Stanford Artificial Intelligence Laboratory
Memo AIM-252

Computer Science Department
Report No. STAN-CS-74-466

July 1974

RECENT RESEARCH IN ARTIFICIAL INTELLIGENCE, HEURISTIC PROGRAMMING, AND NETWORK PROTOCOLS

Edited by
Lester Earnest

ARTIFICIAL INTELLIGENCE PROJECT
John McCarthy, Principal Investigator

HEURISTIC PROGRAMMING PROJECT
Edward Feigenbaum and Joshua Lederberg,
Co-principal Investigators

NETWORK PROTOCOL DEVELOPMENT PROJECT
Vinton Cerf, Principal Investigator

Sponsored by
ADVANCED RESEARCH PROJECTS AGENCY
ARPA Order No. 2494

COMPUTER SCIENCE DEPARTMENT
Stanford University

Stanford Artificial Intelligence Laboratory
Memo AIM-252

July 1974

Computer Science Department
Report No. STAN-CS-74-466

RECENT RESEARCH IN ARTIFICIAL INTELLIGENCE, HEURISTIC PROGRAMMING, AND NETWORK PROTOCOLS

Edited by
Lester Earnest

ARTIFICIAL INTELLIGENCE PROJECT
John McCarthy, Principal Investigator

HEURISTIC PROGRAMMING PROJECT
Edward Feigenbaum and Joshua Lederberg,
Co-principal Investigators

NETWORK PROTOCOL DEVELOPMENT PROJECT
Vinton Cerf, Principal Investigator

ABSTRACT

This . is a progress report for ARPA-sponsored research projects in computer science for the period July 1973 to July 19'73. Accomplishments are reported in artificial intelligence (especially heuristic programming, robotics, theorem proving, automatic programming, and natural language understanding), mathematical theory of computation, and protocol development for computer communication networks. References to recent publications are provided for each topic.

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC 15-73-C-0435. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.

TABLE OF CONTENTS

1. INTRODUCTION

This is a report of accomplishments by three ongoing projects that have been supported by the Advanced Research Projects Agency (ARPA) in the period July 1973 to July 1974. Some related research supported by other agencies (mainly NSF, NASA, NIH, and NIMH) is also discussed. Where not otherwise stated, the work reported below was supported by ARPA.

The Artificial Intelligence Project is the oldest and largest of the activities treated here. It was organized by John McCarthy, Professor of Computer Science, in 1963 and has received ARPA support continuously since then. It has Included work in computer vision, robotics, mathematical theory of computation, theorem proving, speech recognition, natural language understanding, programming language development, and a number of other activities. ARPA budgeted $1.25 million in support of this work for the year of this report.

The Heuristic Programming Project was formed in 1965 by Edward Feigenbaum, Professor of Computer Science, and Joshua Lederberg, Professor of Genetics, and was initially an element of the Artificial Intelligence Project. It became a separate organizational entity with its own budget in January 1970. The central interest of this project has been artificial intelligence applied to scientific endeavor and the problems of knowledge acquisition, representation, and use that arise in constructing high-performance applications of AI. ARPA support for the year amounted to $200K.

The Network Protocol Development Project was formed in July 1973 by Vinton Cerf, Assistant Professor of Computer Science and Electrical Engineering, and has been concerned with communication protocols for computer networks, especially the ARPA network. ARPA support to this activity was $50K for the year.

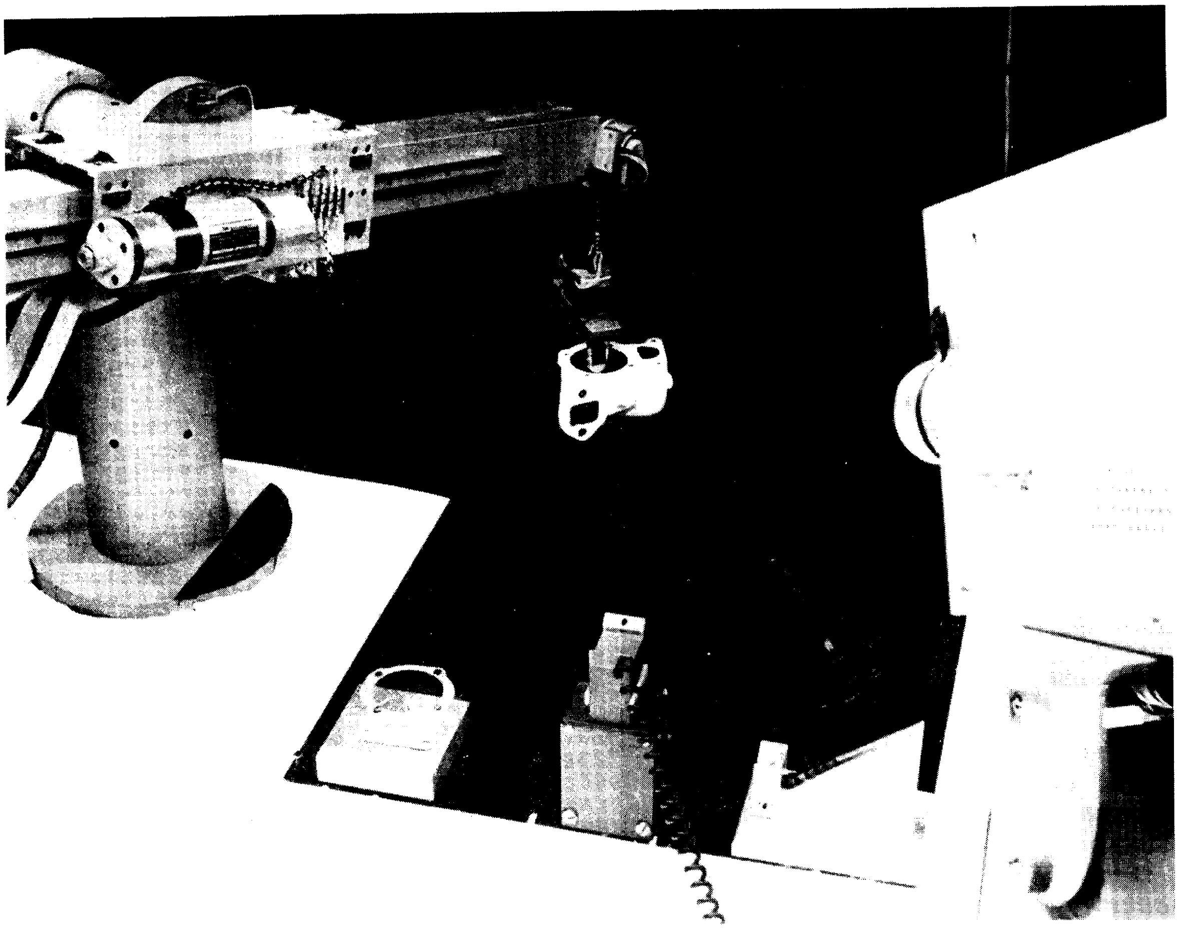
This report updates and builds upon our ten year report [11. Like most progress reports, it is mainly a concatenation of segments written by the individuals who did the work. Consequently, there are substantial variations in style and depth of treatment.

The following sections summarize recent accomplishments and provide bibliographies in each area. Appendlces list theses, films, books, articles, and reports produced by our staff.

Bibliography

[1] Lester Earnest (ed.), FINAL REPORT: The First Ten Years of Artificial Intelligence Research at Stanford, Stanford A. I. Memo AIM-228, July 1973.

INTRODUCTION



Stanford Arm Assembling Water Pump

2. ARTIFICIAL INTELLIGENCE PROJECT

The work of the Artificial Intelligence Project has been basic and applied research in artificial intelligence and related fields, such as mathematical theory of computation. Here is a short list of what we consider to have been our main accomplishments during the past year.

Robotics

We have devrlopcd a two-arm synchronized manipulation capability and tested it on several mechanical assembly tasks that are beyond the capability of a single arm. A new high-level "hand language" called HAL has been developed for specifying advanced manipulation tasks.

Computer Vision

We have used near and far field stereo visron and motion parallax to locate objects spatially and to automatically generate contour maps. Another program can recognize things of the complexity of a doll or a hammer in various positions, using a laser triangulation system.

Mathematical Theory of Computation

Using out LCF proof-checker, we have produced an axiomatization of the programming language PASCAL. This represents a major step toward using LCF as a practical program verification system.

Theorem Proving

An interactive system has been developed for structured top-down programming in PASCAL. It guides the user in constructing a program in successive refinements and in proving its correctness.

Automatic Programming

A successful new automatic programming system accepts descriptions of library routines, programming methods, and program specifications in a high level semantic definition language. It returns programs written in a subset of ALGOL that satisfy the given specifications. Experimental applications include computing arithmetical functions and planning robot strategies.

Another system works with algorithms expressed in a higher-level language and automatically chooses an efficient representation for the data structure. It then produces a program that uses this representation. Representations considered include certain kinds of linked-lists, binary trees, and hash tables.

Natural Language Understanding

A system called MARGIE was completed that links natural language understanding, inference, and generation of natural language output. This is the first such system with a strong theoretical basis, namely *conceptual dependency*.

Training

During the year, six members of our staff published Ph.D. dissertations and another 32 graduate students received direct support.

The following sections review principal activities, with references to published articles and books.

2.1 Robotics

A group led by Jerry Feldman, Tom Binford, and Lou Paul has been developing automatic assembly techniques using general purpose manipulators, visual representation and descriptive techniques using television camera and other sensory data. The vision work is covered in Section 2.2.

2.1.1 Manipulation

The robotics group has established leadership in manipulation and notes particularly advances in two arm synchronized manipulation, and in design of a new hand language for manipulation.

The completion last year of automated assembly of a water pump by Paul and Bolles [Bolles] marked a change in direction of manipulator research. In the previous phase of system building, Paul [Paul] had developed software for control of the Scheinman arm [Scheinman] using touch (one switch per finger wrth 10 gram sensitivity) and force (measured from joint motor currents, about 350 gram sensitivity). The pump assembly task showed the use of touch, force, tools, and vision in a complete system task. The new emphasis has been on application of the system to programming of repetitive assembly tasks, and executing tasks chosen to develop new manipulation abilities.

Our conception of the assembly task as a planning task carried out once on a large system, and a small repetitive execution task seems suited to industrial assembly. The plan can be intelligently tailored to the individual task; the small system repeatedly executes a plan and modifies the plan at runtime to take into account part to part variations.

In order to provide this runtime modification it was necessary to move the arm solution routines from the planning stage to the runtime system and to communicate positions in terms of rectangular table coordinates Instead of in terms of arm joint angles. By communicating part positions in terms of rectangular coordinates it was possible to 'translate and rotate sets of positions as necessary to adapt the manipulator to each actual part and its relative manipulator placement.

