplish specified construction. Path trajectory planning is performed by a version of Lee's algorithm (ref 55) in a program called ZPLAN. ZPLAN uses the concepts of "difference" (distance from goal) and evaluation functions (ref 56), and can select short-term or long-term strategies. A four-degree-of-freedom manipulator and vision system execute the plans by performing identification and pick-and-place operations on a group of blocks on a planar surface.

LEARNING

Generally, learning can be considered as knowledge, understanding, or skill acquired by study, instruction, or experience. The concept of knowledge and understanding implying discernment and direct cognition may be too broad for automated systems. A more useful definition may be the modification of a knowledge base, world model, or control structure as a result of instruction or experience. This definition is general enough to include analogy to both the mental learning process and to acquisition of physical skills. Winston (ref 57) notes that there is a hierarchy of learning; from learning by being programmed, to learning by being told, to learning by seeing examples, to learning by discovery.

This progression is one in which more of the work is done by the student and less by the teacher. Knowledge representation and learning are major areas of research in Artificial Intelligence (AI). Knowledge can be represented as an ordered network of links (relations) and nodes consisting of symbolic concepts. Organization of such networks is discussed in the section on Data Bases. Learning, then, involves building and appropriately modifying the network structure. The concept of learning is involved in Control Theory also. Saridis (ref 58) defines a learning system as one in which information pertaining to the unknown features of a process or its environment is learned, and the obtained experience is used for future estimation, classification, decision, or control such that the performance of the system will be improved. Learning, in both the AI sense and the Controls sense, is discussed subsequently.

Expert Systems

One of the major successes in AI has been the development of "Expert Systems." Feigenbaum (ref 59) defines an expert system as an Intelligent Computer Program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. The knowledge base, consisting of facts and heuristics, was based on the expertise of practioners in that field. Expert systems bring together a large body of knowledge in a particular domain, and then draw conclusions from input data. Expert Systems have been developed for diagnosis and treatment of infectious diseases (MYCIN, ref 60), analysis of pulmonary function (PUFF), mass spectrometry (DENDRAL, ref 61), and proton magnetic reasonance spectroscopy (APSIS, ref 62).

A widely used form of knowledge representation is the "Production Rule," an If-Then statement, often with an associated "certainty factor." The knowledge base structure can be changed by introducing new production rules. Reference 63 notes that a great deal of time and cooperation between the human expert and the computer scientist or knowledge engineer is required to develop such rules. Recently, an induction program called Meta-DENDRAL (ref 64) has been developed which examines the data, and then does automatic rule-formation; discovering for itself rules for determining molecular structure from both nuclear magnetic resonance and mass spectrometry data.

Learning Concepts and Linguistic Constructions Through English Dialogue

A promising area of AI research, described in reference 65, is a system that can acquire new concepts and linguistic constructions through interactive English dialogue. The system, called KLAUS (Knowledge Learning And Using System), is preprogrammed with deduction algorithms, a set of syntactic/semantic rules, and a seed vocabulary to support effective tutorial interaction. The knowledge acquisition process used is "learning by being told." The system supports interactive, mixed-initiation, natural-language dialogues. A concept-defining syntactic structure is recognized, then an acquisition procedure is invoked to acquire the new concept and generate new entries in the system lexicon. Finally, the system can apply its acquired knowledge in a prescribed range of problem solving situations. Although still in an early development stage, this appears to be an interesting approach to knowledge acquisition.

Natural Language Understanding

An effort that is related to both pattern recognition and learning is the program SAM (ref 66), which understands stories. SAM can answer questions about English language stories, and can paraphrase and summarize them in English and Chinese. The program, which was developed by the Yale AI group, creates a linked causal chain of concepts that represents what took place. In order to fill in gaps in the causal chain which cannot be inferred directly SAM (Script Application Mechanism) uses "scripts". A script (ref 67) is a preformed, stereotyped sequence of actions that define a well-known situation. When a script becomes applicable SAM makes inferences about events that must have occurred between events about which it was specifically told. SAM is an important advance in computer understanding of natural language.

The METAL system (ref 68) is an example of an advanced machine language translation system. The system is used to translate German science and technology documents into English. The translation process is entirely autonomous in the computer. The METAL system has three parts: lexicon, grammar, and computational rules; and the latter can be used with the lexicon and grammar of any natural language.

Robot Learning

A system called RECOGNIZER (ref 52) has been developed to provide autonomous learning for the JPL robot rover. RECOGNIZER employs Common Sense Algorithms (CSA) language (ref 69) to construct causal networks. For each action in the robot's repertoire there is an "outcome" net which predicts a recognizable, measurable state

as a consequence of an action. The system learns from discrepencies between predicted consequences and experience. Two types of learning are possible: (1) learning how to categorize specific unknown situations through modification of the template at a node of a semantic net, and (2) learning how to avoid future errors as a result of error correction and acquisition of new stimulus-response pairs.

The Computer as an Intelligent Partner

Reference 70 discusses the possibilities of reducing human errors and workload by employing a computer to support an operator's intelligence. A symbiotic relationship is proposed whereby the computer assumes the role of an intelligent associate. In this man-computer team each partner assumes the most fitting task. This differs from partial automation where only routine work is done by the computer, and the operator makes decisions without help. It also differs from complete automation because the computer is used to support the man's intelligence. Consequences of designs of future man-machine systems resulting from cooperation between human and artificial intelligence are discussed in four areas: (1) computers capable of learning and adapting; (2) computer support in preparation and evaluation of information; (3) computer support in decision making; and (4) computer assistance in search and problem solving. The report cites examples of methods useful for assisting in the latter three areas. In the learning area the computer can be trained to exhibit intelligent behavior by explicit programming or by implicitely constructing a model based on continuous observation of human behavior (learning by example). If the operator behaves consistently for an extended period of time, a descriptive model of ever increasing reliability can be created, and can be used from simulating the human and predicting his action. Once the computer is trained, cooperation can occur at several levels. Tasks can be delegated to the computer for autonomous processing, the computer can wait and act only if the operator does not, or the system can act as a monitor and advise the operator.

Learning Control

Learning and hierarchical intellgent control systems is a regime common to both Artificial Intelligence and Control Theory. Fu (ref 71) defines a learning system as "a system that learns the unknown information about a process and uses it as an experience for future decisions or controls, so the performance of the system will be gradually improved." Several mathematical schemes have been used for the study of learning processes (refs 58, 71, 72): (1) trainable systems using pattern classification; (2) reinforcement learning; (3) Bayesian estimation; (4) stochastic approximation; (5) fuzzy automata; (6) stochastic automation; and (7) linguistic methods. Saridis discusses each of these in reference 58. Adaptive control, adaptive-learning control, and self-organizing control are active areas of Controls Theory research, which are addressing complex problem areas (eg, ref 73 through 77).

THEOREM PROVING

In theorem proving, one starts with a set of true statements (axioms or premises) and, by application of an allowable set of inference rules or operators, proceeds to expand these statements so as to include the theorem to be proved (ref 78). The theorem to be proved is at the onset a statement whose truth is unknown. In state-space terminology, the question is whether or not one can proceed from an

initial state (premises) to some goal state (theorem) using the allowable operators or controls (inference rules).

It is natural for man to consider the use of computers to carry out lengthy routine steps in theorem proving to avoid careless errors associated with this tedious task (ref 1). The fatigable effort accompanying straight forward manipulations (which must be done carefully to avoid introducing mistakes) is averted to allow man to concentrate his efforts at a higher level in the theorem-proving process.

Logical Inference

Hunt (ref 79) discusses an exhaustive technique which is guaranteed to provide a proof of a provable theorem; however, it may take a long time. Essentially, the technique (called the British Museum Algorithm) starts with a set of true statements and begins to expand the statements by logical inferences. The new statements are added to the list and further inferences are made. This process is repeated until the theorem to be proved appears in the process. Clearly, this is not the way to go, although one might discover some interesting statements along the way.

Hence, in theorem proving there are just too many possibilities that evolve, and methods are needed to alleviate this so-called combinatorial explosion problem.

Disproof by Counterexample

Only one counterexample is needed to disprove a theorem--if one is lucky enough to find it. The computer generation of counterexamples in topology has been examined by M. Ballantyne at the University of Texas.

Proof by Contradiction

A favorite approach used by mathematicians and logicians in theorem proving is proof by contradiction. If the theorem to be proved is assumed to be true and if, by logical inference, a contradiction is reached, then the assumption must be wrong. Hence, the theorem must actually be false. This is a fruitful approach if the theorem is false because it appears that there should be paths leading to contradictions.

If the theorem to be proved is true, then a lot of time can be spent looking for contradictions - which do not exist. Generally, the theorem to be proved is felt to be true so the approach in using proof by contradiction is to assume the negation of the theorem to be true and search for a contradiction. A contradiction means that the negated theorem cannot be true. If the negated theorem is false, then the theorem itself must be true. In other words, the theorem is shown to be true by showing that its negation is false.

Although the proof by contradiction is an improvement on the British Museum Algorithm, there is still need for guidance in the search for a contradiction.

Resolution Principle

The resolution principle has been around since the 1960's and is attributed to J.A. Robinson (ref 80). Cooper (ref 78) notes that it is specially suited for mechanical proof finders and for reducing the amount of search time required in proofs. Robinson's work, which was a shift away from trying to mimic human solutions to that of machine-oriented solution methods, brought about a renewed interest in mechanical theorem proving and information retrieval (ref 79). An advantage of the resolution principle is that it only involves a single rule of inference, which makes the expansion of statements by inference more manageable.

