T. Ely

INFORMATION PROCESSING 68 - NORTH-HOLLAND PUBLISHING COMPANY - AMSTERDAM (1969)

ARTIFICIAL INTELLIGENCE: THEMES IN THE SECOND DECADE

EDWARD A. FEIGENBAUM
Computer Science Department, Stanford University, Stanford, California, USA

In this survey of artificial intelligence research, the substantive focus is heuristic programming, problems solving, and closely associated learning models. The focus in time is the period 1963-1968. Brief tours are made over a variety of topics: generality, integrated robots, game playing, theorem proving, semantic information processing, etc.

One program, which employs the heuristic search paradigm to generate explanatory hypotheses in the analysis of mass spectra of organic molecules, is described in some detail. The problem of representation for problem solving systems is discussed. Various centers of excellence in the artificial intelligence research area are mentioned. A bibliography of 76 references is given.

1. INTRODUCTION

The purpose of this talk is to survey recent literature in artificial intelligence research, and to delineate and assess trends in the research. For an infant field of research that has been growing as rapidly as this one has, with emphasis on pragmatics and techniques, without benefit of much theoretical underpinning, both the delineation and assessment present problems.

The most memorable scientific talk I ever attended was delivered entirely impromptu to an informal Stanford group by my colleague Professor Joshua Lederberg. The talk ranged over research in what might be called "RNA and DNA information processing". Though his interests range broadly, the ground he covered that day was clearly his own ground - a territory in which he has few peers.

Like a double helix, his talk had two intertwined strands. One strand carried the basic information on what experiments had been carried out and the empirical findings. The other strand consisted of Lederberg's personal scientific assessment of the quality of individual experiments and the value of the results; of judgments as to the potential fruitfulness of pursuing certain lines of endeavor, and the likely unfruitfulness of pursuing others; of an assessment of core issues needing resolution versus issues that were merely interesting but peripheral; and of many other threads of an evaluative nature. In general, this strand of the talk consisted of comments covering a broad spectrum, from those for which there was a strong scientific justification ("almost proven") to those based on the subjective and intuitive feelings that long experience in a field is supposed to give ("I have a hunch that..."). In sum, the listener was left with a mental map of the problem-experiment-theory maze that constituted the current state of this area of molecular biology research, with values for present status and futures associated with the alternate paths through the maze.

This double-stranded approach is the model I have taken for what a survey should attempt. It should be something other than a comprehensive set of pointers into the literature. Careful selection based on sometimes personal criteria of relevance and importance is essential; evaluations based on sometimes subjective criteria of plausibility and potential are useful.

This is a talk, not a book, so I cannot survey all the areas that have a rightful place under the umbrella of "artificial intelligence research". My choice of topic headings is not intended as a definition of "artificial intelligence" by implication. Vigorous subareas, with their own scientific "culture" and established publishing patterns, were left to fend for themselves. Thus, for example, the strong subarea that calls itself "pattern recognition research" was not surveyed, nor was the linguistics-translation subarea, bionics, neurophysiological information processing models, and others.

The focus of this talk is heuristic programming, problem solving, and closely associated learning models. Within the beam of this spotlight, I will concentrate on research of the period 1963-68, since I feel that the book *Computers and Thought* [22] is already an adequate reference work for the 1956-62 period. As a practical measure, I will use the abbreviation "A.I." for "artificial intelligence".

2. SOME GLOBAL CHARACTERISTICS OF THE A.I. RESEARCH ENDEAVOR

Of prime interest is the explosion of problems .cked, projects established, and reports published in the past five years. In spite of this rapid growth, quality has been maintained at a reasonably high level, in my opinion *.

From the very beginning of the A.I. research at Carnegie Tech in 1955-56 (which I regard as "The Beginning" for all practical purposes) Newell and Simon called their research "Complex Information Processing". They still do, gh many projects have been born since as "artificial intelligence" projects. In this, Newell and Simon are to be credited with considerable foresight. For A.I. research is becoming ever more enmeshed at its periphery with other areas of computer science research and application that can well be described as "complex information processing". For example, is the research on intelligent question-answering programs still to be regarded as A.I. research, or is it the natural direction for progress in the field called information retrieval research? Is the effort to develop a problem solving program to write computer operating systems [25] and A.I. effort or is it research in systems programming? Is a program [23] that forms chemical hypotheses in the analyses of mass spectra of organic molecules a piece of A.I. research, or is it chemistry?

These questions are not as trivial as the obvious "Who cares?" answer would make them seem. There is a general tendency in virtually all lines of scientific endeavor for research disciplines to fragment into specialities as the early and difficult problems become better understood and as practitioners move into the discipline to make use of the results. "Successes" are then attributed to the specialities, ignoring the contributions from the spawning discipline.

In A.I. research, examples of this process at work are numerous. Consider character recognition (i.e. what those "optical readers" do). Much of the early work on the pattern recognition problem focused on character recognition as an interesting initial task. This research, circa 1955, was motivated more by the question, "What are the interesting kinds of behavior that a computer might be made to perform, in contrast to the mundane tasks of the day (such as calculating function tables and payrolls)?" than by the question, "How can we make a machine read characters of the alphabet reliably?" [64]. The pursuit of the more general question inspired the early work on problem solving programs. Eventually the line of research turned into the applied art of designing character recognition machines and thereby, for all practical purposes, passed out of the field of A.I. research. List processing provides another example. As a body of techniques it was invented (as far as I can determine) by Newell, Shaw and Simon for handling the complex problems of memory allocation and hierarchical (and recursive) control of processing in the Logic Theory program and the earliest version of the General Problem Solver. List processing received additional refinement by another A.I. researcher, McCarthy (LISP). It underwent further change (threaded lists, knotted lists, symmetric lists, etc.) as it made the transition from "something those A.I. researchers are doing" to "software system techniques". By now list processing is an every-day working tool of a number of specialty areas, particularly compiler and operating system implementation.

Every discipline thrives on its successes, particularly in terms of attracting talented individuals and research support funds. There is a danger that the A.I. area, as the residual claimant for the problems not yet solved, the problems not yet well understood, the problems for which failure was the reward for the initial foray, will come to be viewed as the "no win" area of computer science and the home of the "pie-in-the-sky guys". There is a scattering of evidence that such a process is already at work, and I regard this as most unfortunate and undeserved.

Finally, the rapid growth of A.I. research has made the "Invisible College" of the area less viable. There is now a Special Interest Group within the ACM for Artificial Intelligence; a new journal is being prepared; and an international conference is being organized.

* Some observers have commented upon a dip in productivity in the period 1960-1963, and this appears to be documentable. I believe that this was attributable to: a shift of emphasis at some of the major centers toward difficult technological problems of tool building (e.g. MAC work at MIT, McCarthy's Time Sharing Project at Stanford); a much-needed reassessment of t' 'mplications and significance of efforts of the late s; a substantive shift of attention to problems for which a long gestation time was needed (e.g. natural language analysis, integrated robots, representation); and the establishment of academic computer science departments, programs, curricula, etc., which absorbed a significant portion of the energies of the available talent. Each of these was to have its eventual payoff in the productive 1963-1968 period.

3. THE SEARCH FOR GENERALITY

This is the title of the Newell-Ernst IFIP 65 paper, one that deserves rereading [49]. Others have since joined the quest. There appear to be two roads: the high road and the low road.

Those who walk the high road seek a generality of the total problem solving system that will allow a core of problem solving methods that are not task-specific to discover solutions in a wide variety of problem domains. Here the problem of the internal representation in terms of which the core of methods will operate is crucial. If the representation is made general enough so that each new application does not have to be tortured to fit into it, can the methods and associated processes be made general enough to cope with it without a consequent loss of problem solving power? We lack a good understanding yet of this problem of generality and representation. A view of existing problem solving programs would suggest, as common sense would also, that there is a kind of "law of nature" operating that relates problem solving generality (breadth of applicability) inversely to power (solution successes, efficiency, etc.), and power directly to specificity (task-specific information). We do not now know how to write problem solvers that will accept problems in a rather general representation at the start but then alter the representation systematically toward greater specificity and power as more problem-specific information becomes available during the problem solving. An example of this process has been worked out in detail [4], but mechanization is not in view.