When this work was completed, Bolles programmed automatic assembly of the piston-crankshaft subassembly of a two-stroke gasoline engine, allowing considerable variation in work piece positioning. Bolles programmed tool changing. The arm automatically removes one tool and mounts another from a set of socket tools for an automatic tool driver. A second arm was interfaced to our computer. Paul developed a system in which two arms could be run in synchronization. He programmed assembly of a hinge, using two arms. (These examples are recorded in a film [3]). To explore arm dynamics, Finkel programmed throwing objects into bins sorted by size.

These tasks were carried out in the hand language WAVE [Paul]. Programming in a hand language gave a generality which might be described as: given that we had carried out one task, programming assembly of a similar object would be simple. For example, programming assembly of a generator would require about 10 hours work, following the pump assembly. A set of macros were developed which were applicable to a variety of tasks: put a pm in a hole, insert a screw. About 8 hours are required to program a macro of the complexity of Inserting a screw. The hand language has made it simple to teach students how to program the arm. In a robotics course, students programmed "sword in stone", inserting a shaft into a hole. The language WAVE is on the level of assembler code.

In order to take advantage of more than one manipulator, it is necessary that the manipulators can be run simultaneously, either performing independent subtasks or acting together. The existing language, Wave, was not designed to incorporate parallel operation, and was Inadequate to take advantage of the runtime modification feature already described. A new language was needed to specify structures and attachments of parts, and to provide a suitable syntax in which to express parallel operation. It was also necessary to incorporate general expression evaluation, including matrices and vectors. A new language design was undertaken to incorporate these features. Superficially, it resembles Algol, as it provides for structured programming and block structure.

HAL

Paul, Finkel, Bolles and Taylor have begun a new hand language, HAL. The effort began as a higher level language version of WAVE, to Include coordinated motion of two arms. The design was broadened to Include some strategy generation. The system is made partially model-based (versus procedure-based as WAVE, ALGOL, etc). Some degree of automatic generation of sequences for non-independent operations whose order is important, has been included in the design. In order to carry on the next stage of generalization beyond WAVE, the system must maintain models of its world.

Consider modifying the pump assembly program to assemble a generator. An expert programmer is needed to modify the program, while for a model-based system, the engineer could input a new model (presumably from a previous design) and allow the system to do the low level interfacing. The system could not perform that Interfacing if given only low level trajectory commands. We regard this system as a first level of model-based system. A major technical advance will be coordinated two arm motion, as opposed to independent two arm motion. An Important part of the design process has been to express a number of tasks (gedanken experiments) in the new language. The design of HAL was three-fourths completed during the period of this progress report.

New work has gone into touch sensor development. A new sensor with adequate sensitivity and small size was built and tested by-. Perkins. The sensor seems adequate for use in task execution, but requires more development in packaging and mounting on usable fingers.

A new collision avoidance package has been programmed by Widdoes. The package finds a collision-free path for the first three joints of the arm, using an interesting strategy. The previous collision avoidance program [Pieper] used a very local search around objects and was very time-consuming.

During the period covered in this report, we have begun conversion of arm hardware to a PDP-11/45 and begun converting to a new hand/eye table which allows room for two arm manipulation.

Bibliography

[Bolles] Robert Bolles, Richard Paul, The use of **Sensory** Feedback **in** a Programmable **Assembly System**, Stanford A. I. Memo AIM-220, October 1973.

[Dobrotin] Dobrotin, Boris M., Victor D. Scheinman, **Design** of a Computer **Controlled** Manipulator for Robot Research, *Proc. Third ht. Joint Conf. on Artificial Intelligence*, Stanford U., 1973.

[Paul] R. Paul, **Modelling**, Trajectory **Calculation** and **Servoing** of a Computer Controlled **Arm**, Stanford A. I. Memo AIM-177, *Ph.D. Thesis in Computer Science,* September 1972.

[Pieper] Donald L. Pieper, The Kinematics of **Manipulators** under **Computer** Control, Stanford A. I. Memo AIM-'72, October 1968.

[Scheinman] V. D. Scheinman, Design of a **Computer** Manipulator, Stanford A. I. Memo AIM-92, June 1969.

Films

[1] Richard Paul and Karl Pingle, Instant Insanity, 16mm color, silent 6 mm, August 1971.

[2] Richard Paul and Karl Pingle **Automated Pump** Assembly, 16mm color, Silent, 7 min, April 1973.

[3] Pingle, Paul and Bolles, **Automated** Assembly! Three Short Examples, 1974 (forthcoming).

2.1.2 Assembly Strategies

Our work at programming manipulation sequences applies to programming the class of programmable assembly devices. The goal is to program at the level of assembly instruction manuals: insert shaft B into hole C. That is to go from a high level program to a program in terms of device motions, including force and control information. There is an enormous scope for such applications; the ease of programming assembly devices is crucial to their wide application, from high volume special automation to low volume general purpose devices. The effort has produced outline programs for assembly of the water pump (without manipulation programming) by Taylor, and by Luckham [Luckham]. A typical sequencing task is to choose a sequence which does not involve putting down the tool and picking it up again in pulling out the guide pins and inserting screws to fasten the pump cover. As another facet of compatible sequences, semantic constraints such as x on y are translated into mathematical constraints; Taylor has programmed a linear constraint solution package to solve the resulting mathematical conditions.

Bibliography

[Luckham] David Luckham, Jack Buchanan, Automatic Generation of Programs Containing **Conditional** Statements, *Proc. A.I.S.B. Summer Conference,* Sussex, England, July 1974.

2.2 Computer Vision

The theme of our work in visual perception of complex objects has been *description and not classification.* We have concentrated on building up capabilities for generating structured descriptions useful in a rich universe of objects.

2.2.1 **Description**

This project has been extended by Nevatia [Nevatia], with programs which recognize objects of the complexity of a doll, glove, toy horse, or hammer. The work has included new, stable hardware for the laser triangulation system [Agin]. The programs use depth data from the laser system, find boundaries of continuous surfaces, and make descriptions of armlike parts according to a representation based on **generalized** cones [Binford]. Other groups have begun to use special cases of such representations. The programs then make complete structured descriptions of objects as a part/whole graph of parts and relations of parts at joints.

Compact summary descriptions are abstracted from complete object descriptions and used to index into a structured visual memory of models of previously seen objects to locate a subclass of model similar to the test object. The index procedure limits comparison of the test object description to relatively few models from memory, even with a large visual memory of many objects (although only about six objects were used). Models in memory were descriptions made by the program of previously seen objects, sometimes modified by hand. An important feature of the description matching process is that it depends on generating structured symbolic descriptions of differences between test object description and the model.

The descriptions themselves are intuitively familiar for humans, so that the decisions of the program are easy to understand and debug. Although a great deal more work is necessary for that system, it represents a first and significant step in such description and recognition, particularly since it can tolerate moderate obscuration. The same techniques are applicable to edge images from TV data; they give good descriptive ability for that domain. However, that is only a small part of the necessary system for analysis of TV images and although useful, in no way resembles a solution to that complex problem.

Stereo Vision

It is difficult for humans to perform manipulation tasks from a single TV image, without stereo. We intend to make considerable use of stereo in applications of vehicle navigation and visual feedback for assembly. Hannah [Hannah] has demonstrated some techniques of stereo matching for stereo pairs with moderate stereo angle, 10 degrees, without calibration information. By matching a minimum of six points in the two images, it was possible to obtain the relative calibration of the two cameras. Further search was limited by calibration information. Techniques were developed to match corresponding areas in outdoor pictures from features including color, mean and variance of intensity. A program was able to define regions bounded by depth discontinuities.

Motion Parallax

Thomas and Pingle [Thomas] have applied motion parallax to simple scenes. They limit attention to a few points defined by variance or edge measures. These points are tracked as the scene is rotated on a turntable, equivalent to moving the camera. The program requires only about 1 second per frame, using the SPS-41 computer. Although the research was performed on scenes of blocks, it is not limited to such scenes. The mechanism for selecting points of interest would be Inadequate for scenes with texture, however. Ganapathy has developed techniques for wide angle stereo matching in scenes of blocks. These programs use a variety of conditions of the form that planes remain planar under matching.

Bullock [Bullock] has made a systematic study of available operators for description of texture, and made a library of standard textures. His informal conclusions are that spatial domain descriptions are necessary, and that known techniques are very weak.

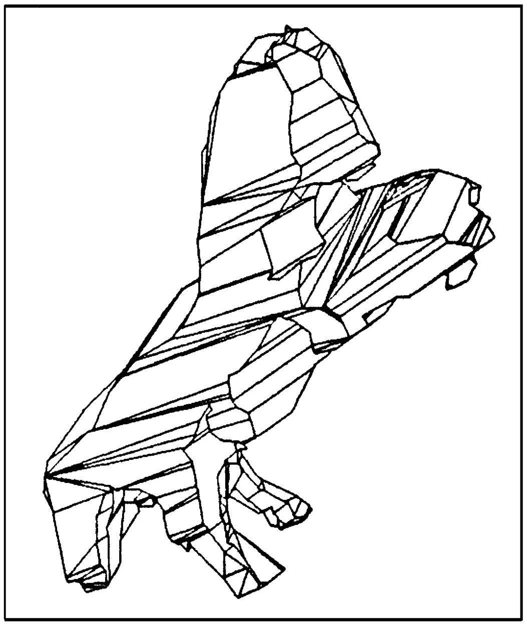
We have continued our study of techniques for visual feedback to deal with scenes with realistic contrast (not just black versus white) and with realistic complexity (curved objects). Perkins [Perkins] has made a program which finds corners for visual feedback in block stacking. Although block stacking itself is of little interest, the program is interesting for its ability to function with realistic contrast levels (no special preparation of scene) and interesting for its global edge finding strategy. Perkins also made a program which found elliptic curves among edge points from the Hueckel operator [Hueckel 1971]. The program was able to identify cylinders.

Vision Language

Binford has made a beginning on a language for vision research. Previously, the laboratory has built a hand/eye monitor system to systematize cooperating routines at a job level; a library of simple procedures has been implemented [Pingle]. It has been found that the job level is too coarse to be useful for accomplishing our objectives: to allow research to build on previously built data structures and modules; to allow a common vocabulary. The new effort is not predominantly a system software effort, but a scientific effort, aimed at providing a language in which strategies can be expressed. Our experience is that it is difficult for humans to program in LISP or SAIL, and that we cannot reasonably expect strategy programs to be expressed at that low level of language. The language will be embedded in SAIL. Our previous work in representation of shape has been significant; now we are extending the study of representation to visual program structure, including intermediate internal structures. Our experience with the hand language is that this is a valuable step.

Polyhedral Modeling

Baumgart [Baumgart 19'73, 1974A,1974B] has developed a system for descriptive computer vision based on polyhedral modeling and image contouring. Baumgart's overall design idea may be characterized as an inverse computer graphics approach to computer vision. In computer graphics, the world is represented in sufficient detail so that the image forming process can be numerically simulated to generate synthetic television images; in the inverse, perceived television pictures are analyzed to compute detailed geometric models. To date, polyhedra (such as in the figure) have been automatically generated by intersection of silhouette cones from four views of a white plastic horse on a black turntable. The viewing conditions are necessarily favorably arranged, but then the claimed results are honest.