Hunt (ref 79) expresses the single rule of inference, which is the basis of the resolution principle, as follows: (A or B) and (not A) implies (B or C). Thus, if the left-hand side of the implication is true, then so is the right-hand side. If (B or C) is the theorem, it is proved.

Usually, one assumes the negation of the theorem to be proved and tries to reach a contradiction by application of the resolution principle. By combining statements in the theorem-proving process by the resolution principle, one hopes to reach a contradiction such as:

(A) and (not A) implies (null clause or contradiction)

With the resolution principle, there is still a need for guidance in choosing the proper statements to resolve.

Heuristic Methods

A heuristic is a rule-of-thumb which appears to work most of the time but which may occasionally fail. The fact that a solution to a problem is attained by using a particular heuristic (which may not work in another problem) does not invalidate the solution. A solution is a solution regardless of how it is found.

Cooper (ref 78) notes that the Logic Theorist (LT) of Newell, Shaw, and Simon (1957) was the first computer program to prove theorems, and he proceeds to describe several heuristic devices used. Working backwards is the main heuristic. Suppose the objective is to prove theorem T. The computer program recognizes that T would follow if other expressions could be proved. Another heuristic is used to choose which of these expressions is probably the easiest to prove. Additional heuristics are used to reject the expressions not likely to be theorems.

Four basic methods used in the LT to prove theorems are (1) detachment, which says that if it is desired to prove T and if it is known that A implies T, then attempt to prove A; (2) forward chaining, which says that if it is desired to prove that A implies B, and if it is known that A implies C, then attempt to prove C implies B; and (3) backward chaining, which says that if it is desired to prove A implies B and if it is known that C implies B, then try to prove that A implies C (ref 78).

The LT was successful in proving 38 out of 52 theorems of Chapter 2 in Whitehead and Rusell's, Principia Mathematica; and, later, on a larger computer managed to prove all 52 theorems (ref 80). The LT was the predecessor of the General Problem Solver (GPS) program, perhaps the most famous program in Artificial

Intelligence (ref 79). The GPS used "means-end" analysis, which means trying to reduce the difference between what one has now and what one wants to show. Insights into the structure of the GPS and LT are given by Minsky (ref 81) to reveal the basic ideas in these programs.

Cooper notes that the LT served an important purpose as an early test bed for heuristics but that heuristic methods are needed at a higher level in theorem proving.

The Geometry-theorem-proving machine of Gelernter (1959) is another theorem-proving program which uses a heuristic, namely the suggestive features of a diagram. Although done computationally, the computer in essence draws a geometric diagram and attempts to prove facts (subproblems). Particular coordinates are needed to construct a diagram; however, the heuristic tries to extend the resulting features to the general case (ref 78).

Also, splitting and reduction heuristics have been used to divide a theorem into smaller pieces that are more easily proved in order to speed up an automatic theorem proving routine (ref 82).

Nonstandard Techniques

Balantyne and Bledsoe (ref 83) have developed a procedure for automatically proving theorems in analysis using the methods of nonstandard analysis. In nonstandard analysis, the real field is extended so that a set of infinitesimals is clustered around zero, and out beyond the infinitesimals and finite numbers, are the infinitely large numbers (positive and negative). The infinitesimals are nonzero but smaller than any ordinary positive real number except zero. They are analogous to the differential elements dx and dy in calculus. This is an attempt to define infinitesimals to aid automatic theorem provers. For example, an element e is defined as an infinitesimal if $e < 1/n$ for any integer n. For more details, refer to reference 83.

Balantyne and Bledsoe describe their successful attempts in proving theorems on the computer with nonstandard analysis. They also point out their failures. They note that nonstandard methods appear only to aid in searching for the proofs of a certain class of theorems, and in other cases, offer no assistance.

Mathematical Induction

Two approaches to overcome the lack of computer theorem-proving power are (ref 83):

1. Proof checking program directed by user

2. Weak theorem-proving program with axioms introduced freely

Boyer and Moore (ref 84) have found that proof-checking programs are frustrating and require too much attention.

One reason for the difficulty with theorem-proving systems is that the researchers have not implemented the principle of mathematical induction. If induction is not used in a problem involving an inductive concept, then one must

ultimately assume that which is to be proved. The computational logic of Boyer and Moore (ref 84) includes the concept of mathematical induction.

Analogies

There are well-known analogies or correspondences between problems in electrical engineering and mechanical engineering. Whereas, for example, a mechanical engineer has a "good feel" at getting solutions in his domain, he may not relate readily to an electrical-type problem. However, by making certain associations, the mechanical engineer can transform the electrical problem into an analogous mechanical one, which, based on his experiences, may be solved immediately. For speed and economy, mechanical systems are regularily simulated by electrical systems.

In theorem proving, a correspondence is established between an unsolved problem A and a solved problem B in hopes that this may assist in solving problem A. Of course, creative insight is often required to recognize analogous problems cast in different formats.

An example of analogy used in Artificial Intelligence is Gelernter's program which solved axiomatic geometry problems by making use of a geometric diagram (ref 79).

Question-Answering System

Theorem proving is applicable to question-answering systems in that the world knowledge items stored in the computer (data base) represent axioms, and a question to the system corresponds to a theorem to be proved (ref 85).

In question-answering systems, a table of relations is stored in the computer. Then, by a set of axioms superimposed upon this structure, one can deduce facts which are implicit in the data base (ref 86). For example, suppose that the table lists the mothers and fathers of children and that it is desired to know who is the grandfather of a certain child. An illustrative axiom useful in this respect is: Father (x,y) and Father (z,x) implies Grandfather (z,y) where x is the father of y, together with z is the father of x, imply that z is the grandfather of y, although the grandfathers are not specifically stored in the data base.

A comprehensive data base system (MRPPS 3), developed at the University of Maryland, is described by Minker (ref 86). The system is based on mathematical logical and, in response to a question, has natural language and spoken output. A query to the system is first negated (i.e., an opposite query is assumed true) and then, consistent with the data base and knowledge axioms, a contradiction is reached by theorem-proving techniques (resolution, etc). Relational data bases per se are discussed in a separate section.

Man-Machine Systems

Man may roughly sketch the likely steps in the proof of a theorem, while the computer fills in the missing details or even suggests other lines of attack (ref 78). Man could provide sophisticated guidance to a theorem-proving machine, for computers sometimes spend too much effort in pursuing dead-end paths, which are

obvious to man at first sight (ref 2). A cooperative effort between man and machine systems requires a language of communication. Hunt (ref 79) doubts that any presently known computing language is satisfactory. Theorem proving is a non-numeric process and, as such, requires a symbol manipulation language. The Artificial Intelligence community heavily emphasizes the importance of the programming language LISP (ref 10). MACYSMA is a large computer programming system written in LISP for performing symbolic as well as numerical mathematical manipulations.

A man-machine, theorem-proving system has been used by Bledsoe and Bruell (ref 87) to prove theorems in topology. The steps in the theorem are presented on a scope in an easily understood form to a user. A feature called DETAIL allows the human to interact at any point that he deems necessary and only to the extent that he desires in the proof of a theorem. Hopefully, the system will evolve into a useful tool for researchers in topology. So far, only well-known theorems have been handled, and the mathematicians get very annoyed when they have to intervene in the proof in a trivial way.

Other Comments

A great deal of progress has been made in the area of theorem proving by computer; but, despite this, theorem proving by resolution generates too many fruitless paths. Consequently, some investigators use more restrictive methods, specifically depending on their particular problem area (ref 79).

The development of a higher-order calculus which can handle complex relationships better than the current language (which is limited to conjunctions of disjunctions) is another alternative which has been suggested in theorem proving (ref 79).

The more a theorem-proving system knows, the more difficulty it has in proving a theorem, because the system is often tempted to use irrelevant information. Boyer and Moore (ref 84) note that their system tends to avoid being confused by what it knows.

As with most aspects of Artificial Intelligence, advances in theorem proving are hampered by the fact that man does not know much about how man thinks.

DISTRIBUTED SYSTEMS

A distributed system is a structure or architecture in which elements are not centrally located, but which all contribute in a task. For example, elements could be sensors, controls, resources, computers, facilities, or people. Accomplishing the task with such a system is addressed by a number of methods. A single text would be inadequate to address all the methods, but the following sections illustrate typical problems and some applicable methods.

Decentralized Control

Decentralized control problems involve: (1) multiple decision makers (controllers); (2) either a single objective for all controllers (team theory), or multiple objectives (hierarchical control); and (3) decentralized information, where each

controller's information might depend not only on its own private observations, but also on the observations and actions of the other controllers (ref 88). Decentralized statistical decision making (team theory) is concerned with the problems of multiple decision makers having access to different information having uncertainities. Ho (ref 89) notes that the main ingredients of a theory of team decisions are (1) the presence of different but correlated information for each decision maker about some underlying uncertainty, and (2) the need for coordinated actions on the part of all decision makers in order to realize a "payoff". Team theory was originally developed by economists to model economic problems under the constraint of imperfect information. Reference 90 discusses similarities in team theory, market signaling, and information theory, and illustrates how problems in one discipline can sometimes be posed as related problems in another discipline, resulting in insight and alternative solution methods. For example, team theory may assume n distinct team members, each making one decision, while control theory may assume one controller making a sequence of n decisions. Reference 89 notes several variations in problem structure, such as whether each controller has knowledge of all previous information, and the extent to which the information of later decision makers depends on decisions (or controls) of earlier decision makers. Reference 91 develops an optimal control logic for a system having a variable time interval between information updates, and with the controllers knowing only the past history of the update intervals. A recent paper by Sandell (ref 92) presents a thorough survey of the control theoretic literature on decentralized and hierarchical control, and methods of analysis of large scale systems. The survey looks at techniques for simplifying model descriptions, procedures for testing stability, techniques for decentralized control problems, and techniques for hierarchical control problems.