The General Problem Solver traveled the high road alone for nearly a decade and established the search for generality as a viable research path. Ernst and Newell's new monograph [18], exploring the successes and problems encountered in applying GPS to a variety of task environments, appears to signal the end of the first phase of the GPS trek. More recently, GPS has acquired a life of its own, independent of its creators. For example, the GPS paradigm has appeared in a slightly more generalized form as a program called FORTRAN Deductive System [55]; and, considerably transfigured, as the Graph Traverser [14, 16]. Another GPS variant has just emerged in Sweden [63].

Travelers on the low road seek not a general problem solving system but theorems and generalizations of technique concerning underlying mechanisms that are common to a class of problem solving programs. Perhaps the best example involves the heuristic search paradigm

As it was a decade ago, the central paradigm of A.I. research is heuristic search. A tree of "tries" (aliases: subproblems, reductions, candidates, solution attempts, alternatives-and-consequences, etc.) is sprouted (or sproutable) by a generator. Solutions (variously defined) exist at particular (unknown) depths along particular (unknown) paths. To find one is a "problem". For any task regarded as nontrivial, the search space is very large. Rules and procedures called heuristics are applied to direct search, to limit search, to constrain the sprouting of the tree, etc. While some of this tree-searching machinery is entirely task-specific, other parts can be made quite general over the domain of designs employing the heuristic search paradigm. The so-called "alpha-beta" procedure is a classical example [12, 62]. Its employment is "obvious" if one is careful and thoughtful about search organization. It was employed as early as 1958 in the Newell-Shaw-Simon chess programs, it being so much a part of the underlying machinery that the employers did not consider it worthy of bringing to the attention of others.

But what is obvious to some is not obvious to others. Each new program designer, flirting with the heuristic search paradigm, should not be forced to reinvent (or what is worse pass over in ignorance) generally applicable search-facilitating procedures, particularly if the procedures are subtle.

A small, but growing, body of knowledge of this type has emerged. The work of Slagle is noteworthy, particularly his MULTIPLE [72], which is in effect a "system" of such techniques. Other papers of interest are those of Nilsson ("minimum cost paths") [51], Floyd ("nondeterministic algorithms") [24], and Golomb and Baumert ("backtrack programming") [26].

In a sense the travelers on the low road are tool builders, but their tool building is often of an abstract and elegant sort.

4. INTEGRATED ROBOTS

History will record that in 1968, in three major laboratories for A.I. research, ar
grated robot consisted of the followir
a) a complex receptor (typicall
camera of some sort) se
nals to...
b) a computer of consider
memory; a variety of p
lyzing the afferent video

decisions relating to the effectual movement of....

c) a mechanical arm-and-hand manipulator or a motor-driven cart.

The intensive effort being invested on the development of computer-controlled hand-eye and eye-cart devices is for me the most unexpected occurrence in A.I. research in the 1963-68 period.

Research of this type began effectively with Ernst's thesis on a computer-controlled mechanical hand (MH-1) [19]. He wrote interesting heuristic programs for solving problems of manual manipulation in a real environment. MH-1 was almost totally "blind", but it did store a symbolic internal representation of the external situation (a "model") in terms of which it did its problem solving. The seminal piece of research for the visual ("eye") processing was the oft-cited thesis of Roberts [59] on the three-dimensional perception of solids from two-dimensional picture input.

The three current robot projects are direct descendents. They are: the Stanford Hand-Eye Project (McCarthy et al.), the MIT Hand-Eye Project (Minsky and Papert), and the Stanford Research Institute's Robot Project (Nilsson, Raphael, Rosen et al.).

Not much information about these projects has been published. Hence, what follows is to some extent anecdotal.

As one might expect, the design, implementation, and use of the robot hardware presents some difficult, and often expensive, engineering and maintenance problems. If one is to work in this area solving such problems is a necessary prelude but, more often than not, unrewarding because the activity does not address the questions of A.I. research that motivate the project.

Why, then, build devices? Why not simulate them and their environments? In fact, the SRI group has done good work in simulating a version of their robot in a simplified environment. So it can be done and the questions raised above are relevant.

The answer given is as follows. It is felt by the SRI group that the most unsatisfactory part of their simulation effort was the simulation of the environment. Yet, they say that 90% of the effort of the simulation team went into this part of the simulation. It turned out to be very difficult to reproduce in an internal representation for a computer the necessary richness of environment that would give rise to interesting behavior by the highly adaptive robot. **It is easier and cheaper to build a hardware robot to extract what information it needs from the real world than to organize and store a useful model.** Crudely put, the SRI group's argument is that the most economic and efficient store of information about the real world is the real world itself.

The task of building an integrated robot is one, I believe, that contains the possibility of studying some problems of major interest in artificial intelligence research, among which are: strategy formation and planning; the problems of representing situations for problem solving processes and subsequent modification of representations as new information becomes available; and visual perceptual processing. Of the three groups, only the SRI group has published papers discussing the more general aspects and goals of this research [58, 61].

Both the MIT and Stanford University groups have worked on programs for controlling a variety of arm-hand manipulators, from the very simple to the very complex, from the anthropomorphic variety to the very non-anthropomorphic. None of the more esoteric manipulators seems to have worked out very well, though there is no published documentation of successes, failures, and reasons.

Visual scene analysis programs are important in all of these projects. Most of the programming effort is being invested to build the proper tools and techniques to gain control of this piece of the task. The scene analysis problem is this: the TV image of a scene (digitized) is available to be read into the computer memory; scan and process it as necessary to produce a symbolic description of the objects in the scene and their various interrelationships. Guzman [30] at MIT attacked the scene analysis task in a somewhat abstracted form (no TV camera, a no-noise symbolic simulation of the input scene) with striking success. Guzman's work, incidentally, should be of interest to psychologists working on models of human visual perception processes.

The name of the game however is integrated behavior on a problem. Both the MIT and Stanford University hand-eye-computer aggregates have performed a few types of block-finding and block-stacking behaviors. A paper in the IFIP 68 Proceedings describes the Stanford work [54]; I have not found a paper describing the MIT block-stacking activity.

So you want to build an integrated robot? Wait! The three lively groups, whose levels of talent and funding are hard to match, have not yet uncovered all the first-level problems.

These will be found, reported, and assessed, perhaps within the next two years. The projects are still very much in the tool-building and debugging stage. Whether the integrated robot is a useful and appropriate task for making progress on the general problems of A.I. research remains to be proven.

5. THEOREM PROVING

Since Robinson is presenting an invited survey paper on automatic theorem proving at this congress, it would be inappropriate for me to survey this literature here. But perhaps a few comments and a few citations are in order.

Many in the field persist in calling their theorem proving programs *deduction* programs (thus, the aforementioned FORTRAN *Deductive* System, DEDUCOM [41], the term "deductive question-answering programs"; Hunt's survey [32] contains numerous instances of this cause). This is a terminological mistake. If one looks carefully at how these programs find their proofs, much more than deduction in the strict sense is involved. When a theorem proving program applies a rule of inference, say *modus ponens*, in taking a trial step forward, it is clearly making an elementary deduction. But the search for a proof is not a deduction. In practice, the terminological error has tended to inhibit clear thinking on key problems, for example those involved in the attempt to unify parts of the field (like heuristic search) by a careful examination of the basic mechanisms used in a variety of successful programs. The usage I am vilifying is not harmless because it tends to sort the work of the field into the wrong categories.

I prefer Amarel's term, "problems of derivation type", as correct, clear, and meaningful. I feel the term "discovery processes" is an appropriate and useful one for describing the processes by which the proof of a theorem (or the move from a chess position, or the chemical hypothesis that explains a mass spectrum, etc.) is found.

Robinson's resolution method for theorem proving in the predicate calculus has received much attention in the past few years. Unfortunately this has been accompanied by a sentiment that the resolution method "liberates" one from the guessy, messy, chaotic world of heuristic search. Again a false distinction, based on unclear understanding either of resolution or of existing heuristic search proof-finding programs or both, sorts the world along the wrong lines. The resolution method does provide a systematic formal mechanism for guaranteeing the completeness of the space of proof candidates, but it does not by itself deal with the well-known and inevitable problem of the proliferation of problem states [60]. Thus search strategies have been overlaid to bring about effective problem solving using resolution (e.g. "unit preference", "set of support"). The net result is that the processes these programs carry out are much the same as (in some cases, identical to) those carried out by the heuristic search proof-finding programs that are thought to be so different [18]. I predict that much more will be heard on this issue in the second decade.