Bibliography

[Agin] Agin, Gerald J., Thomas 0. Binford, Computer Description of Curved Objects, *Proc. Third International Joint Conf. on Artificial Intelligence,* Stanford University, August 1973.

[Bajcsy] Bajcsy, Ruzena, Computer Description of Textured Scenes, *Proc. Third Int. joint Conf. on Artificial Intelligence,* Stanford U., 1973.

[Baumgart 19731 Bruce C. Baumgart, Image Contouring and Comparing, Stanford A. I. Memo AIM-199, October 1973.

[Baumgart 1974A] Bruce G. Baumgart, CEOMED - A Geometric Editor, Stanford A. I. Memo AIM-232, May 1974.

[Baumgart 1974B] Bruce G. Baurngart, Geometric Modeling for Computer Vision, Stanford A. I. Memo AIM-249, October 1974.

[Binford] T.O. Binford, Visual Perception by Computer, Invited paper at IEEE Systems Science and Cybernetics, Miami, December *1971.*

[Bullock] Bruce Bullock, in preparation.

[Hannah] Marsha Jo Hannah, Computer Matching of Areas in Stereo Images, *Ph.D. Thesis in Computer Science,* Stanford A. I. Memo AIM-239, July 1974.

[Hueckel 1971] M. H. Hueckel, An Operator Which Locates Edges in Digitized Pictures, Stanford A. I. Memo AIM-105, December 1969, also in JACM, Vol. 18, No. 1, January 1971.

[Hueckel 1973] Hueckel, Manfred H., A Local Visual Operator which Recognizes Edges and Lines, *J. ACM,* October 1973.

[Nevatia] R. K. Nevatia and T. 0. Binford, Structured Descriptions of Complex Objects, *Proc. Third International Joint Conf. on A.I.*, Stanford Univ., February 1973.

[Perkins] Walton A. Perkins, Thomas 0. Binford, A Corner Finder for Visual Feedback, Stanford A. I. Memo AIM-214, September 1973.

[Pingle] Karl K. Pingle, Hand/Eye Library, Stanford Artificial Intelligence Laboratory Operating Note 35. 1, January 1972.

[Sobel] Sobel, Irwin, On Calibrating Computer Controlled Cameras for Perceiving 3-D Scenes, *Proc. Third Int. Joint Conf. on Artificial Intelligence*, Stanford U., 1973; also in Artificial Intelligence J., Vol. 5, No. 2, Summer 1974.

[Thomas] A.J. Thomas and K.K. Pingle, in preparation.

[Yakimovsky] Yakimovsky, Yoram, Jerome A. Feldman, A Semantics-Based Decision Theoretic Region Analyzer, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford University, August 1973.

2.2.2 Visual Guidance of a Vehicle

Lynn Quam and Hans Moravec are working on a "cart project" that has as one of Its goals the development of a set of techniques to enable a vehicle to guide Itself through a unknown environment on the basis of visual information. As a first step, a program has been written which takes a motion parallax pair of pictures of a scene, finds "interesting" feature points scattered over one image, and tries to locate the same features in the other image, deducing their location in three dimensions.

This program has been tried on about 40 pairs of pictures of outdoor scenes, and in all cases was able to line up the horizons properly. In about 60% of the cases one or two nearby obstacles were located accurately. In the remaining 40% the "matching" features 'found were typically pieces of the same road edge farther along the path than the desired feature in the first picture. This kind of error precludes exact measurement of distances, but still provides enough information so that the edge can be avoided.

Significant subtasks completed include the operator which locates "interesting" features by thresholding and locally maximizing directional variation in grey level. A minor innovation is a distortion of the pictures in the horizontal direction, tantamount to transforming the original planar images into a cylindrical projection, thus making the scale of features invariant over camera pan.

Considerable effort has been expended in getting our existing cart hardware to the point where these techniques can be tried on a running vehicle. A set of control routines, which calculate an optima.1 path for a requested position and orientation change and transmit the appropriate commands to the vehicle, were also written this past year.

Near-field Stereo

Near-field stereo has the problem that a high degree of distortion and occlusion occurs in most scenes when the baseline distance between the camera positions is comparable to the distance from either camera to objects in the scene.

For our immediate cart project goals, we are primarily interested in objects in the direction of motion. Such objects undergo predominantly a scale factor change, but previous efforts in area matching have not allowed an unknown scale change between the areas in the two images.

To handle this problem, we have developed a technique for area matching under scale change using a model for the camera position and orientation of one image relative to the preceding Image. Whenever a point in the second image is proposed as a match for a point in the first image, one can use the camera position model to determine at what depth the 3-space point must lie. The ratio of the distances between this point and the two camera positions corresponds to the observed scale factor change. This scale factor ratio is used to geometrically scale the points in the area of Interest in one image prior to computation of the area correlation operator.

This technique was applied to a sequence of road Images gathered as a "typical" road envrronment. The results indicated that areas with scale changes of up to 1.5 or 2.0 to 1 could be efficiently and- reliably be matched. An extension of this technique which allows unequal scale changes in the vertical and horizontal directions is planned.

2.2.3 Mars Picture Analysis

The NASA Viking Project supported a feasibility study at to determine if computer image processing techniques could be used for aerial/orbital photogrammetry. The object was to take pairs of orbital photographs of portions of planets (the Moon and Mars in the study) and construct contour maps for the terrain. These techniques are under study for checking the suitability of the proposed 'landing sites for the 1975 Viking missions to Mars.

The approach we took was to first match up as much as possible of the two images wrth a program that used correlation, the region grower, and an approximate camera model derived from spacecraft position and pointing data. The parallaxes were then converted to elevations by a second program and contoured at the desired intervals by a third program.

Early results are fairly promising. Given images which are of sufficient resolution, reasonably free from noise and have sufficient information content, the computer can produce contour maps in a small fraction of the time required by traditional photogrammetry techniques [Quam1974].

Several articles have recently been published based on earlier work supported by NASA on interactive analysis of photographs of Mars taken by the Mariner satellites [Quam 1973, Sagan, Veverka].

Bibliography

[Quam1973] Quam, Lynn, Robert Tucker, Botond Eross, J. Veverka and Carl Sagan, **Mariner** 9 Picture **Differencing** at **Stanford**, Sky *and Telescope*, August 1973.

[Quam1974] Quam, Lynn and Marsha Jo Hannah, An Experimental System in Automatic Stereo Photogrammetry, Stanford A. I. Memo, **1974 (forthcoming)**.

[Sagan] Sagan, Cart, J. Veverka, P. Fox, R. **Dubisch**, R. French, **P. Gierasch**, L. **Quam**, J. Lederberg, E. **Levinthal**, R. Tucker, B. Eross, **J. Pollack**, Variable Features on Mars II: Mariner 9 Global Results, *Journal of Geophysical Research, 78*, 4163-4196, 1973.

[Veverka] Veverka, J., Carl Sagan, Lynn Quam, R. Tucker, B. Eross, Variable Features on Mars III: Comparison of Mariner 1969 and Mariner 1971 Photography, Icarus, 21, 317-368, 1974.

2.3 Mathematical Theory of Computation

Several articles based on our earlier work in mathematical theory of computation were published during the year [1, 2, 3, 4, 5] and Manna published the first textbook in this field [6].

Vuillemin's Ph.D. thesis examines the connection between the concept of least fixed-point of a continuous function and recursive programs [7].

2.3.1 FOL

The FOL project was designed by Richard Weyhrauch and John McCarthy to create an environment in which first order logic and related traditional formal systems can be used with ease and flexibility. FOL is a proof checker based on the natural deduction style of representing the proofs of first order logic. The ability to use FOL to do substantive experiments is just becoming feasible. Some of these are described below.

Eventually we expect FOL to act as a practical system in which the verification of the correctness and equivalence of programs can be carried out in the language of ordinary mathematics. The theoretical discussion of how this can be accomplished has been outlined in the papers of McCarthy, Floyd, Manna and others. The reduction of these ideas to practice is still in the experimental stage.

The above task requires a system that can represent the traditional arguments of mathematics. Thus a major part of our effort is devoted to developing a smooth and useful embedding of traditional set theory into this environment, and for ways to deal correctly with the metamathematics necessary to completely represent any substantive part of mathematical practice.

An example of using FOL to prove a very simple theorem follows. Lines beginning with "*****" are input and the others are output.

```
*****DECLARE INDVAR x y;DECLARE PREDCONCT F 1;
*****TAUT F(x)∨¬F(x);
1 F(x)∨¬F(x)
*****∃I 1, xcy occ 2;
2 ∃y.(F(x)∨¬F(y))
*****∀I 2, x;
3 ∀x.∃y.(F(x)∨¬F(y))
```

The first large proofs using FOL are reported by Mario Aiello and Richard Weyhrauch [9]. They describe an axiomatization of the metamathematics of FOL and prove several theorems using the proof checker.

Weyhrauch has also expanded McCarthy's idea of a computation rule using a notion he has called *semantic attachment*. This is a uniform technique for using the computation to decide sentences like 32 or 3+7=8 or ISON(BLACKKING, BKN1, BOARD) or HAS(monkey, bananas). Independently of this, Arthur Thomas suggested using FOL in a similar way to explore models of perception and their interaction with the actual world. Robert Filman is using these ideas extensively to axiomatize basic chess knowledge.

Several preliminary users manuals were produced for FOL, and an AI memo [10] will appear soon.

2.3.2 LCF

Progress was made in two directions. Mario and Luigia Aieilo and Richard Weyhrauch produced an axiomatization of the programming language PASCAL using LCF. This project represents a major step towards using LCF as a practical program verification system. This work is reported on in [8] and [11]. PASCAL was chosen in order to compare the techniques of Dana Scott's extensional approach to program semantics with that of Robert Floyd and C.A.R. Hoare. Thel latter approach is represented at the AI lab by David Luckham and his work on the PASCAL program verification system [Section 2.4.1].

Fredrich von Henke rewrote the LCF program and expanded it to Include an axiomatization of the type-free logic originally devised by Dana Scott, Robin Milner and Richard Weyhrauch. In addition, von Henke used the type-free logic to study the functionals definable over data structures which have recursive definitions.

Bibliography

[1] Ashcroft, Edward, Zohar Manna, Amir Pnueli, Decidable Properties of Monodic Functional Schemas, J. *ACM,* July 1973.

[2] Igarashi, S., R. L. London, D. C. Luckham, Interactive Program Verification: A Logical System and its Implementation, Acta Informatica, (to-appear).

[3] Katz, Shmuel, Zohar Manna, A Heuristic Approach to Program Verification, *Proceedings of the Third International Joint Conference on Artificial intelligence,* Stanford University, August 1973.

[4] Manna, Zohar, Program Schemas, in *Currents in the Theory of Computing* (A. V. Aho, Ed.), Prentice-Hall, Englewood Cliffs, N. J., 1973.

[5] Manna, Zohar, Stephen Ness, Jean Vuillemin, Inductive Methods for Proving Properties of Programs, *Comm. ACM,* August 1973.