Many current and potential applications are so large and complex as to preclude analytical modeling of the entire system. Efforts to develop a theory of optimal control for decentralized stochastic problems have been difficult and frustrating. Very often, an implementable suboptimal control is preferred to an optimal control solution which might be too complex or sensitive to be useful (ref 93). Consequently, researchers have tried decomposition, reduced order modeling, and heuristics. Reference 94 presents a method of parameter reduction by an orthogonal projection of the exact generalized vector onto a subspace of lower dimension based on minimization of the error norm for a chosen scalar product. Reference 92 discusses aggregate methods and singular and nonsingular perturbation methods for reducing the order of large systems.

Distributed Computer Systems

A large and fast-moving area of computer science involves developing architecture and communications for networks of distributed computer facilities. Many excellent references are available. An example of a distributed microprocessor system is presented in reference 95, which describes the computer network used to control the Shiva laser facility at Lawrence Livermore Laboratory. It is a hierarchical network with one large computer controlling four smaller system controllers which in turn control 50 small computers acting as front-end processors. A parallel communications link, a serial asynchronous link, and shared common memory interconnect various parts of the system.

Hierarchical Systems

Hierarchies abound in nature and in human organizations. The reasons for hierarchic structuring are (1) the components at each level of the hierarchy are themselves stable entities, (2) the hierarchic systems require much less information transmission among their parts than other types of systems of the same size and complexity, and (3) the complexity of an organization, as viewed from any particular position within it, becomes almost independent of its total size (ref 96). A hierarchical structure has several advantages: (1) decisions are made throughout the system, so the rate of decision-making at any point can be consistent with the time required to collect data and analyze a problem; (2) the type of decision and the detail involved can be made at the point or level having the most relevant data and with direct awareness of the result; and (3) authority for implementing a decision can be delegated to the level at which it is made. Generally, decisions in a hierarchical structure have increasing precision with decreased overview. That is, goals and policies are set at the highest levels based on the "big picture," and detailed decisions are made at lower levels but with a limited field of view. Hierarchical structures for control of distributed systems and for the architecture of distributed computers have already been mentioned. Barbara (ref 97) describes a robot control system with a five-level hierarchical structure. The highest level assigns tasks to a group of robots and manages the data base, and the lowest level controls individual servos. Saridis (ref 98) suggests that hierarchical control structures for robot control represent a systhesis of several disciplines; with Artificial Intelligence at the highest level for goal selection and interface with man, Operations Research methods in the middle for planning and scheduling, and Control Theory at the lower level for optimal task execution.

KNOWLEDGE BASES

There seems to be a widely held belief among those who are attempting to discover the underlying principles and essentials for the synthesis of intelligent systems that one of the most fundamental breakthroughs in the field will be discovering how to organize knowledge. Although a considerable number of innovative approaches to the problem have been tried with varying degrees of success, there appears to be consensus that the problem remains unsolved. References 8, 9, 35, 86, 99, 100 through 106 provide material on knowledge bases related to the discussion in this section.

In this brief discussion of knowledge bases, the intent is not to comment on data bases in general but only on those which are specifically designed for use in implementations of Machine Intelligence or Artificial Intelligence. For the most part, approaches to knowledge representation differ primarily in what they emphasize. Some emphasize relationships such as those among primary objects (parts) which make up a more complex object. Others emphasize procedures and some do little more than minimal organization of facts. The latter tend to rely heavily on the program which uses the knowledge base to extract relationships which could have otherwise been stored explicitly.

A characteristic which underlies most schemes is organization by sets or classifications. Members of the top level set are broken each into a subset etc. Subdivision into subsets continues until further decomposition is impossible. For example, the residence of person X might be stored as planet Earth, North American

Continent, United States, Los Angeles, 201 Anywhere Street. More or less, this is just putting elemental units of information (facts) in the right boxes so that they can be retrieved in a quick, orderly, consistent way. Even though procedural knowledge as well as that of states of being may be encoded and stored in this manner, some of a system's (computer program) knowledge must be considered to reside in the program which manipulates the data base. The program "knows" what to do with the data.

The list processing computer language, LISP, and its derivatives constitute the single most powerful tool used by Artificial Intelligence Research. LISP is much more of a symbol manipulation language than it is a number crunching language. Of particular importance to A.I. researchers is its characteristics of treating both data and instructions exactly the same. Thus, a program may readily modify its own instructions. Some programmers regard LISP as a language just one level above machine code. Practically all Artificial Intelligence programs use LISP-like languages to contain their static representations (construct the data base) as well as to encode the procedures which process the information. In a sense, therefore, LISP might be said to remain one of the foundational aspects of the state-of-the-art in knowledge base construction.

Among the more common forms or ways of thinking about the organization of data bases are tables, network graphs, tree-structures, and frames. Tabular structures are probably the simplest of these. Often they are called relational tables because data are related in predefined ways according to their positions in the table. Network and tree representations are much the same in appearance and may in some applications be exactly the same. They both consist of nodes with interconnecting links. Trees usually depict a space of alternative sequences from which paths of actions or paths from state-to-state may be evaluated and chosen. Examples of these range from ordinary highway maps with cities being the nodes and highways the links to outlines of potential moves and resulting game states in a game like chess. Networks, on the other hand, are more often used to merely express definite properties of some object and how they are related in more of an existential sense than in a path finding sense. Winston (ref 8) shows how a simple children's blocks arrangement to form an arch may be depicted by a network. The arch consists of a rectangular block resting on top of two other rectangular blocks. These blocks are represented by nodes connected by tagged links. These tags tell how the blocks are to be arranged with respect to each other.

Having had very little success in research aimed at discovering the essence of intelligence in a very general sense and from it defining general algorithms for use in intelligent systems, AI research has turned to more domain-specific applications. Current research operates on the premise that specific problems should be solved but the generation of their solutions should be done in an atmosphere of sustained awareness of the need to generalize. This approach as a natural consequence has been applied to knowledge base development as well. It is useful to remember that while this section of this paper is devoted specifically to knowledge bases that they and their development are not distinctly separate from the rest of AI research. Quite the contrary is actually the case. In fact, it is often difficult to break out the knowledge base of an intelligent system as a separate entity.

In the remainder of this section, some specific approaches taken by particular researchers and research teams will be treated one at a time through very brief discussions.

Research in this area is going on at several institutions world-wide. The authors of this paper are most aware, however, of the work being done by Jack Minker (ref 86) at the University of Maryland. Relational data bases consist of tabulated facts whose relationships are expressed by the structure of the table. The objective is to merge these data bases with program implementations of mathematical logic (formal logic, predicate calculus) along with appropriate axioms to permit new facts to be derived. (The principles of mathematical logic are discussed in another section of this paper). Minker and his students have put together a fascinating small computer program to demonstrate how these systems operate. The data base contains the family relationships of several generations of hypothetical family members. Each person is identified by an integer. The explicitly stored information is in the form of 3 is the mother of 5, 2 is the grandfather of 4, etc. The axioms (intensional data base) contains more general relationships and assumptions. An axiom could state, for instance, that the father of an individual is the husband of the mother of that same individual. The computer program might be asked to determine the paternal great grandmother of person 62. Through a series of self-initiated proofs of theorems the answer would be produced along with a explanation of how it was derived. Minker is presently extending this work to apply formal inference mechanisms to large systems of relational databases so that complex queries can be decomposed into more conventional queries which can be readily handled by the component relational data bases.

Frames

The idea of representing knowledge in structures which he calls "frame" was put forth by Marvin Minsky at the Massachusetts Institute of Technology in 1974. Frames are descriptive outlines or general structures of complex objects or events especially ones that commonly recur. An event frame for "eating out at a nice restaurant" would likely include expectations of such things as making reservations, being served by waiters or waitresses, tipping, paying after the meal, etc. A given frame will contain a set of descriptors at one level. Unit constituents of this frame may also be described by their own subframes at another lower level. For example, a frame could contain a general expectation of an executive office at the top level. Its units would be such objects as a wooden desk, a conference table, windows, chairs, a telephone and the like. In turn the table might have a subframe whose units are four legs which support a flat surface etc. Thus, the frame not only retains sets of facts, but makes explicit how they are related as well. The example of the table subframe would make clear that the table's legs are attached to the flat surface in a more or less particular way. Minsky suggests that humans use a mechanism such as the frame to store and provide a framework for processing information. By noting the aspects of the real situation which differ from the expectation, a human is able to acquire an awareness of a new instance of a stereotyped situation to the level of detail required for his needs. Frame transitions can be used to describe cause-effect relations and the like. In a sense somewhat different from the examples cited, the frame could also be used to store the parsed parts of a sentence. To the authors' knowledge the "Frame Theory" has not been applied in specific computer programs thus far. It is an elaborate theory which will likely undergo extensive applications testing and much additional research.