In 1959, McCarthy proposed a class of programs (advice-takers) that would reason about the world in a common-sense way. The "world" was to be given a homogeneous representation in the predicate calculus. Problems which presented themselves would be solved as proofs over the space of expressions. Recently, Green and Raphael [27] have incorporated the machinery of the resolution method as the proof-finding "engine" in an advice-taker-like question-answering and fact retrieval program. The idea is important (and works), but as just mentioned the necessary heuristic overlay to guide the search effort will have to be provided if the system is to be useful and practical.

6. GAME PLAYING PROGRAMS

In the first decade, there were those who wrote chess playing programs because chess provided an interesting and complex task environment in which to study problem solving processes (the capstone of this line of research is a gem of a paper by Newell and Simon [50] examining in great detail an example of human chess play in the light of what we have come to understand about problem solving processes in chess from building chess playing programs). There were others who wrote chess programs because the activity presented such a challenge: chess is THE great centuries-old human intellectual diversion.

Such a group are Greenblatt et al. They have written the first program that plays very good (but not yet expert) chess. I have seen only one paper on the Greenblatt program [28]. Along with a brief description, it gives some examples of the program's play. Competing against humans under ordinary tournament rules, it is

said to have won a Class D tournament in Boston in mid-1967, reportedly beating a Class C player in the process. It is also reported to be much better by now. In the exhibit area of the 1967 FJCC, it took on all comers, and playing with its "hack" settings (quick play) it still did a creditable job, winning more games than it lost. Apparently, its most celebrated victory was a handy win over Hubert Dreyfus.

Why does it play so well as compared with previous chess programs? I do not know anyone who yet has a convincing answer to this (partially because of the paucity of information about the program). As I view it, the program embodies no fundamentally new ideas about how to organize chess playing programs. It contains much more specific information about chess play than any previous program had. Computer time, top programmer talent, fancy tools of the second decade (CRT displays, interactive access, big core memory) - the patient has received an order of magnitude more loving care than any other previous patient (all other patients, you will remember, were released in a weak condition; most died). Finally, an excellent "learning loop" is available - through a human at the console. Blunders are analyzed, and quickly fixed with patches and/or new chess knowledge. The system-wide effects of the patch or the new knowledge, if any, can be fairly quickly detected and revised if causing problems. It is a feasible way to improve (or educate) a program, and a useful one if you are interested in a high level of performance in a specific task rather than in general models for the organization of problem solving. I foresee this technique, albeit in more sophisticated forms, being widely adopted in the second decade.

The first international tournament between chess playing programs was won by a program developed in the Soviet Union at the Institute for Theoretical and Applied Physics in Moscow (by Adelson-Velskii et al.; no writeup available). The loser was the old MIT program, developed by Kotok [38] and slightly modified by McCarthy's group at Stanford. The play is available for inspection in the SICART Bulletin [57]. Neither program played well when compared with the average level of performance of the Greenblatt program.

Samuel's well-publicized checker playing program has undergone extensive revision [62], and now stands near the top of its profession. The major revisions are in the area of the learning routines, and will be discussed later.

Williams [76] has attacked the problem of modeling the generality with which human players approach common board and card games. His program, General Game Playing Program, is given as input a description of the objects used in playing the game; and the rules of the game taken from *Hoyle's Rules of Games*, transformed straightforwardly into a Hoyle-like input language. The program will then play at least a legal game, for most of the games described by *Hoyle*.

7. MACHINE LEARNING (SPECIFICALLY, INTERNAL MECHANISMS; NOT HUMAN-DIRECTED)

The A.I. field still has little grasp of the machine learning problem for problem solvers. For many years, almost the only citation worth making was to Samual's famed checker playing program and its learning system. (Great interest arose once in a scheme proposed by Newell, Shaw, and Simon for learning in GPS, but the scheme was never realized). Surprisingly, today we face the same situation.

Samuel's new paper [62] describes a major revision of the position evaluation scheme of the checker player and its attendent learning processes. Evaluation using the linear polynomial function is abandoned in favor of complex nonlinear evaluation processes. The features of positions, which used to be represented as terms in the polynomial, are now grouped according to their (perceived) interdependencies. "Crude" values for the features are used, e.g. 3-,5-, or 7-valued functions. (This is an old idea, whose rationale was given as far back as 1951 by Simon for chess-playing; it is used in the Newell-Shaw-Simon Chess Playing Program). Vectors of feature values are used to enter "signature tables" at the first level. A table-lookup takes place; the resulting table value is quantized into "crude" states (five-valued) and passed up to a second level of "signature table" aggregation (over a set of first level signature tables). This process is repeated once again at a third level of aggregation, and from this "top" signature table a final score for the evaluation emerges. Samuel shows the considerable performance advantages of this hierarchical scheme over the (already quite successful) linear polynomial evaluation.

Samuel titles one of his sections, "The Heuristic Search for Heuristics", and maintains therein "that the task of making decisions as to the heuristics to be used is also a problem which can only be attacked by heuristic procedures, since it is essentially an even more complicated

task than is the paying itself". An interesting case study of the learning of heuristics by a heuristic program is emerging in the dissertation research of Waterman at Stanford [74], nearly complete. The task environment is draw poker. The heuristics are not cast as programs in the traditional mode, but are "brought to the surface" as situation-action rules in a "production" list. Initially the list contains only the null rule: whatever the situation, play randomly. The program is trained by any one of the following techniques: accepting advice from a human as to the quality of its particular moves; similarly, accepting advice from an "expert" program, which is in essence a theory of good poker play (Waterman's theory, of course); or by just playing poker, for better or for worse, and inferring the quality of its particular moves from what happens (which is the slowest method of training the program). Basically, four things can happen to the table of rules. Situation-action rules can be added. The order of the rules can be altered (since the table is scanned in a top-to-bottom fixed order, this can make a big difference in behavior). A rule can be "generalized" by altering its situation-side so as to ignore certain dimensions of the game situation, thereby causing it to "catch" more situations. Or the situation-side can be "specialized" so as to be more discriminating among game situations and hence "catch" fewer situations. This learning scheme works well; it leads to good poker playing performance by the program. The program should be of great interest to cognitive psychologists, particularly S-R learning theorists, since it is an operational S-R learning model that is performing and learning in a complex problem-solving task.

8. A NOTE IN PASSING: TURNING INWARD TO THE PROGRAMMING TASK ITSELF

In the checker-playing program, Samuel used a "rote memory" learning scheme, in which many checker board positions were stored ever along with their scores from previous look-ahead search. If a "memorized" position were encountered in a new analysis, the value was available and would not have to be recomputed. Theorem proving programs use "rote memory" for analogous purposes in building their theorem memories. So do many other programs, e.g. Heuristic Dendral, described later.

The issue of store versus recompute is quite general and classical. One looks forward to the day when the programming system one is using is smart enough to figure out when it should assign function values by table lookup in a rote memory it builds, and when by computation (it would make this decision by an analysis of the uses of and performance characteristics of the function as it encounters this information during execution). A first step in this direction has recently been made with the introduction of "memo functions" into the Edinburgh POP-2 language [41]. With memo functions, however, the programmer still has decisions to make, and I look forward to a further "Samuelization" of programming language systems for the store versus recompute decision.

Simon [65] once wrote a problem-solving program (Heuristic Compiler) that solved problems of writing simple programs in IPL-V, given descriptions of how the programs were supposed to alter the state of the IPL-V machine. The program was organized as a simplified version of GPS, and was successful, but it has not been followed up with a more substantial effort.

9. SEMANTIC INFORMATION PROCESSING AND NATURAL LANGUAGE

This area of research is of great importance to the A.I. endeavor. Its importance arises not so much from the presumed practical advantages of being able to converse in natural language with a program that "understands", but because the research raises and focuses certain key issues that are quite general. Research of high quality has been done in the past few years. A book [43] covering much of it will be available shortly. Minsky's introduction to the book should be consulted for an extended treatment of the subject, which is impossible here. Parts of a paper by Coles [15] also give a good treatment. Nevertheless, a few comments may be useful.