[6] Manna, Zohar, *Introduction to Mathematical Theory of Computation,* McGraw-Hill, New York, 19'74.

[7] Jean Etienne Vuillemin, Proof Techniques for Recursive Programs, *Thesis; Ph.D. in Computer Science,* Stanford A. I. Memo AIM -2 18, October 1973.

[8] Luigia Aiello, Mario Aiello, Richard Weyhrauch, The Semantics of PASCAL in LCF, Stanford A. I. Memo AIM-221, August 1974.

[9] Mario Aiello, Richard Weyhrauch, Checking Proofs in the Metamathematics of First Order Logic, Stanford A. I. Memo AIM-222, October 1974.

[10] Richard W. Weyhrauch, Arthur J. Thomas, FOL: A Proof Checker for First-order Logic, Stanford A. I. Memo AIM-235, June 1974.

[11] Luigia Aiello, Richard W. Weyhrauch, LCFsmall: an Implementation of LCF, Stanford A. I. Memo AIM-241, August 1974.

2.4 Heuristic Programming

Heuristic programming techniques are being applied to theorem proving and automatic programming problems.

2.4.1 Theorem Proving

A group headed by David Luckham has directed their recent research toward the application of mathematical theorem proving in the area of program verification. There are now have two theorem provers:

(1) A general proof program that has been developed for research in different areas of mathematical problems. This is based on the Resolution principle and rules for equality. It contains a wide selection of proof search strategies, and incorporates an interactive user language for guiding proofs and selecting strategies. It can be used either as a theorem prover or as a proof checker. There is a facility for an automatic selection of search strategies based on an analysis of the problem, so that prior knowledge of theorem proving techniques on the part of the user is unnecessary. We summarize recent developments with this program below.

(2) A fast special purpose prover (called the

Simplifier) designed specifically fot program verification. This program makes use of documentation submitted with the program to reduce the complexity of logical verification conditions (the truth of these conditions imply the correctness of the program). Originally, this program was Intended as a preprocessor for the general theorem prover. However, it includes a limited theorem-proving capability aimed at eliminating the "easy work" and this has turned out in experiments to be a powerful component of the verification system (see below).

A user's manual for the general theorem prover is available [1]; and publications [2,3,4,5] deal with interactive applications of this program to mathematics and information retrieval. Recent experiments in using the prover to obtain new characterizations of varieties of Groups and to check tedrous proofs in Euclidean Geometry are given in [6]. A primitive "HUNCH" language has been programmed by J. Allen. This enables the uset to describe complex proof search procedures in which the strategies may vary during the search for a proof. This language is currently being extended to permit outlines of proofs to be described in a natural way. We regard this as a necessary development for more difficult applications in mathematics.

During the last year the prover has been used to provide the automatic deduction capability for the PASCAL program verification system [7]. In particular, J. Morales has made an extensive study of the verification of sorting algorithms from first principles (including, for example, SIFTUP [8] BUBBLE SORT [9], and INSERTION SORT [10]) and is working on modifications of the HUNCH language to aid in these problems.

The simplifier is a fast theorem prover incorporated into the program verification system for PASCAL programs [11]. The verification system as originally described in [7], has been extended to permit the user to submit axiomatic descriptions of data structures and specifications of (possibly uriwritten) subroutines with the program to be verified. The simplifier uses these documentation statements either as algebraic simplification rules or as logical goal reduction rules in a depth first proof search. A methodology of using the verification system to debug ad verify real life programs depending on nonstandard data structures is being developed [12]. A user's manual for this system is available [13], and experiments using the Simplifier to verify sorting and pattern matching programs on the basis of user-defined concepts are reported in [12,141. A version of this system for PL/1 (including the data type POINTER) is being programmed.

Further developments and applications of heuristic theorem proving are described in the section on Automatic Programming (c.f. Luckham and Buchanan), and an ambitious proof checking system for higher order logic has been developed by R.W. Weyhrauch (see Section on Mathematical Theory of Computation).

Bibliography

[1] Allen, J.R., Preliminary **Users Manual** for an Interactive Theorem-Prover, Stanford Artificial Intelligence Laboratory Operating Note SAILON-73.

[2] John Allen, David Luckham, **An Interactive Theorem-proving** Program, *Proc. Fifth International Machine Intelligence Workshop*, Edinburgh University Press, Edinburgh, 1970.

[3] Richard B. Kieburtz and David Luckham, Compatibility and Complexity of **Refinements** of the Resolution Principle, *SIAM J. Comput.*, Vol. 1, No. 4, December 1972.

[4] David Luckham, Refinement Theorems in Resolution Theory, *Symposium on Automatic Demonstration*, Springer-Verlag Lecture Notes in Mathematics No. 125, Berlin, 1970.

[5] David Luckham, Nils J. Nilsson, Extracting Information from Resolution Proof Trees, *Artificial Intelligence*, Vol. 2, No. 1, Spring 1971.

[6] Jorge J. Morales, Interactive Theorem Proving, *Proc. ACM National Conference*, August 1973; also Theorem Proving in Group Theory and Tarski's Geometry, forthcomrng A.I. Memo.

[7] Igarashi, S., London, R., Luckham, D., Automatic Program Verification I, Logical Basis and Its Implementation, Stanford A. I. Memo AIM-200, May 1973; to appear in Acta Informatica.

[8] Floyd, R.W., Algorithm 245, TREESORT 3, *Comm. ACM*, June 1970.

[9] Knuth, D.E., The Art of Computer Programming, Vol. 3: Sorting and Searching, Addison-Wesley, Reading, Mass., 1973.

[10] Pratt, V.R., Shellsort and Sorting Networks, Ph.D. Thesis in Comp. Sci., Stanford Univ., February 1972.

[1 1] N. Wirth, The Programming Language PASCAL, *ACTA Informatica, I* 1., 1971.

[12] F. von Henke, D. Luckham, A Methodology for Verifying Programs, A.I. Memo, forthcoming (submitted to the 1975 International Conference on Reliable Software).

[13] N. Suzuki, Short Users Manual for the Stanford Pascal Program Verifier A.I.Lab. operating note (forthcoming).

[14] N. Suzuki, Verifying Programs by Algebraic and Logical Reduction, A.I. Memo, forthcoming (submitted to the 1975 International Conference on Reliable . Software).

2.4.2 Automatic Programming

Research in automatic programming has progressed on several fronts, summarized below.

Automatic Program Generation

A heuristic theorem proving system for a Logic of Programs due to Hoare [1] forms the basis for a successful automatic programming system that has been developed over the past two years. This is an experimental system for automatically generating simple kinds of programs. The programs constructed are expressed in a subset of ALGOL containing assignments, function calls, conditional statements, while loops, and non-recursive procedure calls. The input to the system is a programming environment consisting of primitive programs, programming methods for writing loops, and logical facts. The input is in a language similar to the axiomatic language of [1]. The system has been used to generate programs for symbolic manipulation, robot control, every day planning, and computing arithmetical functions.

Two papers concerning the theory and applications of the automatic programming system have been written [2, 31. Applications of the system to generatrng assembly and repair procedures within the HAND CONTROL language [5] for simple machinery are described in [2]. Report [3] presents a full overview of the system with many examples. Details of the implementation are in Buchanan's thesis [4]. The loop construction and program optimization methods have been extended by John Allen and more ambitious applications in programming and mechanical assembly are being tackled.

Automatic Selection of Data Structures

A system has been developed which, given an algorithm expressed in terms of high-level information structures such as sets, ordered sets, and relations, automatically chooses efficient representations and produces a new program that uses these representations. Representations are picked from a fixed library of low-level data structures including linked-lists, binary trees and hash tables. The representations are chosen by attempting to minimize the predicted space-time integral of the user's program execution.

Predictions are based upon statistics of information structure use provided directly by the user and collected by monitoring executions of the user program using default representations for the high-level structures. In performance tests, this system has exhibited behavior superior to human programmers, and is at the stage where it could be Implemented in a very high level language, like SAIL. This work is reported in Jim Low's thesis [6].

Program Understanding Systems

Progress has also been made in the design of systems which can be said to have some "program understanding" ability. In our case, the primary evidence for such ability lies in the capability to synthesize programs, either automatically or semi-automatically, but such a capability alone is insufficient for understanding: the line of reasoning which the system follows during the synthesis process must also support the claim of "understanding", and we feel that most of our systems behave well in this regard.

One experimental system used its knowledge base of "programming facts" to synthesize (among others) programs which interchange elements, perform a 3-element non-recursive sort, and find the integer square root, basing choices at decision points on user responses to questions posed by the system. Another experimental system can synthesize programs from example input/output pairs, and has written about 20 simple list-processing functions.

These experiments have led us to several preliminary conclusions and to a view that two of the major research areas in program understanding systems are the exploration of various manners of program specification, and the codification of programming knowledge.

Looking at the two experimental systems mentioned above, we see two different methods of specifying the desired program: example input/output pairs and user responses to questions from the system. There seem to be many other ways in which the desired program could be specified, ranging from very formal to very informal. A unifying paradigm would seem to be a kind of dialogue between the user and the system. In such a dialogue any of these methods (or even several of them) might be employed, depending on suitability for the program, and preferences of the user. Work is currently progressing on various methods of modeling and conducting such dialogues.

Our experimental systems and numerous hand simulations of program understanding systems indicate that satisfactory behavior can only be expected when the system contains a large body of knowledge. For the understanding of programs in a given domain, there is considerable domain-specific knowledge. Additionally, there seems to be a large body of "pure" programming knowledge which is relatively domain-independent. Much of our work is aimed at isolating, codifying, and representing this knowledge.

Our early experimental systems as well as discussions of conclusions and future plans are reported in [7] and in papers by Green and Barstow, and by Shaw, Swartout, and Green, which are in preparation.

Bibliography

[1] C.A.R. Hoare, An Axiomatic Approach to Computer Programming, *Comm. ACM*, October 1969.

[2] David Luckham, Jack Buchanan, Automatic Generation of Programs Containing Conditional Statements, *Proc. A.I.S.B. Summer Conference*, Sussex, England, July 1974.

[3] Jack Buchanan, David Luckham, On Automating the Construction of Programs, Stanford A. 1. Memo AIM-236, May 1974, (submitted to the JACM).

[4] Jack Buchanan, A Study in Automatic Programming, *Ph.D. Thesis in Computer Science*, Stanford A. I. Memo AIM-245, September 1974.

[5] R. Bolles, L. Paul, The Use of Sensory Feedback in Programmable Assembly Systems, Stanford A. I. Memo AIM-220, October 1973.

[6] Low, Jim, Automatic Coding: Choice of Data Structures, *Ph.D. Thesis in Computer Science*, Stanford A. I. Memo AIM-242, (forthcoming).

[7] Green, Cordell, R.J. Waldinger, David R. Barstow, Robert Elschlager, Douglas B. Lenat, Brian P. McCune, David E. Shaw, Louis I. Steinberg, Progress Report on Program-Understanding Systems, Stanford A. I. Memo AIM-240, (forthcoming).