Common Sense Algorithms and the Bypassable Casual Network

At the University of Maryland, Chuck Riegor has devised a theory for qualitatively describing causality and functional relationships. He and his students are actively expanding, applying, and testing this theory that includes a formalism within which to express algorithmic knowledge. Especially in simple mechanical devices built by humans, cause and effect relationships are definite, repeatable, and often easily understood qualitatively. To capture this qualitative understanding, Reigor has devised a set of symbols and conventions for interconnecting them. Descriptions which use these have an appearance similar to that of a flow chart and emphasize such qualities as enablement, gating, and the prescence or absence of particular preconditions. These graphical descriptions can, beyond some minimum, have almost as much detail as their user wants. Let's imagine that a nightwatch person wants to turn on the lights in a particular office during the night when the building is unoccupied. The common sense algorithm (CSA) for this process would be relatively simple. First the correct circuit breaker to enable the lighting circuit in the required bank of offices would be activated. The effectiveness of this action would be subject to the precondition that no interruption of electricity from the power company existed. Activation of the light switch in the particular office would permit electricity to flow through the filaments in the overhead light bulbs. The resistence of the filaments to the flow of electricity would generate heat which would cause the filaments to glow and give off light. More detailed preconditions could have been added such as a requirement that the filaments in the light bulbs be intact. Of course, CSA's are applicable to the description of other dynamic behavior involving cause and effect, state changes, etc. which are different and perhaps more complex than the functioning of simple mechanical or electrical devices.

Riegor proposes that the CSA could be used as a unit in a large network for containing organized knowledge. The knowledge in this network, which he calls a bypassable causal network, would be organized to include and express knowledge of what other knowledge is useful or required to solve particular kinds of problems. The system would have the capacity to, in a sense, iterate on relevance and context in such a way as to play them off against each other to sort out the knowledge or algorithm applicable to the solution of the problem at hand. The use of context and relevance in antagonistic feedback is based on Reigor's almost axiomatic contention that "relevance seeks context" and context restricts that which is relevant. The term "bypass" is used to indicate "going around" a system node whose information requirements have already been satisfied indirectly by shared information obtained from previous system requests of information for other system nodes. This parallel information use improves the efficiency with which the system can assimilate and construct a model to use for solving the problem. The "short cuts" which come about can be combined by the system into longer sequences of bypasses which may be remembered for use in new environments in which they may again be applicable. Reigor argues that these systems would have an advantage in flexibility because their bypasses would be context sensitive. Collectively or on an individual basis bypasses could change or cease to exist depending on context. Reigor compares his causal selection network to "a mother who asks all the right questions before choosing which son will be the best for which job."

Knowledge Representation Language (KRL)

The term KRL refers to a Knowledge Representation Language under development at the Stanford Artificial Intelligence Laboratory. Daniel G. Bobrow and Terry Winograd have been responsible for much of the basic KRL research. On the surface the

language appears to be very similar to other schemes for encoding and containing knowledge in the sense of having the characteristic of organizing information directly through multilevel sets of classifications. In fact, however, the underlying organization concepts that it uses are quite different. Since humans tend to often use the framework of telescoping sets to classify themselves and entities within their environment (management structures, filing systems), it seems somewhat logical to assume that knowledge organization within the human mind is of a similar structure. This assumption may, however, not be true; and even if it were, might not indicate that intelligence mechanizations for information processing should use that type of knowledge organization.

Bobrow and Winograd have based KRL on a different premise which they believe more nearly duplicates human knowledge organization. They maintain that generally human knowledge of particular entities (be they physical objects, concepts, or procedures) does not exist in the form of explicit definitions. Rather, humans store knowledge of a new object in the form of comparisons and contrásts with what is already known about similar objects. The new object is considered from several perspectives giving perhaps a richer description of the object than would be obtained through explicit definitions. The basic idea is to match the new object to a prototype and further specify the prototype until a suitable match is obtained. The prototype might be thought of as a collection of general characteristics of members of some class to which the new object may belong. The prototypical knowledge may be viewed as default characteristics which are true of a class stereotype. The prototype acts as a standard with respect to which the comparative knowledge is acquired. The KRL attempts to describe from a wholistic point of view, rather than from a reductionistic one.

The actual implementation of KRL uses essentially LISP and catagorized descriptions. Chunks of knowledge are contained in what are called units. Units fall into one of seven primary categories called basic, abstract, proposition, manifestation, individual relation, and specialization. These are considered mutually exclusive in that a given "chunk" of knowledge cannot be fitted into more than one unit. Although, the descriptors within a unit might be construed to be devised to telescope information into its most basic primatives, they are actually intended to provide several different ways of viewing the information. It seems that the categories are organized in more of an ordered parallelism than in telescoping sets. It also appears that KRL and Minsky frames have very much in common.

Expert Systems

There is a whole class of computer programs which attempt to capture the very specialized knowledge of experts in very narrow fields. Probably the best known of these is MYCIN which has resulted from the heuristic programming research at Stanford University. MYCIN is very effective (on a competitive level with human physicians) in diagnosing infectious disease of the blood. The program operates from an elaborate set of rules determined from exhaustive interviews with human experts who specify conditions and preconditions under which certain inferences are appropriate. Both expert systems and production rules on which they depend are discussed several places within this review of intelligent systems; thus, no more detail will be presented here. The few statements at this point are intended simply to convey the idea that expert systems are an important form of storing knowledge.

Hierarchically Ordered Knowledge Bases

At least two examples of systems which use explicit hierarchical knowledge structures may be cited. These are Carnegie-Mellon's Hearsay II speech understanding system and the Automated Inflight Monitoring System of the University of Illinois.

At the Coordinated Science Laboratory of the University of Illinois, R.T. Chien and his students are putting together programs for automated, inflight monitoring and diagnosis of aircraft systems and subsystems. The primary research objective is to uncover key methods of organizing intelligent knowledge bases. It is presently believed by this research team that knowledge at several hierarchical levels is a very significant aspect of the solution. They have divided the domain of aircraft systems knowledge into four levels. At the top level are overall flight goals (destination, takeoff parameters, cruise parameters, landing parameters etc.). The second level contains basic aerodynamic information in force diagram form. At this level, the system knows about lift, drag, angle of attack, etc., and how they interact. On the third level, is knowledge of aircraft controls, where the effects of control surface deflections, throttle settings and other similiar variables on the aerodynamic forces of level 2 are stored. At the fourth level, which is the lowest, is subsystem knowledge at the component level. Level 4 is concerned on a functional relationship basis with things like fuel pumps, valves, electrical generators, and circuit breakers. The qualitative, functional model of the subsystems can be analyzed to determine the consistency of sensor information, the presence of propagated failures, and the most probable source of component failures through dependency-based backtracking. The combined result of these analyses should determine first whether a sensor or a component has failed. Second, it should specify the particular failed device.

The four knowledge levels described represent knowledge at different kinds and degrees of abstraction. Together they form a composite knowledge that will permit information processing which might have been impractical with other forms of organization because of combinatorial difficulties. The monitoring process can be propagated through the knowledge base from either the top or bottom. In top down monitoring, inadequate progress toward achievement of overall flight goals triggers analysis at the lower levels. Out of this might come a result such as the desired altitude has not yet been obtained because the rate of climb has been low because thrust has been low because engine No. 2 has failed because its fuel filter is clogged. Bottom-up monitoring is just the opposite. In that case, the consequences at the top level of failures which occur at the bottom level are determined. Internal simulations can be performed to explore "what-if" actions and situations. This capability is especially important to the maintenance of a prioritized set of flight parameters and goals.

The Hearsay II speech understanding program developed at Carnegie-Mellon University uses knowledge hierarchies as a part of its overall scheme for organizing knowledge. Pattern recognition and systems of supporting hypothesis of the meaning and correct interpretation of various patterns are the underlying principles on which the system is based. Speech patterns to be recognized are grouped in several hierarchical levels which are designated as knowledge sources. Some example levels, are phonemic, phonetic, syllabic, lexical, and phrasal. These knowledge sources (levels) each monitor incoming data for patterns applicable at its level. When these occur, the knowledge source will respond. One important benefit of this organization is that it facilitates a great deal of parallelism in the pattern search.

SEARCH

Suppose a father wants to drive into town, but does not have the car key. His son, who is asked to find the key, goes through some search process to locate the key. The son may recognize the correct key when it is found, or he may have to give his father a ring of keys, which hopefully contains the car key, which his father can subsequently identify. In this situation, the father does not care how the son found the key, only that he found it. Had the son returned with no key, or with a set of wrong keys, then the father might have suggested other places to look. In general, search involves a strategy (which may be very crude) to move from one point to another to seek out possible solutions to a problem, and a method of recognizing an acceptable solution, either at points along the search path or at the end of one or more paths.

In some cases, under certain conditions, a search may be guaranteed to find a solution; whereas, in other cases, the search may be unsuccessful because the solution does not exist, or the search strategy simply overlooks the answer.

Search is associated with any type of problem solving where one is "looking" for a solution or desired goal. Hence, it is only natural that similar search techniques can be applied in a variety of disciplines such as Control Theory, Operations Research, and Artificial Intelligence. Only graph searches which are applied in Artificial Intelligence are mentioned here.

Graph Searches

A good representation aids in understanding a problem and in finding a solution. Graphs are one means of displaying the structural relationships in a problem.