The research grapples in various ways with the problem of the meaning of ordinary natural language utterances and the computer understanding of the meaning of these utterances as evidenced by its subsequent linguistic, problem-solving, or question-answering behavior. Meaning is viewed not as something one "puts into" a program (for example, ordinary dictionary entries help very little) but as an *emergent* from the interplay of syntactic analyzers, models that link real-world objects and relations, appropriate data structures that link together symbols of the internal-world, a logical calculus and associated discovery processes for solving deriva-

tion-type problems. (This list is not necessarily exhaustive).

Syntactic analysis, which has received massive attention from the computational linguists (with elegant results), is not enough to handle problems of meaning and understanding *. Consider the following example.

Bobrow's STUDENT [9] is a problem solver that accepts natural language input (English sentences). High-school-level algebra word problems constitute the domain of discourse. The sentences are simplified and parsed; idioms are transformed. Reference to the real-world is made through a table of global relations. Typically the amount of global information made available to STUDENT has been quite small; hence, STUDENT's understanding of the algebra word problems is largely "syntactic". The appropriate simultaneous algebraic equations are set up and solved for the answer.

STUDENT can be made to solve the following problem: "A board was sawed into two pieces. One piece was two-thirds as long as the whole board and was exceeded in length by the second piece by four feet. How long was the board before it was cut?" STUDENT will show that in one sense it understands this problem by issuing the correct solution, that the board length was minus twelve feet. In the psychological experiments done by Paige and Simon in the light of STUDENT [52], some subjects solved the problem just as STUDENT solved it (with a focus on the syntax leading to the correct equation system), but others immediately recognized its physical impossibility and refused to set up the equations. These people were the model builders, who attached the "givens" to the appropriate model of the physical situation, immediately noticing the misfit**. They were exhibiting another level of "understanding of the problem" not available to STUDENT.

The notion of interpretation of natural language in terms of stored models is central to much of the research on semantic information processing. Raphael's Semantic Information Retrieval question-answering system [56] uses a node-link relational model (with a restricted set of relations) to organize its data universe. This model is grown to incorporate new information about objects and their relationships that is extracted from a simple analysis of declarative sentences typed in at the console by the user. Other programs use internal representations of two-dimensional pictures as a model. Coles [14], extending the work of Kirsch et al. [34] on the Picture Language Machine, wrote a program that uses a picture (input by the user at a CRT with a light pen) to resolve syntactic ambiguities in English sentences about the picture, and to answer questions about the picture.

Pushing the subject of models and data structures a bit further, consider restructuring a traditional dictionary into a "semantic graph" in the following way. Represent entities as symbols at the nodes, and the various general and special relations between entities as associative links between the nodes. An entity's node is the start of a subgraph which, when traced out, encompasses the various "meanings" of the entity in terms of other entities and relations in the semantic graph. Quillian [55] has constructed such a graph for a small (but definitely nontrivial) set of dictionary definitions, and a processor for performing a variety of associative and semantic reference tasks. Quillian's program, I believe, is a good foundation for further research on models of human associative memory. This possibility is worth a vigorous pursuit.

10. SIMULATION OF COGNITIVE PROCESSES

Space limitations here preclude a survey of the work in this interesting territory at the intersection of computer science and psychology; the formulation and validation of information processing theories of human problem solving and learning processes using primarily, but not exclusively, the techniques of heuristic programming and the methodology of computer simulation. Fortunately, two thorough reviews [32] have appeared recently and are recommended. Nevertheless, I cannot resist giving my own personal set of pointers into the literature.

Problem solving: Newell [44, 47, 48].
Analysis of human behavior in crypto-arithmetic puzzle solving tasks; major methodological advances in analysis of human problem solving "think-aloud" protocols and the study of human eye movements during problem solving.

Verbal learning and memory: Simon and Feigenbaum [68]; Gregg and Simon [29]; Feigenbaum [21]; Hintzman [31].
Further results with Elementary Perceiver and

* Overemphasis on syntax analysis at the expense of research on semantic processes has hindered the development of the mechanical translation area, I believe.

** My own informal replications of the experiment at Stanford led me to believe that the effect is independent of the brilliance, or lack of it, of the subject, but a function of whether he is a "visualizer" or a "symbolizer". STUDENT, of course, is a "symbolizer".

Memorizer (EPAM) model; reinterpretation in terms of theory of various levels of human memory; extensions by Hintzman to handle additional phenomena.

Concept Learning and Pattern Induction: Hunt, Marin, and Stone [33] report many experiments with Concept Learning System (CLS).
Simon and Kotovsky [69]; Simon and Sumner [70]. Simple, elegant program that handles sequence completion tasks from standard intelligence test and also can infer and extrapolate patterns in melodies.

Affect, emotion, beliefs:
Tessler, Enea, and Colby [73]; Abelson and Carroll [2]. Models of human belief systems and their use in studying neurotic and "normal" behavior.
Simon [66]. Emotional and motivational concomitants to cognitive processes.

Judgment: Kleinmuntz [35, 36].
Model of the clinician's judgmental process in constructing personality profile from subject's answers in Minnesota Multiphasic Personality Inventory; and studies of other clinicians' tasks.

11. THE SHAPE OF THE FIELD, 1968: BY EXAMINATION IN DEPTH OF ONE PROGRAM

In attempting to characterize state-of-the-art in a field, a careful look at one research effort is often revealing. As with an etching, the lines of the work - lines of ideas, of methods and techniques, of technology employed - are seen with greater clarity from close by.

I have chosen for the close-up the research work with which I am personally involved. It is not only more appropriate for me to do this than to sketch another's work, but also the sketch carries with it an assured knowledge of detail. Most important, the research lies squarely in what I consider to be the mainstream of the A.I. research endeavor: problem solving using the heuristic search paradigm.

Our primary goal was to study processes of hypothesis formation in a complex task of a scientific nature involving the analysis of empirical data. The task environment chosen as the medium for this study was the analysis of the mass spectra of organic molecules: the generation of a hypothesis to best explain given mass spectral data. This is a relatively new area of organic chemistry of great interest to physical chemists. In this sense, the problem is not a "toy" problem; and a program that solves problems of this type is a useful application of A.I. research to a problem of importance to science.

We have written a program to infer structural hypotheses from mass spectral data. The program is called Heuristic Dendral. It was developed at the Stanford University Artificial Intelligence Project by a small group including Professor Joshua Lederberg of the Stanford Genetics Department, Dr. Bruce Buchanan, Mrs. Georgia Sutherland, and me, with the assistance of chemists and mass spectrometrists of the Stanford Chemistry Department. It is an 80,000 word program written in LISP for the PDP-6 computer, and was developed (and is run) interactively under the time-sharing monitor [10, 23, 40].

Heuristic Dendral will perform the following two classes of tasks:

1. Given the mass spectrum of an organic molecular sample and the chemical formula of the molecule, the program will produce a short list of molecular "graphs" as hypotheses to explain the given data in the light of the program's models of mass spectrometric processes and stability of organic molecules. The list is rank-ordered from the most satisfactory explanation to the least satisfactory.
2. If no mass spectrum is given, but only a formula, the program will produce a list of all the chemically plausible isomers of the molecule in the light of its model of chemical stability of organic molecules.

The flow diagram of the system is a closed loop consisting of phases of data inspection, hypothesis generation, prediction, and test, corresponding closely to a simple "scientific method" loop.