[8] Manna, Zohar, Automatic Programming, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford University, August 1973.

2.5 Natural Language

Research continued on three aspects of natural language:

1) speech recognition, which typically deals with acoustic waveforms,
2) natural language understanding, which generally starts with text, and
3) higher mental functions, which deals with psychiatric problems manifested through natural language.

A lack of funding support for speech recognition has resulted in a progressive reduction of that activity.

2.5.1 Speech Recognition

During the past year the focus of speech recognition research at Stanford has changed from machine learning based phoneme recognition [1] to linguistically structured acoustic-phonetic processing [2]. The philosophy of the research has been to attempt to extract a maximum of linguistic information from the speech signal. This led to using waveform type segmentation, pitch synchronous analysis of voiced regions, waveform level steady state detectors and syllable detectors. The major effort has gone into developing algorithms which automatically extract the linguistic information at each level; waveform and short time frequency spectra.

Neil Miller has developed a semantic pitch detector which used the expected pattern of excitation and exponential decay of the acoustic signal during voicing. The purpose of the earliest version of the pitch detector was to mark the beginning of each pitch period during voicing, making a voicing decision along the way. Various versions of this program find the pitch in less than real time [4].

Waveform level acoustic segmentation algorithms were developed by Jim Hieronymus [3]. On the sub phonemic level, areas of steady frequency spectra where continuant phonemes most closely approach their target frequency values are found by pitch synchronous waveform comparisons. A process for segmentation into continuants and transitions was developed based on a model of the way a human visually compares waveforms. An algorithm for waveform type segmentation into voiced, nasalized vowel, voiced fricative, unvoiced fricative, and silence was developed based on amplitudes, integrals under the acoustic peaks in a pitch period and zero crossings.

Pitch synchronous short time frequency spectra were found to contain clearly delimited formants, so that linear predictive modeling of the spectrum was not necessary in order to readily find the formants. In addition, pitch synchronous analysis preserves the maximum information about formant transitions. Transrtions in and out of stop consonants are clearly seen. A formant extractron algorithm was developed by Arthur Samuel to pick the formant peaks from the pitch sychronous FFT spectra. Visual comparisons with the output of the M.I.T. Lincoln Labs formant tracker and sonograms have been made. Generally the formant tracking is as good as or better than much more complicated tracking programs using LPC data. Pitch synchronous analysis also preserves the true formant bandwidths, which may be useful in nasal 'detection.

Moorer has developed a very efficient scheme for performing pitch period analysis [5].

A system for displayrng speech waveforms, their frequency spectra, and for hearing the utterance being examined has been developed. Hard copy plots can be made from the display program using the Xerox Graphics Printer.

After April 1974, the group worked on refin i ng the pitch detector, syllable detection and rate of speech measures based on syllable counts. A plan to do some research in contest dependent vowel recognition was formulated, since this is a significant problem area in present speech recognition systems.

This work is continuing at a very low level for lack of funding. Several articles are in preparation from the research work done in 1973-74.

Bibliography

[1] Thosar, Ravindra B., Recognition of Continuous Speech: Segmentation and Classification using Signature Table Adaptation, Stanford A. I. Memo AIM-2 13, September 1973.

[2] Hieronymus, J. L., N. J. Miller, A. L. Samuel, The Amanuensis Speech Recognition System, *Proc.* IEEE *Symposium on Speech Recognition,* April 1974.

[3] Hieronymus, J. L., Pitch Synchronous Acoustic Segmentation, *Proc. IEEE Symposium on Speech Recognition*, April 1974.

[4] Miller, N. J., Pitch Detection by Data Reduction, *Proc. IEEE Symposium on Speech Recognition,* April 1974.

[5] Moorer, James A., The Optimum Comb Method of Pitch Period Analysis of Continuous Speech, *IEEE Trans. Acoustics, Speech, and Signal Processing,* Vol. ASSP-22, No. 5, October 1974.

2.5.2 Natural Language Understanding

This was a transitional year for our program of research in natural language. Roger Schank, who previously directed some of the work, was on leave at the *Instituto per gli Studi Semantici e Cognitivi*, in Switzerland. He continued his research into conceptual dependency theory for natural language understanding at the institute [2]. His work, along with that of Chris Riesbeck, Neil Goldman, and Charles Rieger, led to the completion of the MARGIE system [1].

One aspect of this is reported in Rieger's thesis [11], which develops a memory formalism as a basis for examining the inferential processes by which comprehension occurs. Then, the notion of inference space is presented, and sixteen classes of conceptual inference and their implementation in the computer model are examined, emphasizing the contribution of each class to the total problem of understanding. The idea of points of contact of Information structures in inference space is explored. A point of contact occurs when an inferred unit of meaning from one starting point within one utterance's meaning graph either confirms (matches) or contradicts an inferred unit of meaning from another point within the graph, or from within the graph of another utterance.

The work of the other members of the group will be published in the coming year, including a book edited by Schank, summarizing research in conceptual dependency.

Yorick Wilks continued his work on machine translation [3, 4, 5, 6, 7, 8, 9, 10]. In particular, he studied the way in which a Preference Semantics system for natural language analysis and generation tackles a difficult class of anaphoric inference problems (finding the correct referent for an English pronoun in context). The method employed converts all available knowledge to a canonical template form and endeavors to create chains of non-deductive Inferences from the unknowns to the possible referents.

Annette Herskovits worked on the problem of generating French from a semantic representation [13]. She concentrated on the second phase of analysis, which binds templates together into a higher level semantic block corresponding to an English paragraph, and which, in operation, Interlocks with the French generation procedure. French sentences are generated, by the recursive evaluation of procedural generation patterns called stereotypes. The stereotypes are semantically context sensitive, are attached to each sense of English words and keywords and are carried into the representation by the analysis procedure.

In addition, members of the translation group entered into discussions with others in the laboratory in a series of conversations dealing with some of the issues connecting artificial Intelligence and philosophy [14]. The major topics included the question of what kind of theory of meaning would be involved in a successful natural language understanding program, and the nature of models in AI research.

Terry Winograd spent the year at Stanford as a visitor from MIT, and continued his work on natural language understanding and its relationship to representation theory. He published a number of papers outlining his theories [15, 16, 17, 18, 20] and an introduction to artificial intelligence and the problems of natural language [19]. He gave a number of talks, including a lecture series at the Electrotechnical Laboratory in Tokyo, the Tutorial on Natural Language at the International Joint Conference on Artificial Intelligence (Palo Alto, August 1973), an invited lecture at the ACM SIGPLAN-SIGIR interface meeting (Washington D.C., November, 1973), and "A Computer Scientist Looks at Memory", a part of Sigma Xi Lecture Series (Palo Alto, February 1974).

Bibliography

[1] Schank, Roger C., Neil Goldman, Charles J. Rieger III, Chris Riesbeck, MARGIE: Memory, Analysis, **Response Generation and** Inference **on** English, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford University, August 1973.

[2] Schank, Roger C., Kenneth Colby (eds), *Computer Models of Thought and Language*, W. H. Freeman, San Francisco, 1973.

[3] Wilks, Yorick, The Stanford Machine Translation and Understanding Project, in Rustin (ed.) *Natural Language Processing, New* York, 1973.

[4] Wilks, Yorick, Understanding Without Proofs, *Proceedings of the Third International Joint Conference on Artificial Intelligence,* Stanford University, August 1973.

[5] Wilks, Yorick, Annette Herskovits, An Intelligent Analyser and Generator of Natural Language, *Proc. Int. Conf. on Computational Linguistics,* Pisa, Italy, *Proceedings of the Third International Joint Conference on Artificial Intelligence,* Stanford University, August 1973.

[6] Wilks, Yorick, The Computer Analysis of Philosophical Arguments, *CIRPHO,* Vol. 1, No. 1, September 1973

[7] Wilks, Yorick, An Artificial Intelligence Approach to Machine Translation, in Schank and Colby (eds.), *Computer Models of Thought anti Language,* W. H. Freeman, San Francisco, 1973.

[8] Wilks, Yorrck, One Small Head -- Models and Theories in Linguistics, *Foundations of Language,* Vol. 10, No. 1, January 1974.

[9] Wilks, Yorick, Preference Semantics, E. Keenan (ed.), *Proc. 1973 Colloquium on Formal Sonuntics of Natural Language,* Cambridge, U.K., 1974.

[10] Wilks, Y., Machine Translation, in *Current Trends in the Language Sciences,* T.A. Sebeok, (ed.), forthcoming.

[11] Rieger, Charles J., Conceptual Memory: A Theory and Computer Program for Processing the Meaning Content of Natural Language Utterances, Stanford A. I. Memo AIM-233, June 1974.

[12] Wilks, Yorick, Natural Larrguage Inference, Stanford A. I. Memo AIM-2 11, September 1973.

[13] Herskovits, Annette, The Generation of French from a Semantic Representation, Stanford A. I. Memo AIM-212, September 1973.

[143 Anderson, D. B., T. O. Binford, A. J. Thomas, R. W. Weyhrauch, Y. A. Wilks, After Leibniz . . . . Discussions on Philosophy and Artificial Intelligence, Stanford A. I. Memo AIM-229, April 1974.

[15] Winograd, Terry, A Process Model of Language Understanding, in Schank and Colby (eds.), *Computer Models of Thought and Language,* W. H. Freeman, San Francisco, 1973.

[16] Winograd, Terry, The Processes of Language Understanding in Benthall, (ed.), *The Limits of Human Nature,* Allen Lane, London, 1973.

[17] Winograd, Terry, Language and the Nature of Intelligence, in C.J. Dalenoort (ed.), *Process Models for Psychology,* Rotterdam Univ. Press. 1973

[18] Winograd, Terry, Breaking the Complexity Barrier (again), *Proc. SIGPLAN-SIGIR Interface Meeting, 1973.*

[19] Winograd, Terry, Artificial Intelligence -- When Will Computers Understand People?, *Psychology Today,* May 1974.

[20] Winograd, Terry, Parsing Natural Language via Recursive Transition Net, in Yeh (ed.) *Applied Computation Theory,* Prentice-Hall, 1974.

2.5.3 Higher Mental Functions

The Higher Mental Functions Project is directed by Kenneth Mark Colby and is supported by the National Institute Of Mental Health. The overall objective of the project are to develop computer models for problems in psychiatry.

One model is a simulation of paranoid thought processes (PARRY) which can be interviewed using unrestricted natural language input. Another involves a computer-aided treatment for nonspeaking autistic children.

Bibliography

[1] Schank, R.C., Colby, KM. (eds.), *Computer Models of Thought and Language,* W.H. Freeman and Co., San Francisco, California, 1973.

[2] Colby, K.M. *Artificial Paranoia,* Pergamon Press, N.Y., 1974.

[3] Enea, H. and Colby, K.M. Idiolectic Language-Analysis for Understanding Doctor-Patient Dialogues, *Proc. Third International joint Conference on Artificial Intelligence,* Stanford University, August 1973.