Breadth first algorithm (refs 27 and 9) - The basic idea in this algorithm is to start at the beginning point (or points) and expand or draw lines from this point to all other reachable points. Then, each of these reachable points is treated in a similar fashion, and the process continues until a destination point is reached which is identified as the goal or solution. Once the goal is attained, one simply backtracks to find a solution path from the beginning point to the end goal point. Finding the solution by this "brute force" manner may take a lot of expansions, thereby being expensive in money and time.

Uniform cost search (ref 27) - This algorithm expands the graph from the starting point according to the distance from the starting point and, in general, produces a shorter solution path than the breath first algorithm. The graph grows by the expansion of the point closest to the starting point.

Ordered search (ref 27) - The breadth first and uniform cost methods are special cases of the ordered search. Let $f(n) = g(n) + h(n)$, where the cost $f(n)$ to go from start to finish (goal) is equal to the sum of the cost $g(n)$ to go from start to node (town) n and cost $h(n)$ to go from node n to goal. In the ordered search, the next node to expand is based partially on an estimated solution path of $f(n)$. This estimate depends on the problem and ingenuity of the investigator.

Depth first search (refs 9 and 27) - In this method the search follows one path to its end, then the next path to its end and so forth. If the goal is attained on

one of these paths, then that path will not in general be the least cost solution. Humans are disposed to this type of search.

Heuristic search (refs 9 and 27) - The search is said to be heuristic in that knowledge about a problem is used to aid the search. The problem is structured as a graph using knowledge about the problem. Production rules are used in "expert system" programs in Artificial Intelligence to move through a graph in search of an answer to an inquiry.

Minimax search (refs 9 and 27) - Relative to game theory, one tries to minimize his maximum loss to an opponent. It is assumed that the opponent will always counter with his best move.

N step look ahead (ref 27) - For example, in chess or checkers, one considers all possible situations which can be reached in n moves. Current computerized chess programs have the option of varying n to increase the program's competence.

Alpha-beta procedure (refs 9 and 27) - This is a logical procedure that simply says not to explore a set of paths any further if, at any point, they can be shown to be worse that some other path already explored.

Means-end analysis (refs 9 and 27) - Operators for moving from one point to another are defined, then the difference between the present point and the desired finish point is used to select the appropriate operator to reduce this difference. This has the flavor of feedback in Control Theory.

Common sense algorithm (ref 104) - A graph is constructed in a "cause-and-effect" manner which aids in searching for failures in a system. The algorithm is discussed in another section of this paper and reference 3.

Dynamic programming (ref 39) - The idea in this method is to start at the end or desired goal and work backwards to the starting point. Moving away from the goal, step-by-step, one asks the question: From this point, which path should one take to reach the goal? Continually backing up and asking this question, the investigator finally encounters the starting point. Meanwhile, enough information has been gained to get the forwarded solution. The basic fact used in this method is that the optimum path to the goal from any point along the optimum path from start to goal is this same latter optimum path. A drawback of this method is the enormous amount of computer storage required even for a moderately complicated problem.

There are numerous optimization procedures which are applied in Operations Research, Control Theory, and Artificial Intelligence to search for a solution to a problem.

Control Formulation of Artificial Intelligence Problems

The following example used by Nilsson (ref 9) indicates the application of methods used in Control Theory to Artificial Intelligence. Consider a checkerboard-type structure with 9 spaces. Number chips from 1 to 8 and place these chips on 8 of the 9 spaces. Now, the problem is to move the blank space around until the chips are in some desired positions. This is an Artificial Intelligence type problem, but it is easily formulated in the language of the control engineer. Nilsson defines a control law for this problem as follows. Move the blank space so as to minimize the

error in location, which is defined as the number of chips out of place. Other types of control laws could also be used. This type of formulation is comfortable to the control engineer.

Nilsson (ref 107) states that "the problem of efficiently searching a graph has essentially been solved and, thus, no longer occupies AI researchers. This one core area, at least, seems to be well under control."

Having the basic idea of a search method and knowing enough to implement it are two different matters. For a detailed discussion of the search methods, refer to the appropriate references.

HEURISTICS

Webster's Seventh New Collegiate Dictionary defines a heuristic as that which "serves to guide, discover, or reveal." It further qualifies it to be "valuable for empirical research but unproved or incapable of proof." Various aspects of heuristics can be found in references 8, 9, 35, 108 through 113. Herbert Simon (ref 110) defines a heuristic as a "rule for exploring a reasonable or proper set of choices." Similarly, Feigenbaum (ref 111) calls heuristics "rules for plausible reasoning or good guessing." From these definitions it would seem to follow that heuristics are weak, imprecise and not suited for inclusion in systems of mechanized intelligence. Yet, human intelligence makes extensive use of heuristics.

The number of variables associated with many problems is so large that strict systematic procedures to account for their effects in all combinations are not practical. The solution either takes too much time to obtain or the computational and memory requirements are excessive. Thus, to deal with these problems in a timely, efficient manner, devices which permit the choices to be explored to be redued to a smaller set of the more promising ones are needed. Heuristics fill this requirement, but they provide no guarantees. Solutions obtained may not be the best or even good. In a sense using heuristics is a gamble which trades processing time for an answer which is probably acceptable but not likely optimal.

Heuristics are most often used to reduce the complexity and effort of search through "trees" or graphs of organized or related units of information. For instance, consider the problem of finding the best route from point A to point B through a given network of highways. Knowing that a certain bridge is out will eliminate part of the network as containing possible sections of the desired route, thus, simplifying the problem. Here, the heuristic does not have the probalistic character with which heuristics are generally associated. If, on the other hand, the bridge which is out is replaced by a drawbridge with some probability of being open, the heuristic becomes more conventional. The point is that in addition to reducing the space of available choices the application of the heuristic may or may not introduce the possibility, however small, that good optional paths will also be eliminated. Although some general characteristics of heuristics and the nature of their application may be found, specific heuristics usually apply to a very narrow set of problems.

On a larger scale elements of domain-specific information may be organized to form more comprehensive units of heuristic information called Expert Systems. These systems contain rule-of-thumb knowledge usually obtained from human experts in a

particular field and are constructed to deal in a useful way with problems of a narrow set or type within that field. Computer programs like Stanford's MYCIN (ref 60) for analyzing bacterial infections of the blood interact with a human user to acquire and analyze data. Usually they question the human, becoming more and more specific in their questioning as evidence supplied by the responses narrows the number of possible conclusions. Some of these programs are very sophisticated and useful. MYCIN, for instance, is actively used to identify sources of blood infections and to suggest treatment. Most expert systems also have an explanation capability which may be used by the operator to determine to the desired level of detail the reasons that the system drew specific conclusions.

Many of these expert systems are "rule-based systems." That is, they consist of production rules which are heuristics having a particular form. Winston (ref 8) calls the two elements of this form situation-action pairs. There is a set of preconditions (things to watch for), which when satisfied, trigger certain actions (things to do). The information in expert systems is more appropriately thought of as embodied in productions of the form Winston calls premise-conclusion pairs. In these production specified combinations of facts (evidence) lead to the assertion of particular facts as having been deduced.

Production rules are much more than simply an elemental format for cataloging heuristic information. They may, for instance, be used to implement the "and" and "or" logic functions in a tree or graph of related facts. Once developed, these trees may be used to go from basic facts to logical conclusions through a process called "forward chaining." Conversely through "backward chaining" conclusions can be used as the starting point from which supporting facts may be obtained (ref 112).

A system of productions may be used to operate on data associated with a problem to be solved such that through nondeterministic interaction between the productions and the data base the problem state evolves from its initial condition to the desired termination condition. When productions whose preconditions are satisfied by the data base are triggered the resulting actions of the productions modify the data base. The changed data base may now satisfy the preconditions of other productions, etc. Thus, unplanned behavior can result. According to Winston (ref 8) some psychologists as well as designers of smart computers believe that the mechanism humans use to solve problems can be well modelled by a production system.

Production systems offer the advantage of flexible structure. Individual productions can be easily added to or deleted from the system as required. But these systems can also grow so large that the nature of the interaction among the productions is not understood or controlled. One possible solution to this problem is to organize the productions into subsystem units.

Gerald Thompson (ref 35) is developing techniques which employ heuristics to quickly obtain solutions in operations research to problems which have extremely large numbers of variables. He is dealing with problems so large as to have millions of variables and thousands of constraints. Thompson defines a heuristic simply as "a quick and dirty way to get a feasible solution." A heuristic called "regret" is being applied to problems whose objective is minimizing the maximum shipping time from factories to markets. Regret is a measure of the amount of goods which cannot be obtained from the supplier with the least shipping cost and thus, must be procured in order from the next least costly suppliers. Candidate solutions are grouped according to computed values of regret. From groups with the largest regrets, solutions with smallest costs are chosen at random. The resulting solutions are

regrouped according to regret as before and a new set of solutions are found. The process is applied iteratively until a single solution is found. The solution obtained is generally within 3 to 5 percent of the optimum.

The "rubber band" heuristic is being applied to the classic traveling salesman problem. The objective is to find the route through N cities that goes through each city exactly once and minimizes the total mileage. The heuristic gets its name from the analogy of nails in a board intertwined with a rubber band. The nails represent cities and the rubber band a possible intercity route. The idea is to start with three cities. Find the optimum path for them. Then, continue to add cities until solution for the required number of cities is found. Dr. Thompson states that solutions generally require about 100 iterations of the heuristic. For problems with less than 35 cities, the optimum path is generally found.

Thompson has also been experimenting with using probablistically combined sets of heuristics to reduce the amount of search in problems with a large number of variables. Although he has not proved that this technique is more effective, he states that his experience to date indicates that it is.