At the heart of the program is a systematic hypothesis generator. It is based on an algorithm developed by Lederberg called DENDRAL, which is capable of generating all of the topologically possible isomers of a chemical formula. The generator is essentially a topologist, knowing nothing about chemistry except for the valences of atoms, but the generating algorithm serves as the guarantor of the completeness of the hypothesis space, in a fashion analogous to the legal move generator in a chess program. Since the generating process is a combinatorial procedure, it produces for all but the simplest molecules a very large set of structures, almost all of which are chemically implausible though topologically possible. Implicit in its activity is a tree of possible hypothesis candidates. At the top node of the tree all the atoms are found but no structures. At the terminal nodes, only complete structures are found, but no unallocated atoms. Each intermediate node specifies a partially built structure and a residual set of atoms yet to be allocated. This tree is the implicit problem space for Heuristic Dendral. Various

heuristic rules and chemical models are employed to control the generation of paths through this space, as follows:

. A model of the chemical stability of organic molecules based on the presence of certain denied and preferred subgraphs of the chemical graph. It is called the *a priori* model since it is independent of processes of mass spectrometry.
2. A very crude but efficient theory of the behavior of molecules in a mass spectrometer, called the Zero-order Theory of Mass Spectrometry, used to make a rough initial discarding of whole classes of structures because they are not valid in the light of the data, even to a crude approximation.
3. A set of pattern recognition heuristic rules which allow a preliminary interpretation of the data in terms of the presence of key functional groups, absence of other functional groups, weights of radicals attached to key functional groups, etc. It is called the Preliminary Inference Maker. Its activity allows the Hypothesis Generator to proceed directly to the most plausible subtrees of the space.

The output of the Preliminary Inference and Hypothesis Generation processes is a list of molecular structures that are candidate hypotheses for an explanation of the mass spectrum. They are all chemically plausible under the *a priori* theory and valid explanations of the data under our zero-order theory of mass spectrometry. Typically the list contains a few candidates (but not dozens or hundreds).

Next a confrontation is made between this list of "most likely" hypotheses and the data. For each candidate hypothesis, a detailed prediction is made of its mass spectrum. This is done with a subprogram called the Predictor, a complex theory of mass spectrometry in computer simulation form. The Predictor is not a heuristic program. It is an elaborate but straightforward procedure for deducing consequences of a theory of mass spectrometry extracted by us from chemists and their literature. The spectral prediction for each candidate is matched with the empirical input data by a process called the Evaluation Function. This is a heuristic, hierarchical, non-linear scoring procedure. Some hypothesis candidates are immediately discarded because their predicted spectra fail certain critical confrontations. The remainder are scored, ranked, and printed out in rank order from most to least satisfactory.

For the class of non-ringed organic structures with which we have been working up to the present time, the program's behavior approaches or exceeds the performance of post-doctoral laboratory workers in mass spectrometry for certain classes of organic molecules. These include amino acids, with which for tangential reasons we have done much of our work, and a large variety of simple organic groups which, however, turn out to be considerably more complicated than amino acids from the point of view of mass spectrometry.

Heuristic programming provided only the skeleton for the problem solving of Heuristic Dendral and the computer techniques to handle the implementation. The heuristics of chemical plausibility of structures; of preliminary inference; of evaluation of the predictions; and also the zero-order and complex theories of mass spectrometry, these were all extracted from our chemist colleagues by man-machine interaction, with the program carefully guided by one of our research team. This mixed discipline for pulling out of the heads of practicing professionals the problem solving heuristics they are using has worked far better than we had any right to expect, and we are now considering further mechanization of this process.

12. THE PROBLEM OF REPRESENTATION FOR PROBLEM SOLVING SYSTEMS

A.I. research in the remainder of the second decade will be dominated by a few key problems of general importance. The problem of representation for problem solving systems is one of these, and in my view the most important, though not the most immediately tractable *.

In heuristic problem solving programs, the search for solutions within a problem space is conducted and controlled by heuristic rules. The representation that defines the problem space is the problem solver's "way of looking at" the problem and also specifies the form of solutions. Choosing a representation that is right for a problem can improve spectacularly the efficiency of the solution-finding process. The choice of problem representation is the job of the human programmer and is a creative act. Amarel [6] believes that the process of choosing and shaping appropriate representations for problem solving

* I have used the term "representation *for problem solving systems*" to distinguish this from the much more widely discussed data representation (and data structures) problem. I believe that we will find eventually that the two sets of questions have an important intersection, but for the moment it is best to avoid terminological confusion.

is the essence of the behavior in humans that we call "creative". I agree.

Some examples of the impact of choice of representation on problem solving performance have been discussed in the literature. The classic is the so-called "though nut" proposed by McCarthy and discussed by Newell [46]. Mutilate a chess board by removing two corner squares diagonally opposed; can the mutilated board be covered by dominos? If the standard piece board-move game playing representation is employed, an enormous and almost impossible search would have to be conducted to discover that no covering solution was possible. But a choice of problem representation involving the concepts of parity of red-black covering by a domino and of counting of red and black squares leads immediately to the solution that no covering is possible because two squares of the same color are removed in the mutilation.

Another example of much greater complexity has been worked out by Amarel for the traditional puzzle of transporting missionaries and cannibals from one bank of a river to the other with a boat under certain constraints [4]. Amarel exhibits a succession of representational shifts for this problem, from the one usually used, to a simple but elegant matrix-like representation, in terms of which the solution is available almost immediately by inspection. Amarel has worked out still another example for theorem proving in the propositional calculus [5]. In fact, as early as 1958, Gelernter in the Geometry Machine used a diagram as an auxiliary problem representation to improve the efficiency of searching the problem-subproblem tree.

Until very recently the problem of representation has been treated in the literature by exploring a few examples in detail. Fortunately, a new paper by Amarel [7], offering a synthesis of his view of problem solving and representation, gives a clear formulation and extended discussion of this difficult area.

Why is it that a shift of problem representation can lead to a spectacular change in problem solving effectiveness? There are many reasons; here are a few. Each problem representation has associated with it a set of specialized methods for manipulating elements of the representation. Shifting to a representation that is "rich" in specialized theory and methods from one that is impoverished in this regard allows the power of the former to be applied to the problem at hand. Similarly, specialized relationships associated with an appropriate representation can be imported into the (often incomplete) statement of a problem, thereby supplying missing but crucial augmentations to the problem definition. An example of this has been exhibited by Paige and Simon [52] for alcohol-water mixture problems (the appropriate representation supplies necessary conservation equations). Finally, each representation can have associated with it a data base of descriptions and facts that become available for incorporation into the problem statement or for use in controlling search.

Amarel has discussed the mechanization of the process of shift of representation as a step-by-step process, involving an "evolution" of each representation to a somewhat more powerful one [4, 5]. The evolution is guided by information about the problem (or problem class) that turns up during the problem solving activity within a particular representation of the moment. The alternative to this step-by-step process is a generator of new representations as trial candidates in a heuristic search for an appropriate representation. Design of such a generator is a formidable task at this early stage in our understanding of the representation problem. The simplest design, however, is to generate the elements of a stored repertoire of previously encountered or potentially useful representations. Such a design was employed in a program by Persson [53] for the problem of choosing the appropriate representation of pattern in a mixture of different sequence extrapolation tasks.

In my view, the use of the concept of analogy between problems is a crucial step in the design of a generator of representations. Candidates for an appropriate problem representation are searched for, discovered, and tried, by a search process that uses analogical reasoning over a store of known representations (and their associated methods, data base, etc.). Problem solving search using reasoning-by-analogy has received surprisingly little attention in A.I. research, considering the importance of the problem. The work by Evans [20] on a program to solve "intelligence test" problems involving geometrical analogies is the only work I can cite.

Of necessity, these comments on the problem of representation have been sketchy in the extreme. But because of the central importance of this problem, I felt the need to focus attention on it. I believe that the long gestation period for this problem is ending; that the time is ripe for a major push; that there will be important developments on this front in the next five years;

and that these will come to be viewed as having the same degree of centrality and importance that now attaches itself to the heuristic search paradigm.

13. CENTERS OF EXCELLENCE

It is conventional to survey research by topic and unconventional to survey it by places and people. Yet I am frequently being asked in conversation some form of the question: "In artificial intelligence (particularly heuristic programming), where is the action?" There is no reason for not attempting to answer this question in a public forum.

The reference point in time is mid-1968. The emphasis is on a substantial *quantity* of *high quality* research (my assessment).

In the United States, the three major research centers are the A.I. projects at MIT, Carnegie-Mellon University (née Carnegie Tech), and Stanford University. All three receive the major portion of their support from the Advanced Research Projects Agency (Department of Defense). All three train substantial numbers of Ph.D. students in the A.I. area. The MIT and Stanford projects use dedicated PDP-6 computers with big core memories; the Carnegie project uses an IBM 360/67 with a very large extended core memory.