[4] Colby, K.M. and Parkison, R.C. Pattern-matching rules for the Recognition of Natural Language Dialogue Expressions, *American Journal of Computational Linguistics,* 1, 1974.

[5] Hilf, Franklin, Use of Computer Assistance in Enhancing Dialog Based Social Welfare, Public Health, and Educational Services in Developing Countries, *Proc. 2nd Jerusalem Conf. on Info. Technology,* July 1974.

[6] Wilks, Y. Semantic Procedures and Information, in *Studies in the Foundations of Communication,* R. Posner (ed.), Springer, Berlin, forthcoming.

[7] Colby, K.M. and Kraemer, H. Objective Measurement of Nonspeaking Children's Interactions with a Computer Controlled Program for the Stimulation of Language Development, (in press 1974).

2.6 Programming Languages

We continue to find it profitable to invest a portion of our effort in the development of programming languages and related facilities. We have already discussed the development of "HAL", the advanced "hand language" for robotics research [Section 2.11. We expect that work on automatic programming [Section 2.4.2] will greatly increase the power of programming, though such systems are not very practical yet.

The languages LISP [6,7,8], FAIL [10], and SAIL [11] carry the bulk of the programming workload here and at a number of other PDP-10 installations. We have continued to make modest improvements in these systems, which we originated.

The Higher Mental Functions group, under NIMH sponsorship, has been developing a pattern-directed extensible language called LISP 70 [9].

Professor Hoare spent a sabbatical here, continuing to develop his ideas on structured programming and related concepts [2,3,4].

Swinehart completed a dissertation on an Interactive programming system that controls multiple processes [5].

Bibliography

[1] Feldman, Jerome A., James R. Low, Comment on Brent's Scatter Storage Algorithm, *Comm. ACM,* November 1973.

[2] Hoare, C.A R. Parallel Programming: an Axiomatic Approach, Stanford A. I. Memo AIM-219, October 1973.

[3] Hoare, C.A.R., Recursive Data Structures, Stanford A. I. Memo AIM-223, December 1973.

[4] Hoare, C.A.R., Hints on Programming Language Design, Stanford A. I. Memo AIM-224, December 1973.

[5] Daniel C. Swinehart, COPILOT: A Multiple Process Approach to Interactive Programming Systems, *Ph.D. Thesis in Computer Science,* Stanford A. I. Memo AIM-230, August 1974.

[6] Quam, Lynn, Whitfield Diffie, Stanford LISP 1.6 Manual, Stanford A. I. Lab. Operating Note SAJLON-28.7, 1973.

[7] Smith, David C., MLISP2, Stanford A. I. Memo AIM-195, in diskfile: MLISP2[AIM,DOC], May 1973.

[8] Smith, DC. and Enea, H. Backtracking in MLISP2, *Proc. Third International Joint Conference on Artificial Intelligence,* Stanford University, August 1973.

[9] Tesler, L.C., Enea, H., and Smith, D.C. The LISP70 Pattern Matching System, *Proc. Third International joint Conference on Artificial Intelligence,* Stanford University, August 1973.

[IO] F.H.G. Wright II, R. E. Corm, FAIL, Stanford A. I. Memo AIM-226, April 1974, in diskfile: FAIL.REG[AIM,DOC], update in FAIL.UPD[AIM,DOC].

[1 I] VanLehn, Kurt, (ed.), SAIL User Manual, Stanford A. 1. Memo AIM-204, July 1979; in diskfile: SAIL.KVL[AIM,DOC], update in SAIL.UPD[AIM,DOC].

2.7 Computer Facilities

Our primary computer facility continues to be PDP-IO (KA-10 processor) with 68 display terminals online. It a rather efficient system in that it can gracefully carry a load of forty-some sizable Jobs. Even so, it is chronically overloaded by the local demand.

2.7.1 Hardware

In late 1973 to early 1974, we received the components of a new realtime system, namely a PDP-11/45 processor, an SPS-4 1 processor, and a 192K×16 bit Intel MOS memory. This subsystem is connected to the PDP-10 and is being developed mainly for computer vision and manipulator control.

Late in 1973, we installed an audio switch that connects any of 16 audio sources to any of 64 speakers on display terminals. This permits general audio responses from the computer and also supplies sound to go with television images that are available on the video switch. The cost of the audio switch was kept low by using digital gates for switching. The audio signal is encoded as a pulse-width modulated square wave at a frequency of about 100 KHz.

In December 1973 we received a BBN Pager that had become surplus at NASA-Ames and connected it to our KA- 10 processor. System changes to exploit the pager are under development.

We replaced our six IBM 3330 disk drives with four double density drives in June 1974. This increases our file system capacity to 136 million words but reduces the monthly lease costs slightly.

2.7.2 Software

Generally, recent changes in the timesharing monitor were made only to accomodate hardware changes. Documentation was greatly improved by the new monitor command manual [4] and program interface manual [2].

PUB, the document compiler [I], had a few bells and whistles added (mostly by Larry Tesler, who is now at Xerox PARC) and was used to produce this report.

The online newswire system called APE has been superceded by NS [3], which has a number of new capabilities and accesses both Associated Press and New York Times newswires.

Our display-or-tented text editor "E" had a few features added, and much improved documentation [5]. Though it is not complete, it still appears to be the world's best text editor.

Baumgart Improved his geometric editor [6], which facilitates the interactive design of three-dimensional objects and produces various graphical and photographic representations of them.

Our interactive drawing program for digital logic design [7] continues in use at MIT, Digital Equipment Corporation, Carnegie-Mellon University, and here.

Bibliography

[1] Tesler, Larry, **PUB -- The** Document Compiler, Stanford A. I. Lab. Operating Note SAILON-70, September 1972; in dsikfile: PUB.TES[S,DOC], supplement in PUBSUM.LES[UP,DOC].

[2] Frost, Martin, **UUO** Manual, Stanford A. I. Lab. Operating Note SAILON-55.3, December 1973; in diskfile: UUO.ME[S,DOC], update in UUO.UPD[S,DOC].

[3] Frost, Martin, **Reading** the Wireservice News, Stanford A. I. Lab. Operating Note SAILON-72.1, in diskfile: NS.ME[S,DOC], September 1974.

[4] Harvey, Brian, Monitor **Command** Manual, Stanford A. I. Lab. Operating Note SAILON-54.4, September 1974; in diskfile: MONCOM.BH[S,DOC], update in MONCOM.UPD[S,DOC].

[5] Samuel, Arthur, TEACH, in diskfile: TEACH[UP,DOC], 1974.

[6] Baumgart, B.C., GEOMED, Stanford A. I. Memo AIM-232, May 1974.

[7] Helliwell, Dick, Stanford **University Drawing** System, in diskfile: SUDS.RPH[UP,DOC], April 1974.

3. HEURISTIC PROGRAMMING PROJECT

We begin this annual report by mentioning one of the tasks that the ARPA IPT Office asked one of the co-Principal Investigators, Professor Feigenbaum, to perform during the year: to write a paper explicating the current goal structure of Artificial Intelligence as a scientific and technological endeavor, and suggesting a set of most fruitful lines of advanced research and exploratory development ovei the next five years. This task was completed in November, 1973, and a report prepared for ARPA (available as disk file AI.RPT[1,EAF] at SU-AI on the ARPA net).

That document is used as the basis of organizing the material contained in this annual report, since portions of it provide an excellent framework for the activities of this project. Where quotation marks otherwise unidentified are used, the quotation is from the Feigenbaum report to ARPA (sometimes slightly edited).

The project's research activities continue to be motivated by this global view of AI research: "Artificial Intelligence research is that part of Computer Science that is concerned with the symbol-manipulation processes that produce -intelligent action. By 'intelligent action is meant an act or decision that is goal-oriented, arrived at by an understandable chain of symbolic analysis and reasoning steps, and is one in which knowledge of the world informs and guides the reasoning." The project aims at creating computer programs that act as "intelligent agents" to human problem solvers in areas of scientific problem solving, hypothesis induction, and theory formation; diagnosis and treatment of program failures (automatic debugging) and medical problems. It aims also at a general understanding of the information processes and structures needed to carry out these types of intelligent agent activity; and the construction of necessary programming tools to facilitate the building of such programs.

3.1 Knowledge-based Systems Design

"The AI field has come Increasingly to view as its main line of endeavor: knowledge representation and use, and an exploration of understanding (how symbols inside a computer, which are in themselves essentially abstract and contentless, come to acquire a meaning).'*

"In this goal of AI research, there are foci upon the encoding of knowledge about the world in symbolic expressions so that this knowledge can be manipulated by programs; and the retrieval of these symbolic expressions, as appropriate, in response to demands of various tasks. This work has sometimes been called 'applied epistemology or 'knowledge engineering'."

Two of the major subgoals of the work in knowledge-based systems design constitute the most important thematic lines of research in this project. They are:

"A. How is the knowledge acquired, that is needed for understanding and problem solving, and how can it be most effectively used?

B. How is knowledge of the world to be represented symbolically in the memory of a computer? What symbolic data structures in memory make the retrieval of this information in response to task demands easy?"

Significant advances on these problems have been made in the past year. They are detailed below.

"The paradigm for this research is, very generally sketched, as follows: a situation is to be described or understood; a signal input is to be interpreted; or a decision in a problem-solution path is to be made.

Example: The molecule structure-generator must choose a chemical functional group for the 'active center' of the molecular structure it is trying to hypothesize, and the question is, 'What does the mass spectrum indicate is the 'best guess'?"

Specialized collections of facts about the various particular task domains, suitably represented in the computer memory (call these Experts) can recognize situations, analyze situations, and make decisions or take actions within the domain of their specialized knowledge.

Example: In Heuristic DENDRAL, the Experts are those._ that know about stability of organic molecules in general, mass spectrometer fragmentation processes in particular, nuclear magnetic resonance phenomena, etc."

"Within this paradigm lie a number of important problems to which we have addressed ourselves:

a. Since it is now widely recognized that detailed specific knowledge of task domains is necessary for power in problem solving programs, how is this knowledge to be Imparted to, or acquired by, the programs?
   - a 1. By interaction between human expert and program, made ever more smooth by careful design of interaction techniques, languages 'tuned' to the task domain, flexible internal representations. Our work on situation-action tableaus (production systems) for flexibly transmitting from expert to machine details of a body of knowledge is an example.
   - a2. 'Custom-crafting' the knowledge in a field by the painstaking day-after-day process of an AI scientist working together with an expert in another field, eliciting from that expert the theories, facts, rules, and heuristics applicable to reasoning in his field. This was the process by which Heuristic DENDRAL's 'Expert' knowledge was built. We have also used it in AI application programs for treatment planning for infectious disease using antibiotics, and protein structure determination using X-ray crystallography.
   - a3. By inductive inference done by programs to extract facts, regularities, and good heuristics directly from naturally-occurring data. This is obviously the path to pursue if AI research is not to spend all of its effort well into the 21st Century, building knowledge-bases in the various fields of human endeavor in the custom-crafted manner referred to above. The most important effort in this area has been the Meta-DENDRAL program described below."