The state of the art in heuristics is, perhaps, most importantly its increasing acceptance as a viable and useful problem solving tool. Even now, though, scientists and engineers tend to regard heuristics as merely guessing, not compatible with their accustomed, disciplined mathematical procedures and physical laws. Two situations seem to be altering this view, however. The digital computer is providing the computational power to carry out brute force searches through many combinations of large sets of variables to find ones satisfying particular criteria. However, as the capability to exhaustively search through larger and larger trees of data develeopes, even larger search problems arise. It seems likely that for the foreseeable future, the problem size will stay ahead of the current computing capacity. Thus, techniques which reduce the amount of search within available computing capacity must be utilized. Heuristic paring of the search tree is frequently all that is available. Not only that but as evidenced by the work of Thompson (ref 35) the optimum (or very near it) is often found through the heuristic search anyway.

The second situation which is causing acceptance is the growing belief that much of the human problem solving process consists of trial and error application of heuristics. It is difficult for the scientist or mathematician to dismiss heuristics as an unhealthy contaminant of his vigorous mathematics and established physical laws when he realizes that these tools were devised by clever human application of heuristic information.

The state of the art, otherwise, is probably represented by current research and development in expert systems. At Stanford University, Ed Feigenbaum (ref 111) and Bruce Buchanan (ref 108) are extending from their background in developing expert systems, such as DENDRAL for analyzing mass spectrograms and MYCIN for diagnosing blood infections, to research for understanding and using meta-rules and meta-knowledge. An intelligent expert system should be aware of what it knows and understand how to use what it knows. It also needs knowledge acquisition strategies to obtain particular facts in a particularly good sequence to solve specific problems. Buchanan (ref 108) is studying how to use heuristics to exploit redundancy in the problem data to reduce the uncertainity associated with particular bits of that data. If the same conclusion is implied by several alternative paths from several pieces of uncertain data, the validity of the conclusion is strengthened in proportion to the number of paths and the certainty of the data from which they originate.

The Stanford computer program called Emycin contains the inference procedures of the MYCIN expert in a general form which accepts expert knowledge from fields other than only medicine as input. Thus, the program can be utilized to produce a new expert system in a new field. In addition, the previously human task of acquiring the human expert's knowledge and coding it in the form required for use in an expert computer program has been automated. Programs have been written which hold dialogue directly with the human experts to generate the needed knowledge base.

At M.I.T Gerald Sussman (ref 109) is studying expert systems which deal with engineering problems. In contrast to most other expert systems, the knowledge base for these consists of more rigid and precisely known information and relationships. Even so, these systems must do more than play back information taught in formal engineering courses. The expert program must capture, understand, and utilize the essence of that facility which the expert engineer has acquired from practical experience. The engineer can often quickly analyze a system through trial and error application of informed guesses based on typical system characteristics and configurations. An evaluation of design alternatives at the beginning of a synthesis process, for instance, may require only an expected range of system values. Determining these likely ranges may involve much less time-consuming computation than determining precise limits of particular system variables. Not only can experimental knowledge shortcut analysis computations, but also it can provide good guesses of what system configurations should be evaluated. Sussman, then, is studying how to best blend precise knowledge of certain physical relationships with educated engineering guesses to form more effective engineering expert systems. The value of these systems was dramatically demonstrated, recently, by a very complex microprocessor chip which was built from a design produced in a very short period of time using one of Sussman's electrical design experts (ref 4).

In control theory, Hsu and Meyer (ref 114) are very aware of the usefulness of heuristics in their statement (with respect to conjectures by Aizerman (1949) and Kalman (1957) to infer global asymptotic stability of a certain class of nonlinear systems from the stability of an associated linear system): "The conjectures have been proven wrong in general; however, as they usually only fail in extraordinary situations, they can sometimes provide useful rules of thumb for an engineer."

EVOLUTIONARY PROGRAMMING

Finite-state machines are used in evolutionary programming. A finite state machine produces an output sequence of symbols in response to an input sequence of symbols from its environment in an effort to minimize a certain cost function. Mutation of the machine is influenced by its performance, along with some randomness and transformation logic. The evolution proceeds to find a better and better finite state machine for the task at hand. Unsuccessful machines tend to become extinct (refs 115 and 116).

Evolutionary programming is concerned with prediction (what will be the next output from the environment?) and with control (what should be the next control action based on the predicted next input and cost function?).

In evolutionary programming, a finite-state machine is used to mathematically represent the problem. It does not matter that some function is not differentiable, that the system is nonlinear, or that not enough information exists to model the

system by conventional means. This generality, of course, has its price. In reference 116 evolutionary programming is compared to quasilinearization in a system identification problem. Approximation of the problem by a system of linear differential equations favored the later method significantly. Solutions by the quasilinearization method required about 1 minute of computer time (CDC 3600) and 5 iterations; whereas, evolutionary programming used 6 minutes of computer time and yielded a cost function still about twice the size of the true minimum value. However, if no other method is available, this may not be so bad for a simulated smart computer program.

In evolutionary programs (which have been written in FORTRAN computer language), an initial state machine is specified. Thereafter, the current three best machines are retained for generating offspring. Modes of mutation include (1) adding a state, (2) deleting a state, (3) randomly changing the next state, and (4) randomly changing the initial state. The probability for the selection of the next machine to be mutated is made proportional to the inverse of its error score.

CONCLUDING REMARKS

Automation is a matter of degree. As technology advances, man continues to automate at a little higher level than before. With advances in the electronic computer, automation has become more sophisticated, and there is a continuing effort to use the computer in solving problems and making decisions to as great a degree as possible. Making computers take over some of the more complicated thinking is an impetus to understanding that process by which humans do this task. How do humans solve problems and make decisions? At this point, this process is not understood well enough to be automated. There are examples of automation which appear very intelligent and flexible relative to earlier automation. This has been brought about by the tremendous advances in the computer world. There is no set of algorithms which represent a taxonomy of generic human problem solving methods, although work is proceeding in this direction.

Automating a facet of life, which usually requires human functioning, generally takes a great deal of effort, time and money; however, once the automation technology is completed and is in use, it allows for faster and more economical decision making. This fast decision capability can be a critical factor in an actual, real-time situation. The economy brought about by the automation can allow NASA to pursue endeavors which might otherwise have to be delayed or reduced for economic reasons.

Automation up to a certain level permits man to operate more intelligently above that level. For example, computer graphics, databases, and subroutines for scientific calculations permit the problem solver to concentrate his efforts at a higher level in attacking a problem. The president of a corporation bases his decisions, which decide the fate of the business, on inputs to him from a select committee (expert systems). The mathematician in trying to prove a difficult theorem or uncover new results may employ a computer to generate thousands of logical deductions in different directions and to refer back to him with anything interesting or for further help in resolving a conflict.

Some basic items in future automation are: (1) how to structure knowledge in a computer; (2) how to efficiently retrieve and modify information; (3) how to formulate goals; (4) how to design a machine to learn in an unfamiliar environment; and

(5) how to discover and formulate generic concepts associated with automated decision making and problem solving.

This paper is not a compendium on automated decision making and problem solving, but is rather a presentation of some of the interesting topics being examined in this exciting and difficult area.

REFERENCES

1. Lerner, E.J.: Computers That See. In IEEE Spectrum, October 1980.

2. "Robots Ready to March into Industry." Electronic Engineering Times, April 23-30, 1979, pp 19-21.

3. Industrial Robots. Vol 2: Applications. Society of Manufacturing Engineers, 1979.

4. Albus, J.E.: People's Capitalism - The Economics of the Robot Revolution. New World Books, College Park, MD, 1976.

5. Redman, C.; Simpson, J.; and Friedrick, O.: "The Robot Revolution." Time Magazine, December 8, 1980, pp 72-83.

6. Rosen, C.A.; and Nitzan, D.: Use of Sensors in Programmable Automation. COMPUTER, December 1977, pp 12-23.

7. Duda, R.O.; Nitzan, N.J.; and Raphael, B.: State of Technology in Artificial Intelligence. In Research Directions in Software Technology, The MIT Press 1980, pp 729-749.

8. Winston, P.H.: Artificial Intelligence. Addison-Wesley, Inc., 1977.

9. Nilsson, N.J.: Principles of Artificial Intelligence. Tioga Publishing Co., 1980.

10. Automated Decision-Making and Problem Solving, Vols. I and II, NASA Conference Publication 2180, 1981.

11. Safford, E.L., Jr.: The Complete Handbook of Robotics, TAB Books no. 107 1, 1978.

12. Whitney, D.E.; Watson, P.C.; Drake, S.H.; and Simunovic, S.N.: Robot and Manipulator Control by Exteroceptive Sensors. In: Joint Automatic Control Conference, June 22-24, 1977.

13. Rosenfeld, A.: SURVEY Picture Processing: 1978. Computer Graphics and Image Processing, 9, 1978, pp 354-393.

14. Chakravarty, I.: A Survey of Current Techniques for Computer Vision. Rensselaer Polytechnic Institute Technical Report CRL-51, January 1977.

15. Vainshtein, G.G.; Zavalishin, N.V.; and Muchnik, I.E.: The Processing of Information by Robots - A Review Automation and Remote Control, vol. 35, no. 6, June 1974, pp 959-986.