At MIT, the more senior faculty and research principals are Minsky, Papert, and to some extent, Weizenbaum [75]; at Carnegie, Newell and Simon; at Stanford, McCarthy, Samuel, Colby, and Feigenbaum. The Stanford group has close ties with the neighboring SRI group (Nilsson and Raphael); similarly the MIT group has close ties with Bobrow's group at Bolt, Beranek, and Newman.

Citing some statistics from the Stanford University group, which I obviously know best, there are 75 people (faculty, students, and staff) associated with the A.I. project; about 25 different research projects under way; and a working paper series of 62 papers. I offer these figures to indicate scale of effort at a major center.

Five other centers deserving attention are: Case Western Reserve University (Banerji, Ernst); University of Wisconsin (Travis, Uhr, London); RCA Laboratories at Princeton (Amarel); Heuristics Laboratory, National Institutes of Health (Slagle); and the University of Washington (Hunt).

In Europe, no centers comparable to the major American centers were visible in the first decade. In the past few years, however, a center of the first rank has arisen at the University of Edinburgh, and other centers are emerging in Sweden and the Soviet Union.

At Edinburgh, A.I. research is enshrined in a Department of Machine Intelligence and Perception. The principals are Michie, Gregory, Burstall, Doran and Popplestone. They are supported reasonably generously by the British Government. The research ranges very broadly from the various projects in information processing models of cognition and perception to applications of these models [11, 12] and development of programming languages [13]. The latest "score sheet" from the Department gives bibliographic data for 59 "original research contributions" since 1965! This group is responsible for a major series of collected papers, the *Machine Intelligence* series.

At Uppsala University in Sweden, the Department of Computer Sciences is doing A.I. research (GPS, planning, robot simulations, LISP work). Principal activist is Sandewall, formerly at Stanford. Psychologists interested in simulation of cognitive processes are participating. The Swedish Natural Science Research Council supports the research.

When I cast my mind's eye as far off as the Soviet Union, the image becomes fuzzy, though I make a determined effort to keep current with Soviet computer science literature (particularly the area of discussion, heuristic programming). The few papers I have seen are motivation-suppliers or clarifications at points of contact with philosophy, psychology, and neurophysiology. However, there are talented groups at various locations that are interested (and perhaps actively working) in the area. These are:

Institute of Cybernetics, Ukrainian Academy of Sciences, Kiev (Glushkov, Amosov and coworkers);
Institute of Automation and Remote Control, Moscow (Aizerman's Laboratory);
Moscow State University, Dept. of Higher Nervous Activity (Napalkov);
Computer Center, Siberian Division, Academy of Sciences of USSR, Novosibirsk (Yershov, Marchuk and coworkers).

All of these are major centers, interested generally in the problems of A.I. research. Whether the developers of the chess program mentioned earlier, at the Institute of Theoretical and Applied Physics in Moscow, are interested in problems other than chess program development I do not know.

A Russian translation of *Computers and Thought*, edited by Napalkov and Orfyeev, appeared last year and was an immediate sell-out. A Scientific-Technical Commision on Heuristic Programming has come into existence within the last two years.

A computer scientist, writing in *Izvestiya*, claims "solid successes of Soviet heuristic programming" in pattern recognition, chess, and theorem proving, but cites lags in "breadth of the work being done" and in "equipping these projects with computer hardware" [39]. It would be useful for the A.I. field to have a survey paper by a Soviet computer scientist on Soviet work in heuristic programming.

ACKNOWLEDGEMENTS

Support for the preparation of this paper was provided by the Advanced Research Projects Agency, Department of Defense (Contract SD-183), the Stanford Artificial Intelligence Project, and the Stanford Computation Center. I also wish to acknowledge with gratitude the comments of Allen Newell and Georgia Sutherland during the final stages of preparing the talk and the manuscript.

REFERENCES

The list of references given below represents a personal list relevant to the particular purposes of this survey article. It is not intended as a comprehensive bibliography of the field. A comprehensive bibliography of the literature through 1962 is given in ref. [22]. No such bibliography yet exists for the period 1963-1968. The Special Interest Group for Artificial Intelligence of the ACM is attempting currently to build one.

[1] R. Abelson, Computer simulation of social behavior, in: Handbook of Social Psychology, ed. G. Lindzey, 2nd ed. (Addison-Wesley, New York, 1968).

[2] R. Abelson and J. Carroll, Computer simulation of individual belief systems, Amer. Behav. Sci. 8 (1965).

[3] S. Amarel, On representations and modeling in problem solving and on future directions for intelligent systems, Scientific Report, RCA Laboratories, Princeton, New Jersey (1968).

[4] S. Amarel, On representations of problems of reasoning about actions, in: Machine Intelligence 3, ed. D. Michie (Edinburgh Univ. Press, 1968).

[5] S. Amarel, An approach to heuristic problem solving and theorem proving in the propositional calculus, in: Systems and Computer Science, eds. J. F. Hart and S. Takasu (Univ. of Toronto Press, 1967).

[6] S. Amarel, On the mechanization of creative processes, IEEE Spectrum (1966).

[7] S. Amarel, On the representation of problems and goal-directed procedures for computers, Working Paper Computer Theory Research Group, RCA Laboratories, Princeton, New Jersey, Proc. First Annual Symp. Amer. Soc. for Cybernetics (to be published).

[8] S. Amarel, Problem solving procedures for efficie syntactic analysis, Scientific Report, RCA Labora tories, Princeton, New Jersey (1968).

[9] D. G. Bobrow, A question answering system for hi school algebra word problems, Proc. FJCC 25 (1964).

[10] B. Buchanan and G. Sutherland, Heuristic dendral: program for generating explanatory hypotheses in organic chemistry, Artificial Intelligence Workin Paper No. 62, Computer Sci. Dept., Stanford Univ in: Machine Intelligence 4, eds. D. Michie et al. (be published).

[11] R. M. Burstall, A heuristic method for a job sche uling problem, Operations Res. Quart. 17 (1966).

[12] R. M. Burstall, Computer design for electricity supply networks by heuristic method, Computer J. 9 (1966) 3.

[13] R. M. Burstall and R. J. Popplestone. POP-2 reference manual, in: Machine Intelligence 2, eds. E. Dale and D. Michie (Oliver and Boyd, Edinburg 1968), also in POP-2 Papers (Oliver and Boyd, Edinburgh, 1968).

[14] S. L. Coles, Syntax directed interpretation of natu ral language, Ph.D. dissertation, Computer Sci. Department, Carnegie-Mellon Univ. (1967).

[15] J. D. Darlington, Machine methods for proving lo ical arguments expressed in english, Mechan. Transl. 8 (1965) 3.

[16] J. Doran, New developments of the graph travers in: Machine Intelligence 2, eds. F. Dale and D. Michie (Oliver and Boyd, Edinburgh, 1966).

[17] J. E. Doran and D. Michie, Experiments with the graph traverser program, Proc. Roy. Soc. A 29 (1966).

[18] G. Ernst and A. Newell, GPS: A case study in ge erality and problem solving, ACM Monograph Series (in press).

[19] H. A. Ernst, MH-1. A computer-operated mech ical hand, Ph.D. dissertation, Massachusetts I of Technology, 1961, presented at the WJCC, 1962.

[20] T. G. Evans, A heuristic program to solve geon ric analogy problems, Proc. SJCC 25 (1964).

[21] E. A. Feigenbaum, Information processing and memory, Proc. Fifth Berkeley Symp. on Mathe ical Statistics and Probability, vol. 4, Biology Problems of Health (Univ. of California Press, 1967).

[22] E. Feigenbaum and J. Feldman, eds., Compute and Thought (McGraw-Hill, 1963).

[23] E. A. Feigenbaum, J. Lederberg and R. Buchan Heuristic Dendral, Proc. Hawaii Intern. Conf. System Sciences, Univ. of Hawaii and IEEE (U of Hawaii Press, 1968).

[24] R. W. Floyd, Non-deterministic algorithms, J. Assoc. Computing Mach. 14 (1967) 4.