3.2 The Meta-DENDRAL Program: Knowledge Acquisition by Theory Formation Processes

The research task mentioned above as (a3.) has been studied in the context of a specific inductive theory formation task, a task which is ideally matched to the project's research history: inferring parts of the theory of mass spectrometry of organic molecules (i.e., rules of fragmentation of molecules) from instrument data (i.e., mass spectra). This is an area in which we have not only a vast amount of empirical data, cooperative collaborating experts in the area, and a considerable understanding of the structure of knowledge in the area; but also we have an operating performance program capable of using (thereby testing) the knowledge inferred. This program is the Heuristic DENDRAL program, developed previously wrth ARPA support (and substantial NIH support).

Compared to grammatical inference, sequence extrapolation, or other induction tasks, scientific theory formation as a prototypical task for knowledge acquisition, has received little attention from workers in Artificial Intelligence. This may be partly because scientific theory formation is a relatively complex task, characterized by noisy and ambiguous data which must be organized within Incomplete or controversial models of the discipline. However, many task areas to which AI techniques might be applied have this character.

Meta-DENDRAL (MD) is a computer program that formulates explanatory rules (but not revolutionary reformulations of principles) to account for empirical data in mass spectrometry, a branch of analytical organic chemistry. The problem is one of explaining the mass spectrometry (m.s.) data from a collection of chemical compounds -- in other words, of telling why the given compounds produce the observed data in a mass spectrometer. The most recent description of the Meta-DENDRAL theory formation work is given in "Scientific Theory Formation by Computer", a paper presented to the NATO Advanced Study Institute on Computer Oriented Learning Processes (Bonas, France, Aug. 26 - Sept. 6, 1974). The following summary is taken from that paper.

The rules of mass spectrometry are expressed in texts, and represented in the program, as conditional rules (also called "productions"). The productions indicate what physical changes (e.g., bond breakage) we can expect a molecule to undergo within the mass spectrometer.

A discussion of our work on production system representations of knowledge appears later in this report.

The instances presented to the program are pairs of the form <molecular structure> - <mass spectrum>, i.e., pairs of known chemical structures and corresponding empirical data from mass spectrometry. A rule explains why the mass spectrometer produces some data points for a molecule of known structure. A given molecule may undergo repeated fragmentation in several possible ways, and we want to find a collection of rules to explain the whole mass spectrum.

The program is organized into four main steps:

1) data selection
2) data interpretation
3) rule generation
4) rule modification.

3.2.1 Data Selection

Knowing which data to examine and which to ignore is an important part of science. The world of experience is too varied and too confusing for an unselective scientist to begin finding and explaining regularities. Even within a sharply constrained discipline, experiments are chosen carefully so as to limit the amount of data and their variety. Typically one thinks of a scientist as starting with a carefully selected set of data and generating requests for new data. from new experiments, after formulating tentative hypotheses. Both the initial selection and the additional requests are in the realm of data selection.

The strategy behind data selection is to find a small number of paradigm molecules - i.e., molecules that are likely to exhibit regularities typical of the whole class. Rules formed from these can be checked against other data in the instance space.

3.2.2 Data Interpretation and Summary: The INTSUM Program

Experimental data in science can be interpreted under many different models. Finding explanatory rules within a model thus forces one to interpret the data under that model. For example, when one is looking for biological rules within an evolutionary model, the data (even as they are collected) are interpreted in terms of continuity of properties, similarities of behavior, etc. The rules (if any) suggested by the data are already pre-formed in that the predicates and relations used in the rules -- and often the logical form itself -- are exactly those of the model.

The data Interpretation part of the MD program (named INTSUM) describes the instances selected by the data selection program in terms of one model of mass spectrometry. Thts redescription is necessary for the reasons Just noted. Without it, rules would be formed at the pattern recognition level of statistical correlations between data points and molecular features. Although rules at this level are sometimes helpful, our intent is to produce rules that begin to say something about WHY the correlations should be observed.

Four points are Interesting here because they seem common to scientific rule formation tasks but unusual for other induction tasks:

1) The fact that the presented Instances need reinterpreting at all.

2) The ambiguity of the interpretations. The mapping from data points to processes is a one-many mapping. Sometimes a data point actually (or most probably) results from multiple processes compounding the same result. At other times a data potnt is most probably the result of only one process, even though several processes are plaustble explanations of it. And, at still other times a data point is most probably a "stray", in the sense that it was produced by an impurity in the sample or noise in the Instrument, even though several processes may be plausible explanations of it. This ambiguity in the instances makes the Induction task harder.

3) The strength of evidence associated wrth processes. The data are not merely binary readings on masses of fragments. (Most concept formation or grammatical Inference programs use only a binary separation of instances - "hit or miss", although Winston's program uses the classification of "near miss" to advantage.) The strength of evidence readings on m.s. data points can be used to focus attention on just a few of the many processes the program can consider.

4) There is more than one rule to be formed to explain the data. In the presentation of instances, there is no separation according to the rules to be formed: Instances of many rules are thrown together. The program must separate them.

3.2.3 Rule Generation: The RULEGEN Program

The collected observations of INTSUM, as with Darwin's carefully recorded observations, constitute only the weakest sort of explanation. After describing features and behavior, and summarizing the descriptions, such a record can only give a weak answer to the question, "Why is this X an A?" The answer is roughly of the form, "Because all X's seem to be A's."

The rule generation program, RULEGEN, provides an additional level of explanation by describing what attributes of the input molecular graphs seem to "cause" the molecules to behave in the observed way.

RULEGEN normally begins with the most general class of rules: The bond (or bonds) break regardless of their environment (situation). The program successively hypothesizes more specific classes of rules and checks each class against the data. Likely classes are expanded into specific rules for which additional data checks are made. The process terminates whenever (a) the more specific classes fall the data checks, or (b) individual rules fail the data checks. This procedure is outltned below:

```
1. START WITH INITIAL CLASS OF
   RULES
2. GENERATE NEXT MORE SPECIFIC
   CLASS
3. SEE IF THE CLASS PASSES THE
   FILTER
   3A. IF NOT, STOP
4. ESPAND CLASS INTO INDIVIDUAL
   RULES
5. EVALUATE RULES
6. PRUNE WITH REGARD TO
   EVALUATION
   6A. IF NO RULE REMAINS, STOP
7. FOR EACH REMAINING RULE,
   WRITE OUT RULE AND DO 2 - 7.
```

3.2.4 Rule Modification

While the programs described so far are presently operational, the addition of a rule modification phase is still under way. The program for rule modification will duplicate for its own purposes the steps already described: data selection, data interpretation and rule generation. Data selection in this case will be for the purpose of finding data that can resolve questions about rules. Those data, then, will be interpreted and rules formed from them, as described above. The results of rule generation on the new data will then be used to modify the previous set of rules. The data gathered in response to the questions asked about the old rules will determine, for example, whether those rules should be made more specific or more general. Rule modification opens interesting possibilities for feedback in the system that remain to be exploited.

The Meta-DENDRAL program is the keystone of our study of automatic knowledge acquisition. "Though the main thrust of AI research is in the direction of knowledge-based programs, the fundamental research support for this thrust is currently thin. This is a critical 'bottleneck' area of the science, since (as was pointed out earlier) it is inconceivable that the AI field will proceed from one knowledge-based program to the next painstakingly custom-crafting the knowledge/expertise necessary for high levels of performance by the programs."

3.3 Systems for Semi-Automatic Knowledge Acquisition by Experts

Previously we mentioned that one of the modes of knowledge acquisition (a.2) in wide use is Expert-Computer Scientist interaction. Currently this mode is slow, painstaking, and sometimes ineffective. Therefore, we have been conducting research aimed at introducing intelligent interaction into the process of extracting and "debugging" knowledge from Experts.

Knowledge acquisition is an Important component of an intelligent system because a complex knowledge base will almost certainly have to change as the system is applied to new problems. Gaps and errors in the knowledge base will have to be considered. We have recently designed a system that will provide an expert with high level access to the knowledge base of the system. (Work is in progress on the implementation of these ideas.)

The specific task that is the base for this study is a "diagnosis and therapy" advice system developed by researchers and students of our project, in collaboration with clinical pharmacologists, under an NIH grant -- the MYCIN system.

The knowledge modification and augmentation system will have two entry-points: (i) the expert supplying the knowledge can enter the system during the course of a consultation if something seems wrong, or (ii) at the end of a session, the post-consultation review mechanism automatically enters the system to validate the program's performance.

From the questions that the expert asks in attempting to find the error (or perhaps as a result of what he decides the error is) the problem is classified according to one of the error models. We may view this classification scheme as stupidity, ignorance, incompetence, and system errors. Thus there is either some Incorrect rule in the current knowledge base, some rule is missing, a 'concept (predicate function) is missing, or there is an error in the control structure.

In the first case 'diagnosis' and 'therapy routines in the appropriate error model attempt to discover and remedy the problem. Heavy use is made of contextual information (what subgoal was being traced, which question the user found odd, etc.).

In the second case, the 'therapy' is to invoke a rule acquisition system. This consists of a rule deciphering module and an incorporation module. The former relies on domain and context specific knowledge to aid in interpreting the expert's newly offered rule. The latter uses the current knowledge base of rules and an understanding of the system's truth model to insure that the new rule is consistent with those currently in the system.

While there appears to be little this system will be able to do beyond recognizing the last two types of errors, this can at least provide the hooks by which future, more sophisticated routines can be applied intelligently. In addition, the incompetence case is clearly related to ignorance -- in the latter the system lacks a rule it is capable of expressing, while in the former it lacks the concept necessary for even expressing the rule. Thus failure of the ignorance model to handle the problem should result in the suggestion to the incompetence model that it may be able to recognize what's really wrong.

The error models are characterizations of the types of errors in the knowledge base that we expect the system might encounter. The relevance function for each would take a look at what was known about the current state of the world to decide if it was relevant. The model which found itself most relevant would temporarily take control, using its diagnostic routines to attempt to uncover the source of the problem, and its therapeutic routines to attempt to fix the problem. Thus it effectively offers its advice on how to proceed by actually taking control for a time.

The rule models used by the rule acquisition system are slightly different in two ways. The task here is to decipher the new rule which the expert has typed in, and in this case the models offer advice on what the content of the new rule is likely to be, relying on contextual information to hypothesize the type of the incoming rule. Hence they do not directly take control, but more passively offer advice about what to expect. In addition, the large number of rules currently in the system makes conceivable the automatic generation of rule models. By using a similarity metric to form analogy sets, and then describing the properties of the rules in the set in general enough terms, we obtain a set of models on which to base acquisition. The error models, on the other hand, are both few enough, and specialized enough to require creation by hand.

3.4 Knowledge Acquisition and Deployment: A New AI Application to Scientific Hypothesis Formation

"We have felt for some time that it is necessary to produce additional case-study programs of hypothesis discovery and theory formation (i.e., induction programs) in domains of knowledge that are reasonably rich and complex. It is essential for the science to see some more examples that discover regularities in empirical data, and generalize over these to form sets of rules that can explain the data and predict future states. It is likely that only after more case-studies are available will AI researchers be able to organize, unify and refine their ideas 'concerning computer-assisted induction of knowledge."