16. Lerner, E.J.: Computers That See. IEEE Spectrum, October 1980.

17. Agin, G.J.: Computer Vision Systems for Industrial Inspection and Assembly. IEEE Computer, May 1980.

18. Holland, S.W.; Rossol, L.; Ward, Mitchell, R.: CONSIGHT - 1: A Vision-Controlled Robot System for Transferring Parts from Belt Conveyors General Motors Research Laboratories, GMR-2790, August 1978.

19. Shen, C.N.: Data Acquisition and Path Selection Decision Making for an Autonomous Roving Vehicle, Rensselaer Polytechnic Institute Technical Report MP-66, March 1980.

20. Yakimovsky, Y.; and Cunningham, R.: A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras. Computer Graphics and Image Processing, 7, 1978, pp 195-210.

21. Sutro, L.L.; and Larman, J.B.: Robot Vision. In: Remotely Manned Systems: Exploration and Operation in Space Proceedings of the First National Conference in Pasadena, California, Sept. 13-15, 1972, pp 251-282.

22. Shirai, Y.; and Suwa, M.: Recognition of Polyhedrons With a Range Finder. Proceedings Second International Joint Conference on Artificial Intelligence, 1971, pp 80-87.

23. Agin, G.A.; and Binford, T.O.: Computer Description of Curved Objects. Proceedings Third International Joint Conference on Artificial Intelligence, 1973, pp 624-635.

24. Popplestone, R.J.; and Ambler, A.P.: Forming Body Models from Range Data. Research Report, Department of Artificial Intelligence, University of Edinburg, 1975.

25. Will, P.M.; and Pennington, K.S.: Grid Coding: A Preprocessing Technique For Robot and Machine Vision. Artificial Intelligence, 2, 1971, pp 319-329.

26. Bales, J.W.; and Barker, L.K.: Marking Parts to Aid Robot Vision. NASA TP-1819, 1981.

27. Hunt, E.B.: Artificial Intelligence. Academic Press, Inc., 1975.

28. Barrow, H.G.; and Tenenbaum, J.M.: Interpreting Line Drawings as Three-Dimensional Surfaces. Artificial Intelligence, vol. 17, 1981.

29. Barrow, H.G.; and Tenenbaum, J.M.: Computational Vision. Proceedings of the IEEE, vol. 69, no. 5, May 1981, pp. 542-595.

30. Goksel, K.; Knowles, K.A., Jr.; Parrish, E.A., Jr.; and Moore, J.W.: An Intelligent Industrial Arm Using a Microprocessor. IEEE Transactions on Industrial Electronics and Control Instrumentation, vol. IECI-22, no. 3, August 1975, pp 309-314.

31. Conference on Automated Decision-Making and Problem Solving, Volume I: Executive Summary. May 19-20, 1980, NASA Langley Research Center.

32. Wagner, H.M.: Principles of Operations Research. Prentice-Hall, Inc., 1969.

33. Bellman, R.: Introduction to Matrix Analysis. McGraw-Hill Book Company, Inc., 1980.

34. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton, NJ, 1957.

35. Thompson, G.L.: Recent Research in Network Problems with Applications: Automated Decision Making and Problems Solving, NASA CP 2180, Volume II, 1980, p. 169.

36. Shamblin, J.E. and Stevens, G.T., Jr.: Operations Research, A Fundamental Approach. McGraw-Hill Book Company, Inc., 1974.

37. Sivazilan, B.D. and Stanfel, L.E.: Analysis of Systems in Operations Research. Prentice-Hall, Inc., Englewood Cliff, NJ, 1975.

38. White, D.; Donaldson, W. and Lawrie, N.: Operational Research Techniques. Volume 1: An Introduction. Business Books Limited London 1969.

39. Nahra, J.E.; and Odle, M.P.: A Dynamic Programming Computer Program. NASA CR-121350, 1971.

40. Duda, R.O.; Nilsson, N.J.; and Raphael, B.: "State of Technology in Artificial Intelligence, In "Research Directions in Software Technology," Edited by Peter Wegner, pp 729-749, The MIT Press, 1979.

41. Hart, P.E.: "Artificial Intelligence and National Security." SRI International Menlo Park, CA, March 1978.

42. Sacerdoti, E.D.: A Structure for Plans and Behavior. Elsevier, New York, 1977.

43. Chien, R.T.; Brew, W.; Chen, D.; and Pan, Y.C.: Artificial Intelligence and Human Error Prevention: A Computer Aided Decision Making Approach. University of Illinois Coordinated Science Laboratory Report T79, July 1979.

44. Rieger, C. and Grenberg, M.: The Causal Representation and Simulation of Physical Mechanisms. University of Maryland TR495, 1976.

45. Hewitt, C.: Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot. Ph.D. Thesis, Department of Mathematics, MIT, 1972.

46. Rulifson, J.; Derkson, J.; and Waldinger, R.: QA4: A Procedural Calculus for Intuitive Reasoning. Artificial Intelligence Technical Note 73, Stanford Research Institute, November 1972.

47. McDermott, D. and Sussman, G.: The CONNIVER Reference Manual. Artificial Intelligence Memo No 259, MIT, May 1972.

48. Fikes, R. and Nilsson, N.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, vol. 2, nos 3 and 4, 1971.

49. Fikes, R.; Hart, P.; and Nilsson, N.: Learning and Executing Generalized Robot Plans. Artificial Intelligence, vol. 3, no. 4, 1972.

50. Pease, M.C., III: ACS.1: An Experimental Automated Command Support System, IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-8, no. 10, pp 725-735, October 1978.

51. Fikes, R.E.: Monitored Execution of Robot Plans Produced by STRIPS. Information Processing 71: Proceedings of the Congress. Ljubljana, Yugoslavia August 23-28, 1971 , vol. 1, North-Holland Publishing Co., Amsterdam, 1971, pp 189-194.

52. Friedman, L.: Robot Learning and Error Correction. NASA CR-153202, 1977.

53. Prajous, R.: "Robotics Research in France." In Robotics Age, Spring 1980, pp 16-26. Human Error Prevention: A Computer Aided Decision Making Approach, University of Illinois Coordinated Science Laboratory Report T79, July 1979.

54. Prajoux, R.; Sobek, R.: Laporte, A.; and Chatila, R.: A Robot System Utilizing Task-specific Planning in a Blocks-world Assembly Experiment." Proc. 10th Int. Symp. Industrial Robots, Milan, March 1980.

55. Lee, C.Y.: An Algorithm for Path Connections and its Applications. IRE Trans. on Electronic Computers, vol. EC-10, September 1961, pp 346-365.

56. Ernst, G.W. and Newell, A.: GPS: A Case Study in Generality and Problem Solving. Academic Press, Inc., 1969.

57. Winston, P.H.: Artificial Intelligence. Addison-Wesley Publishing Co, Inc., 1979.

58. Saridis, G.N.: Self-Organizing Control of Stochastic Systems. Marcel Dekker, Inc., 1977.

59. Feigenbaum, E.A.: "Expert Systems in the 1980's." Computer Science Dept., Stanford University, Stanford, CA, 1980.

60. Davis, R.; Buchanan, B.G; and Shortliffe, E.H.: Production Rules as a Representation for a Knowledge-Based Consultation System. Artificial Intelligence, 8, pp 15-45, 1977.

61. Lindsay, R.K.; Buchanan, B.G.; Feigenbaum, E.A.; and Lederburg, J.: Applications of Artificial Intelligence to Chemistry: The DENDRAL Project. McGraw-Hill, Inc., 1980.

62. Lefler, R.M.: Automated Interpretation of 1H NMR Spectroscopy. 1977 Proceedings of Int Conf on Cybernetics and Society, pp 605-610.

63. "Heuristic Programming Project 1980." Heuristic Programming Project, Computer Science Dept, Stanford University.

64. Buchanan, B.G. and Mitchell, T.: Model Directed Learning of Production Rules. In D.A. Waterman and F. Hayes-Roth (Eds), Pattern Directed Inference Systems, Academic Press, Inc., 1978.

65. Haas, N. and Hendrix, G.G.: An Approach to Acquiring and Applying Knowledge. AAAI First Annual National Conference on Artificial Intelligence, August 1980, pp. 235-239.

66. Schank, R.C.; and the Yale AI Project: SAM - A Story Uderstander. Yale Computer Science Research Report 43, August 1975.

67. Schank, R.C. and Abelson, R.P.: Scripts, Plans, and Knowledge. Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR, 1975.

68. Lehmann, W.P.; Bennett, W.S.; Stocum, J.; Smith, H.; Pfluger, S.M.V; and Eveland, S.A.: The METAL System. RADC-TR-80-374, vol. I and II, January 1981.

69. Riegor, C.J.: An Organization of Knowledge for Problem Solving and Language Comprehension. Artificial Intelligence, vol. 7, no. 2, 1976.

70. Kraiss, F.: Decision Making and Problem Solving with Computer Assistance. NASA TM-76008, 1980. Translation of "Entscheiden und Problemlosenmit Rechnerunterstuetzung," Forschungsinstitut fuer Anthropotechnik, Meckenheim, West Germany, Report no. FB-36, February 1978, pp 1-75.

71. Fu, K.S.: Learning Control Systems - Review and Outlook. IEEE Trans on Automatic Controls, April 1970, pp 70-72.

72. Fu, K.S.: Learning Control Systems and Intelligent Control Systems: An Intersection of Artificial Intelligence and Automatic Control. IEEE Trans on Automatic Control, February 1971, pp 70-72.