[25] P. Freeman, Ph.D. dissertation research, Ca Mellon Univ. (1968).

[26] S. W. Golomb and L. D. Baumert, Backtrack pr gramming, J. Assoc. Computing Mach. 12 (19

[27] C.C.Green and R.Raphael, Research on intelligent question answering system, Scientific Report AFCRL-67-0370, Artificial Intelligence Project, Stanford Res. Inst., Menlo Park, Calif. (1967).
[28] G.Greenblatt, D.Eastlake and S.Crocker, The Greenblatt chess program, Proc. Fall Joint Computer Conf., Anaheim, Calif. (1967).
[29] L.W.Gregg and H.A.Simon, An information-processing explanation of one-trial and incremental learning, J. Verbal Learning Verbal Behavior 6 (1967) 5.
[30] A.Guzman, Some aspects of pattern recognition by computer, MIT Masters thesis, Report No. MAC-TR-37, Project MAC, MIT (1967).
[31] D.L.Hintzman, Exploration for a discrimination net model for paired associate learning, J. Math. Psychol. 5 (1968) 1.
[32] E.Hunt, Computer simulation: Artificial intelligence studies and their relevance to psychology, Ann. Rev. Psychol. (1968).
[33] E.Hunt, J.Marin and F.Stone, Experiments in induction (Academic Press, 1966).
[34] R.A.Kirsch, Computer interpretation of english text and picture patterns, IEEF Transaction on Electronic Computers, Vol. EC-13 (1964).
[35] B.Kleinmuntz, Profile analysis revisited: A heuristic approach, J. Counsel. Psychol. 10 (1963).
[36] B.Kleinmuntz, The processing of clinical information by man and machine, in: Formal Representation of Human Judgment, ed. B.Kleinmuntz (Wiley, New York, 1968).
[37] M.Kochen, On the representation of limited information by means of pictures, trees, and English-like sentences, Working Paper, Computer Theory Research Group, RCA Laboratories, Princeton, New Jersey (1965).
[38] A.Kotok, A chess playing program for the IBM 7090, unpublished B.S. thesis (MIT, 1962).
[39] A.Kronrod, The computer becomes 'more intelligent', in: Izvestiya (15 March 1967). Translated in: Soviet Cybernetics: Recent News Items, No. 3, ed. W.Holland (The RAND Corporation, Santa Monica, Calif., 1967).
[40] J.Lederberg and E.Feigenbaum, Mechanization of inductive inference in organic chemistry, in: Formal Representation of Human Judgment, ed. B.Kleinmuntz (John Wiley, New York, 1968).
[41] D.Michie, Memo functions and machine learning, Nature 218 (1968).
[42] M.L.Minsky, Artificial intelligence, in: Information (Freeman, 1966).
[43] M.L.Minsky, ed., Semantic information processing (MIT Press, 1968, to be published).
[44] A.Newell, Eye movements and problem solving, Computer Sci. Res. Rev. (Carnegie-Mellon Univ., Pittsburgh, Pennsylvania, Dec. 1967).
[45] A.Newell, Heuristic programming: Ill structured problems, Working Paper, Computer Sci. Dept., Carnegie-Mellon Univ., 1967, in: Progress in Operations Research (in press).
[46] A.Newell, Limitations of the current stock of ideas about problem solving, in: Electronic information handling, eds. A.Kent and O.E.Taulbee (Spartan, 1965).
[47] A.Newell, On the analysis of human problem solving protocols, Working Paper, Computer Sci. Dept., Carnegie-Mellon Univ., July 1967.
[48] A.Newell, Studies in problem solving: Subject 3 on the crypt-arithmetic task Donald + Gerald = Robert, Working Paper, Computer Sci. Dept., Carnegie-Mellon Univ., July 1967.
[49] A.Newell and G.Ernst, The search for generality, Proc. IFIP 65 Congress (Spartan, New York, 1965).
[50] A.Newell and H.A.Simon, An example of human chess play in the light of chess playing programs, Working Paper, Computer Sci. Dept., Carnegie-Mellon Univ., August 1964, in: Progress in Neurocybernetics, eds. J.Schade and N.Wiener (Elsevier, Amsterdam, 1965).
[51] N.J.Nilsson, A new method for searching problem-solving and game playing trees, Working Paper, Artificial Intelligence Group, Stanford Res. Inst., Menlo Park, Calif., Nov. 1967. Paper presented at the IFIP 68 Congress, Edinburgh.
[52] J.Paige and H.A.Simon, Cognitive processes in solving algebra word problems, in: Problem Solving: Research, Method and Theory, ed. B.Kleinmuntz, Proc. First Carnegie Symp. on Cognition (Wiley, New York, 1966).
[53] S.Persson, Some sequence extrapolating programs: A study of representation and modeling in inquiring systems, Report CS50, Computer Sci. Department, Stanford Univ., Sept. 1966.
[54] K.K.Pingle, J.A.Singer and W.M.Wichman, Computer control of a mechanical arm through visual input, paper presented at IFIP 68 Congress, Edinburgh.
[55] J.R.Quillian and E.B.Hunt, The FORTRAN deductive system, Techn. Report 68-1-01, Dept. of Psychology, Univ. of Washington, Seattle, Wash., Jan. 1968.
[56] B.Raphael, A computer program which 'understands', Proc. FJCC 25 (1964).
[58] B.Raphael, Programming a robot, Working Paper, Artificial Intelligence Project, Stanford Res. Inst., Menlo Park, Calif., Paper presented at the IFIP 68 Congress, Edinburgh.
[59] L.G.Roberts, Machine perception of three dimensional solids, in: Optical and Electro-optical Processing of Information (MIT Press, Cambridge, Mass., 1965).
[60] J.A.Robinson, Heuristic and complete processes in the mechanization of theorem proving, in: Systems and Computer Science, eds. J.F.Hart and S.Takasu (Univ. of Toronto Press, 1967).
[61] C.A.Rosen and N.J.Nilsson, An intelligent automation, Working Paper, Artificial Intelligence Group, Stanford Res. Inst., Menlo Park, Calif. Paper presented at the 1967 IEEE International Convention, New York.
[62] A.Samuel, Studies in machine learning using the game of checkers 2. Recent progress, IBM Journal (Nov. 1967). To be reprinted in: Annual Review of Automatic Programming, ed. M.Halpern (Pergamon Press, to be published).
[63] E.Sandewall, The GPS expressed in LISP-A, Working Paper, Dept. of Computer Sciences, Uppsala, Oct. 1967.
[64] O.G.Selfridge, Pattern recognition and modern computers, Proc. 1955 Western Joint Computer Conf. (1955).
[65] H.A.Simon, Experiments with a heuristic compiler, J. Assoc. Computing Mach. 10 (1963) 4.

[66] H.A. Simon, Motivational and emotional controls of cognition, Psych. Rev. (1967).
[67] H.A. Simon, Scientific discovery and the psychology of problem solving, in: Mind and Cosmos, ed. R.G. Colodny (Univ. of Pittsburgh Press, 1966).
[68] H.A. Simon and E.A. Feigenbaum, An information-processing theory of some effects of similarity, familiarization, and meaningfulness in verbal learning, J. Verbal Learning Verbal Behavior 3 (1964).
[69] H.A. Simon and K. Kotovsky, Human acquisition of concepts for sequential patterns, Psychol. Rev. 70 (1963).
[70] H.A. Simon and R.K. Sumner, Pattern in Music, in: Formal Representation of Human Judgment, ed. B. Kleinmuntz (Wiley, 1968).
[71] J. Slagle, Experiments with a deductive question answering program, Commun. ACM 8 (1965) 12.
[72] J.R. Slagle and P. Bursky, Experiments with a multi-purpose, theorem proving heuristic program, J. Assoc. Computing Mach. 15 (1968) 1.
[73] L. Tessler, H. Enea and K.M. Colby, A directed graph representation for computer simulation of belief systems, Math. Biosci. 2 (1968).
[74] D. Waterman, Machine learning of heuristics, Ph.D. dissertation (to be published), Artificial Intelligence Project, Stanford Univ., 1968.
[75] J. Weizenbaum, Contextual understanding by computers, in: Recognizing Patterns: Studies in Living and Automatic Systems, eds. P.A. Kolers and M. Eden (MIT Press, 1968).
[76] P.G. Williams, Some studies in game playing with a digital computer, Ph.D. dissertation, Computer Sci. Dept., Carnegie-Mellon Univ., July 1965.