73. Thau, F. and Montgomery, R.: An Adaptive-Learning Control System for Large Flexible Space Structures. JACC Joint Automatic Control Conference, San Francisco, August 13-15, 1980.

74. Saridis, G.N. and Graham, J.: Linguistic Decision Making Schemata for General Purpose Manipulators. ASME Joint Automatic Control Conference, San Francisco, August 13-15, 1980.

75. Elliott, H. and Wolovich, W.A.: A Parameter-Adaptive Control Structure for Linear Multivariable Systems. ASME Joint Automatic Control Conference, San Francisco, August 13-15, 1980.

76. Anbumani, K.; Patnaik, L.M.; and Sarma, I.G.: Multivariable Self-Tuning Control of a Distillation Column. ASME Joint Automatic Control Conference, San Francisco, August 13-15, 1980.

77. Gupta, M.M.: Fuzzy Logic Controllers. ASME Joint Automatic Control Conference San Francisco, August 13-15, 1980.

78. Cooper, D.C.: Theorem-Proving in Computers. In Advances in Programming and Non-numerical Computation (L. Fox, Ed.), Pergamon Press, 1966.

79. Hunt, E.B.: Artificial Intelligence. Academic Press, 1975.

80. Robinson, J.A.: Building Deduction Machines. In: Artificial Intelligence and Heuristic Programming (Findler, N.V.: and Meltzer, Bernard, Eds.) American Elsevier Publishing Company, 1971.

81. Minsky, M.: Steps Toward Artificial Intelligence, In Computers and Thoughts, edited by Edward A. Feigenbaum and Julian Feldman, McGraw-Hill, Inc., c. 1963, pp 406-450.

82. Bledsoe, W.W.: Splitting and Reduction Heuristics in Automatic Theorem Proving Artificial Intelligence, Vol. 2, 1971, pp 55-77.

83. Ballantyne, A.M; and Bledsoe, W.W.: Automatic Proofs of Theorems in Analysis Using Nonstandard Techniques. Journal of the Association for Computing Machinery, vol. 24, no. 3, July 1977, pp 353-374.

84. Boyer, R.S.; and Moore, J.S.: A Computational Logic. Academic Press, Inc., 1979.

85. Green, C.: Theorem-Proving by Resolution as a Basic for Question-Answering Systems. In: Machine Intelligence (Meltzer, Bernard; Michie, Donald; and Swann, Michael, Eds.) American Elsevier Publishing Co., Inc., 1969.

86. Minker, J.: An Experimental Relational Data Base System Based on Logic. In: Logic and Data Bases (H. Gallaire and J. Minker, Eds.) Plenum Press, 1978, pp 107-147.

87. Bledsoe, W.W.; and Bruell, P.: A Man-Machine Theorem-Proving System. Artificial Intelligence, vol. 5, 1974, pp 51-72.

88. Kastner, M.P.: A New Look at Large Scale Systems and Decentralized Control. Recent Graduates Speak Out. Proceedings of 1978 IEEE Conference on Decision and Control, San Diego, CA, January 1979, pp 472-473.

89. Ho, Y.: Team Decision Theory and Information Structures. Proc of IEEE, vol. 68, no. 4, June 1980, pp 644-654.

90. Ho, Y.; Kastner, M.P.; and Wong, E.: Teams, Signaling, and Information Theory. IEEE Trans on Automatic Control, vol. AC-23, no. 2, April 1978, pp 305-312.

91. Montgomery, R.C.; and Lee, P.S.: Information Distribution in Distributed Microprocessor Based Flight Control Systems. 1977 IEEE Conference on Decision and Control, New Orleans, LA, December 6-9, 1977.

92. Sandell, N.R., Jr.; Varaiya, P.; Athans, M.; and Safonov, M.G.: Survey of Decentralized Control Methods for Large Scale Systems. IEEE Trans on Automatic Control, vol. AC-23, no. 2, April 1978.

93. Walrand, J.: On Decentralized Stochastic Control. Proceedings of 1978 IEEE Conference on Decision and Control, San Diego, CA, January 1979,

94. Kabama, P.T.: An Euclidean Approach to the Problem of Parameter Reduction in Large Systems. Proceedings of Second VPI SU/AIAA Symposium on Dynamics and Control of Large Flexible Spacecraft, June 21-23, 1979, pp 459-474.

95. Greenwood, J.R.; Holloway, F.W.; Rupert, P.R.; Ozarski, R.G.; and Suski, G.J.: Hierarchically Structured Distributed Microprocessor Network for Control. Third Rocky Mountain Symposium Fort Collins, Colorado, August 19-21, 1979.

96. Heer, E., ed.: Conference on Automated Decision Making and Problem Solving, vol I - Executive Summary, vol II - Conference Presentations. NASA CP-2180, 1981.

97. Barbera, A.J.: Architecture for a Robot Hierarchical Control System. National Bureau of Standards SP 500-23, December 1977.

98. Saridis, G.N.: Intelligent Controls for Advanced Automated Processes. NASA CP-2180, 1981, pp 39-75.

99. Bobrow, D.G.; Winograd, T.: An Overview of KRL, A Knowledge Representation Language, Stanford Artificial Intelligence Laboratory Memo AIM-293, Stanford University, November 1976.

100. Chien, R.T.: Multilevel Semantic Analysis and Problem Solving in the Flight Domain. NASA CR-169282, August 1982.

101. Gallaire, H.; Minker, J.; Nicolas, J.: An Overview and Introduction to Logic and Data Bases. In: Logic and Data Bases (H. Gallaire and J. Minker, eds.), Plenum Press, 1978, pp 3-32.

102. Hayes-Roth, F.: Knowledge Representation, Organization and Control in Large Scale Pattern-Based Understanding Systems, IEEE Computer Society, June 1976.

103. Minker, J.: Logical Inference as an AID to Analysis in Large Data Bases, University of Maryland, College Park, Maryland, February 1980.

104. Riegor, C.: The Common Sense Algorithm as a Basis for Computer Models of Human Memory, Inference, Belief, and Contextual Language Comprehension. Technical Report 373, University of Maryland, College Park, Maryland, May 1975.

105. Smith, R.G.; Davis, R.: Distributed Problem Solving: The Contract Net Approach. Stanford Heuristic Programming Project, Stanford University, Stanford, CA., 1978.

106. Wolf, J.J.: Knowledge, Hypothesis, and Control in the WHIM Speech Understanding System. 1976 Joint Workshop on Pattern Recognition and Artificial Intelligence, IEEE Computer Society, June 1976.

107. Nilsson, N.J.: Artificial Intelligence. Appendix D in Machine Intelligence and Robotics: Report of the NASA Study Group. Jet Propulsion Laboratory Report 715-32, March 1980.

108. Buchanan, B.: Problem Solving With Uncertain Knowledge. Automated Decision-Making and Problem Solving, NASA CP 2180, vol. II, 1980, p. 245.

109. Sussman, G.J.; and Stallman, R.M.: Heuristic Techniques in Computer Aided Circuit Analysis. IEEE Trans. on Circuits and Systems CAS-22, no. 11, 1975.

110. Simon, H.A.: Overview of Artificial Intelligence. ONR A.I. Lecture Series Naval Research Labs, Washington, DC 1979.

111. Feigenbaum, E.A.: Expert Consulting Systems. ONR A.I. Lecture Series, George Washington University, 1980.

112. Wagner P. (Ed.) 1979. Research Directions in Software Technology. MIT Press, 1979.

113. Duda, R.O. et Al: Development of the Prospector Consultation System for Mineral Exploration. Final Report, Grant AFR 77-04499, SRI International, Menlo Park, CA, 1979.

114. Hsu, J.C.; and Meyer, A.: Modern Control Principles and Applications. McGraw-Hill, Inc., c. 1968.

115. Fogel, L.J.; Owens, A.J.; and Walsh, M.J.: Artificial Intelligence Through Simulated Evolution. John Wiley and Sons, Inc., c. 1966.

116. Burgin, G.H.: Systems Identification by Quasilinearization and by Evolutionary Programming. Journal of Cybernetics, vol. 3., no. 2, 1973, pp 56-76.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-83216		

4. Title and Subtitle	5. Report Date
Decision-Making and Problem-Solving Methods in Automation Technology	May 1983
	6. Performing Organization Code 506-54-63-01

7. Author(s)	8. Performing Organization Report No.
Walter W. Hankins Jack E. Pennington L. Keith Barker	
	10. Work Unit No.
9. Performing Organization Name and Address	
NASA Langley Research Center Hampton, VA 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This report presents a brief review of the state of the art in the automation of decision making and problem solving. The information upon which the report is based was derived from literature searches, visits to university and government laboratories performing basic research in the area, and a 1980 Langley Research Center sponsored conference on the subject. It is the contention of the authors that the technology in this area is being generated by research primarily in the three disciplines of Artificial Intelligence, Control Theory, and Operations Research. Under the assumption that the state of the art in decision making and problem solving is reflected in the problems being solved, specific problems and methods of their solution are often discussed to elucidate particular aspects of the subject. Synopses of the following major topic areas comprise most of the report: (1) detection and recognition; (2) planning; and scheduling; (3) learning; (4) theorem proving; (5) distributed systems; (6) knowledge bases; (7) search; (8) heuristics; and (9) evolutionary programming.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Artificial Intelligence Decision Making Control Theory Operations Research Problem Solving	Unclassified - Unlimited Subject Category 59

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	43	A03

For sale by the National Technical Information Service, Springfield, Virginia 22161