DISCUSSION

Question by A. P. Ershov

If someone wants to enter the artificial intelligence field, would you advise him to construct an integrated robot, or to construct a system with a human in the central role and programs and software taking on the role of his faculties?

Answer

That would depend on the character of the organisation attempting to enter the field. In some instances, the time is ripe to exploit the heuristic search paradigm. For other organisations, which lean to research rather than development, there are a few key problems: 1) The representation problem, 2) Reasoning by analogy, 3) Notions of planning, 4) Extensions of work on machines that understand, with bigger data bases.

Wait two years before beginning another robot and let the present groups dig out the first-level problems.

Question

Do you believe that a computer will become world chess champion?

Answer

Maybe someone in the audience can conjecture on that.

Remark by R. S. Scowen

I am sure a computer will be programmed to play chess better than the world champion. One feature of the Greenblatt program is that it considers each position on its merit and never uses any previous analysis; this is both a virtue and a fault. It is good because the computer gets neither downhearted when losing nor over-confident when winning. It is bad because the computer never forms a long term strategy. When playing in tournament mode ($2\frac{1}{2}$ minutes per move) I would guess that the strength of the program is such that the computer is almost worthy of a place in an English County team.

Remark by J. McCarthy

The ex-world champion has written a book on how computers should play chess. Perhaps your suggestions will be realised.

Question by J. Sammet

With the increasing advent of formula manipulation languages, is it likely that mathematics will become a fertile area for artificial intelligence?

Answer

I made an oversight in neglecting to mention MATHLAB. I do consider such work to be artificial intelligence.

Question by C. Tully

How do you see A.I. work linking up with work in the commercial field?

Answer

There has recently been some correspondence in SICART on A.I. in management. Several years ago F. Younge had a heuristic program for ma-

chine assembly and gave impetus to many similar operations research type applications. An article by G. Clarkson in "Computers and Thought" deals with decisions in constructing trust investment portfolios, trying to model the behaviour of investment offices rather than construct optimal portfolios.

INVITED COMMENTARY

SAUL AMAREL
RCA Laboratories, Princeton, N.J., USA

I am in strong agreement with Feigenbaum's assessment of the present state of the art in A.I. and also with his views about the trends of work in this area and the nature of research problems that are now central and deserve more of our attention. I would like to add a few comments that are mainly intended to emphasize some of the points made by Feigenbaum.

It is important to realize that there is no sharp dividing line between the procedures of A.I. and the procedures of today's conventional software - both systems software and applications software. The procedures that are being developed and used for systems software strongly resemble in their overall logical structure - and also in their detail - many of the heuristic procedures that are studied in A.I. research. Also there are many application packages in engineering, science, and management that are constructed on the basis of a combination of systematic and heuristic methods. Today's outstanding example of the application of A.I. ideas to an important "real life" problem is Feigenbaum and Lederberg's computer-based system for spectrometry. It is becoming increasingly clear that the advanced procedures of A.I. and the procedures that direct today's "useful" work of computers lie on some sort of continuum. The key variables in this continuum are the amount of systematic knowledge available about a problem class, and the degree to which this knowledge can be efficiently exploited for the solution of specific problems in the class; the latter depends on the *form* of the available knowledge. At the one end of the continuum where amount of formal knowledge and its grade of utilization are high we have most of today's "conventional" programs. At the other end, we have the procedures of A.I., where a set of relatively weak problem-specific principles are combined with several general heuristic methods for organizing search processes.

One of the important goals in the development of problem solving procedures in A.I. is to enable a user to specify directly his *problem* to a computer without having to specify to the computer an explicit procedure for solving it. This possibility would be a major step in the road towards programming automation. However, even in this case the man is left with the responsibility of formulating his problem to the computer in a manner which promises a reasonably efficient solution-finding process. Here we are confronted with the *problem of problem representation*, which, I fully agree with Feigenbaum, is today's central problem of A.I. In representing a problem to a machine the man provides at present all the knowledge about the problem that the machine can work with, and also he provides it in a specific form which reflects his specific point of view - a point of view which may or may not be fruitful for the solution-searching process that he is forcing upon the machine. The question arises naturally whether it is possible to endow machines with capabilities to shift problem representations in an "appropriate" direction. I have been concerned with this problem in the last few years, and I feel at present that in order to realize beneficial shifts in problem representation we need to know more about (i) how to choose the basic concepts for languages in which problems can be formulated, and (ii) how to proceed in the discovery of useful properties of a problem space that can be used to transform it into a space where search for solution is less difficult.

As a last comment I would like to indicate that most of the progress (in technique and theoretical understanding) in heuristic problem solving to date has centered on problems of derivation type, where the objective is roughly to construct a path between given boundaries (e.g., theorem proving problems). Problems of formation type have received so far less attention. These problems

are more difficult than derivation problems, and they involve reasoning from possible solutions to the problem conditions. Many "real life" problems, notably design problems and diagnostic problems, are of this type. Feigenbaum's spectrometry problem is largely a formation problem. Also, many problems of shift in problem representations are of formation type. I think that as we move more and more into useful applications of A.I. to complex problems, and as we attempt to attain more problem solving generality via computer handling of problem representations, we shall have to do much more work on formation problems.

INVITED COMMENTARY

D. MICHIE

Department of Machine Intelligence and Perception, University of Edinburgh, Scotland

After such an impressive survey of the field there is little that I can add. I would like to say of the Stanford Heuristic Dendral project that it succeeds in illustrating all but one of the key themes of artificial intelligence research. Before referring to the missing theme, let me list the others.

1. Use of the full repertoire of heuristic search and evaluation techniques applied to tree representations, originally developed in the early game-playing studies.
2. Complete "Baconian cycle" - observation, inductive hypothesis formation, deduction from trial hypotheses to yield predictions, comparison of observation with prediction, revision of hypothesis (this last facility exists in Dendral, but so far no case has arisen calling for a second traversal of the Baconian loop).
3. Interactive use of the console in problem-solving, especially for digging heuristics out of skilled humans for incorporation in the program.
4. Actual usefulness of the program, potentially on a large economic scale. Artificial intelligence research is leaving the era to toy problems.

The theme which is missing is what Newell has termed the "search for generality". Heuristic Dendral is task-restricted. The work has produced an intelligent automatic chemist, but not a library of LISP functions which we can use for other tasks. This point is relevant to Machine Intelligence goals as we tend to see them in our Edinburgh group. Our feeling is that intelligent computing systems will emerge through building up a library of non-numerical functions, for rote-learning, for tree-searching, for deductive logic, etc. in a suitably general language system. The historical analogy is with standard numerical functions. The time taken for the user to solve complicated numerical problems has been reduced out of all recognition by the building of powerful integrated libraries of mathematical functions, from floating-point multiplication through matrix packages and minimization procedures. In the same spirit, if the generalizable core of the Stanford work can be embedded as standard functions in their on-line system over the next few years, then when in 1978 someone decides to write a program called Son of Dendral he should find it the work of weeks or days rather than months. Certainly these are the kinds of hopes that we have of the future development of our POP-2 system, designed as a tool of artificial intelligence research by Rod Burstall and Robin Popplestone under the direct inspiration of LISP and CPL.

The long-term validation of the concepts of artificial intelligence research should be in terms of utility. The idea that artificial intelligence research may be the "no win" field of computer science has been referred to - if you start winning then what you are doing is given another name. Yes and no. No, because there is a hard core of work, growing in importance, devoted to building the tool-kit of the "epistemological engineer". In this category we include not only the whole apparatus of heuristic tree-search already referred to, but work of a more theoretical character such as Amarel's studies of the representation problem, and approaches by McCarthy to the construction of formal systems able to handle the concepts of common-sense human reasoning ("can", "should", "know", etc.). I believe that such studies will lead to the design and testing of systems exhibiting the full Baconian cycle of Dendral and possessing the generality which it lacks.