In searching for new case-studies, the Heuristic Programming Project has developed criteria for a successful application, explored several task domains, and has brought one new application, discussed below, far enough along to submit grant applications (to NSF and NIH) for further research. Meanwhile, other laboratories have made significant progress in the design and implementation of AI programs in this general area -- notably the SOPHIE system for electronic troubleshooting (BBN) and the HASP-Sonar work (see Section 3.7).

3.4.1 Background

Our choice of the protein crystallography problem as a task domain in which to study information processes of scientific problem solving followed several informal discussions with Professor Joseph Kraut and his colleagues at UCSD, who were eager to explore new computational techniques for protein structure elucidation. They explained to us how 3-dimensional structures of crystallized protein molecules are Inferred from x-ray and amino acid sequencing data. It was clear from these discussions that, in addition to the necessary but more or less straightforward tasks of data reduction, Fourier analysis and model refinement, there was also a considerable amount of heuristic data interpretation involved in structure determination. The model builder, for example, is often faced with a number of possible substructures which are consistent with an electron density map, and must base his choice on "rules of thumb" as well as principles of chemistry and geometry. The task domain seemed well suited for the application of heuristic programs which could generate plausible hypotheses about a protein's structural elements and test these hypotheses against the crystallographic data.

Professor Kraut suggested that our efforts would yield a high payoff if we could somehow eliminate any of the main bottlenecks in determining protein structures. A major bottleneck is obtaining phase data, which is required to construct an electron density map of the molecule. These data are usually obtained by the process of multiple isomorphous replacement (MIR), in which heavy atoms are implanted in the crystallized molecule. The determination of many protein structures has been delayed for months to years because of the difficulty in obtaining MIR data.

Kraut suggested that a way to eliminate this bottleneck is to use the parent protein data only, in con junction with all the "non-crystallographic" knowledge which the expert would bring to bear on each specific problem. For example, the "unknown" protein is often one member of a family of similar proteins having known characteristic structural features. We assume that one or more of these features is present and test that assumption against the data. This is done readily by first reducing the data to a function of the three crystallographic space variables. This function, the Patterson function, has a simple physical interpretation, which facilitates the verification process.

Havrng delineated the task domain, we continued to work closely with our UCSD colleagues, and developed a program, PSRCH, whose main purpose is to test the feasibility of inferring structure without phase information. We have thus far applied the program to data obtained from two proteins whose structures are already known. In one case we searched for a characteristic tetrahedral structure of iron and sulfur in the protein called HIPIP, by starting with its known relative coordinates and looking for the orientation(s) and posittons in the unit cell which give the best confirmation of the experimental data. The search was successful; however, the task was an easy one and we could only conclude that the procedure might work on more subtle cases. We then moved on to a slightly more difficult case, searching for the position of the heme structure in the protein cytochrome C?. (Incidentally, HIPIP and cytochrome C2 are two proteins whose structures were first discovered and reported on by the UCSD group. There are currently only about three dozen proteins whose complete, i.e., tertiary, structures are known today.) Here, too, tt was possible to find the orientation of the heme group properly. The translation search yielded several positions which were highly confirmed by the data, including the correct one.

At this potnt in our research we entered into discussions with a member of the Computer Applications to Research section of the National Science Foundation, which led to a proposal submission on May 31, 1974. A nearly identical proposal was also sent to the National Institutes of Health in September. Extracts of the research plan, namely our objectives and methods of procedures, are given below.

Cotnputer networking has been and will continue to be an important component of our collaborations with the UCSD group. Until recently, we were using our program on the IBM 370/158 at the RAND Corporation in Santa Monica via the ARPA network. The UCSD group also has access to the RAND Computation Center through their ARPA network connection. We were thus able to exercise the program jointly, peruse the stored output on other files, or simply use the network as a communication facility for rapid interchange of ideas and results. Computations have now been shifted to the new SUMEX facility (a PDP-10I), located at the Stanford Medical School. SUMEX is a national resource sponsored by NIH for use in applying Artificial Intelligence to problems of biomedical interest. SUMEX is also accessible to the UCSD group, as well as others, by a network connection. SUM ES will provide us with computation only. Our ability to proceed with the work will, of course, depend on the continuation of support for the scientists who are designing and Implementing the programs.

3.4.2 Objectives

The objective of the research proposed here is to apply problem solving techniques, which have emerged from artificial Intelligence (AI) research, to the well known "phase problem" of x-ray crystallography, in order to determine the three-dimensional structures of proteins. The work is intended to be of practical as well as theoretical value to both computer science (particularly AI research) and protein crystallography. Viewed as an AI problem our objectives will be:

1. To discover from expert protein crystallographers the knowledge and heuristics which could be used to infer the tertiary structure of proteins from x-ray crystallographic data, and to formalize this knowledge as computer data structures and heuristic procedures.

2. To discover a program organization and a set of representations which will allow the knowledge and the heuristics to cooperate in making the search efficient, i.e., generating plausible candidates in a reasonable time. (This is a central theme of current artificial intelligence research.)

3. To Implement the above in a system of computer programs, the competence of which will have a noticeable impact upon the discipline of protein crystallography.

3.4.3 Methods of Procedure

Our task can be defined at two levels. At the application level the goal is to assist protein crystallographers in overcoming the phase problem in x-ray crystallography. We propose to do this by developing a computer program which Infers plausible "first guess" structures, in the absence of phase information, by applying as much general and task-specific knowledge as possible to constrain the search for a structure consistent with the x-ray intensity data.

At the computer science level, our task is to discover a program organization and a set of representations which can effectively utilize a large and disparate body of knowledge in conjunction with experimental data. The program must allow the various facts, procedures and heuristics, which the experts themselves routinely employ, to cooperate in making the search efficient, i.e., generating plausible candidates in a reasonable time.

The problem of organizing and utilizing a non-homogeneous body of knowledge, a central problem in current Artificial Intelligence research is particularly acute in protein crystallography. Generally speaking, we can divide our overall knowledge into three categories: 1) rules and relationships, i.e., knowledge and heuristics for which there are no well-defined algorithmic procedures; 2) rules and relationships expressed as algorithmic proceclures; and 3) data. The accompanying table shows some members of each category:

KNOWLEDGE
- Amino Acid Sequence-structure correlation
- Symmetry Information
  a) crystallographic
  b) non-crystallographic
- Stereochemical constraints
- Mathematical relationships among structure factors
- When to use which procedures

PROCEDURES
- Patterson search
- Rotation Function
- Superposition
- Trial and Error
- Anomalous dispersion Patterson
- Direct methods

DATA
- X-ray intensities
- Amino Acid Sequence
- Other chemical data
- Coordinates of related molecules if available
- Existence of prosthetic groups or cofactors
- Space group and cell dimensions

These varied sources of information are expressed in an equally varied set of representations. Knowledge about sequence-structure correlations is expressed in terms of amino acid sequences and macro-descriptions of structures (alpha-helix, beta-sheet). Symmetry relationships are usually defined by rotation and translation operators applied to coordinate vectors. Stereochemical constraints are usually expressed in terms of standard bond lengths and angles, allowed values for the (phi, psi) dihedral angles, minimum acceptable distances for non-bonded contacts. Among the various procedures used to extract information from the data we find that the Patterson search technique works in vector space, the rotation function in reciprocal space, superposition methods in vector space and its own superposition space, electron density interpretation in direct space, and so forth. The data as welt are comprised of tables and lists defined in different domains.

We wish to bring as much knowledge to bear on the problems as we have available, just as a practicing protein crystallographer would do. Therefore, we must have the ability to take information obtained by operating on the data base with a particular procedure and communicate it to another procedure even if these two procedures work with different representations of the world. We believe this problem can be solved by careful system design, as is discussed in the followrng section.

3.4.4 Overall design of the program

One approach to protein structure determination would be to write a battery of programs, each of which had a specific capability -- a Patterson interpretation program, a superposition program, etc. The investigator would esamine the results from one of these programs, decide what to do nest, make appropriate transformations of the data to conform with the input requirements of the next program, and submit the next job. This process, although conceptually straightforward, has several drawbacks:

1) There is a great deal of manual shuttling back and forth between programs, with the concomitant task of re-representation of input.
2) It is difficult to assess the value of each program in advancing toward a solution.
3) The ability of the individual programs to cooperate in an iterative fashion is limited by the investigator's stamina and willingness to keep the iterations going.
4) The manual control of the sequence of programs used Increases the probability of errors in data transference.
5) Unless very careful records are kept, it will be difficult to trace the reasoning process which led to the solution.
6) As new heuristics are elicited from experts, it may be necessary to incorporate them in several different programs.

Another approach is to adopt the program organization used in the HEARSAY system [1] where cooperating "experts" work in a quasi-parallel fashion to build the hypothesis toward a complete solution. Figure 2 shows how this program structure might look for our application; it is essentially isomorphic to figure I shown for the HEARSAY organization. Instead of "recognizers" we have substituted "analysts", experts in applying a specialized technique to the data at hand in order to propose and verify a partial structure. At the left of the figure are well-established pre-processing routines which can reduce the data and make the transformations into forms that can be used by the analyst programs. Fot HEARSAY's lexicon, we have substituted our own dictionary of superatoms, i.e., a polyatomic structure which is considered as a unit. Examples are alpha helices, the amino acid residues, heme group, and beta sheets. The controller at the bottom of the figure plays the same role as ROVER, the recognition overlord in HEARSAY. The controller can examine the list of current best hypotheses and pass control to the appropriate analyst for further synthesis of a structure or verification of an extant structure.

Although the representations of knowledge required by the various analysts may be incompatible, they communicate through a standardized representation of the hypotheses which they can generate and verify. The hypothesis may be thought of as a global data base which communicates information between the analysts, as shown schematically in the figure. The hypothesis is a partial structure, which may be represented by a list of atomic coordinates plus a description of allowed and forbidden regions of occupancy of the unit cell. We have not yet settled on a single representation; it is currently under study.

The particular analysts shown in the figure are only a representative subset of the full panoply that can eventually be added. The addition of a new expert to the system would be relatively straightforward. The new program could be written and tested Independently, providing the designer adopts the standard hypothesis representation. To merge the analyst with the full system would require adding new rules to the controller's scheduling heuristics. The controller is driven by a table of situation-action rules, as in the planning phase of Heuristic DENDRAL.

Although we have used the structure of the HEARSAY program to guide the organization of our protein structure inference program, there will likely be some significant differences even at the schematic level shown in the two figures. For example, not all analysts will contain both an hypothesizer and a verifier. Some analytical techniques are capable of one or the other but not both. Also, the general knowledge box shown at the top of figure 1 may contain subcategories of information which are not compatible with all of the underlying analysts. These changes should not Interfere with the basic idea of building an hypothesis by a set of cooperating specialized procedures, under the coordination of a rule-driven executive.

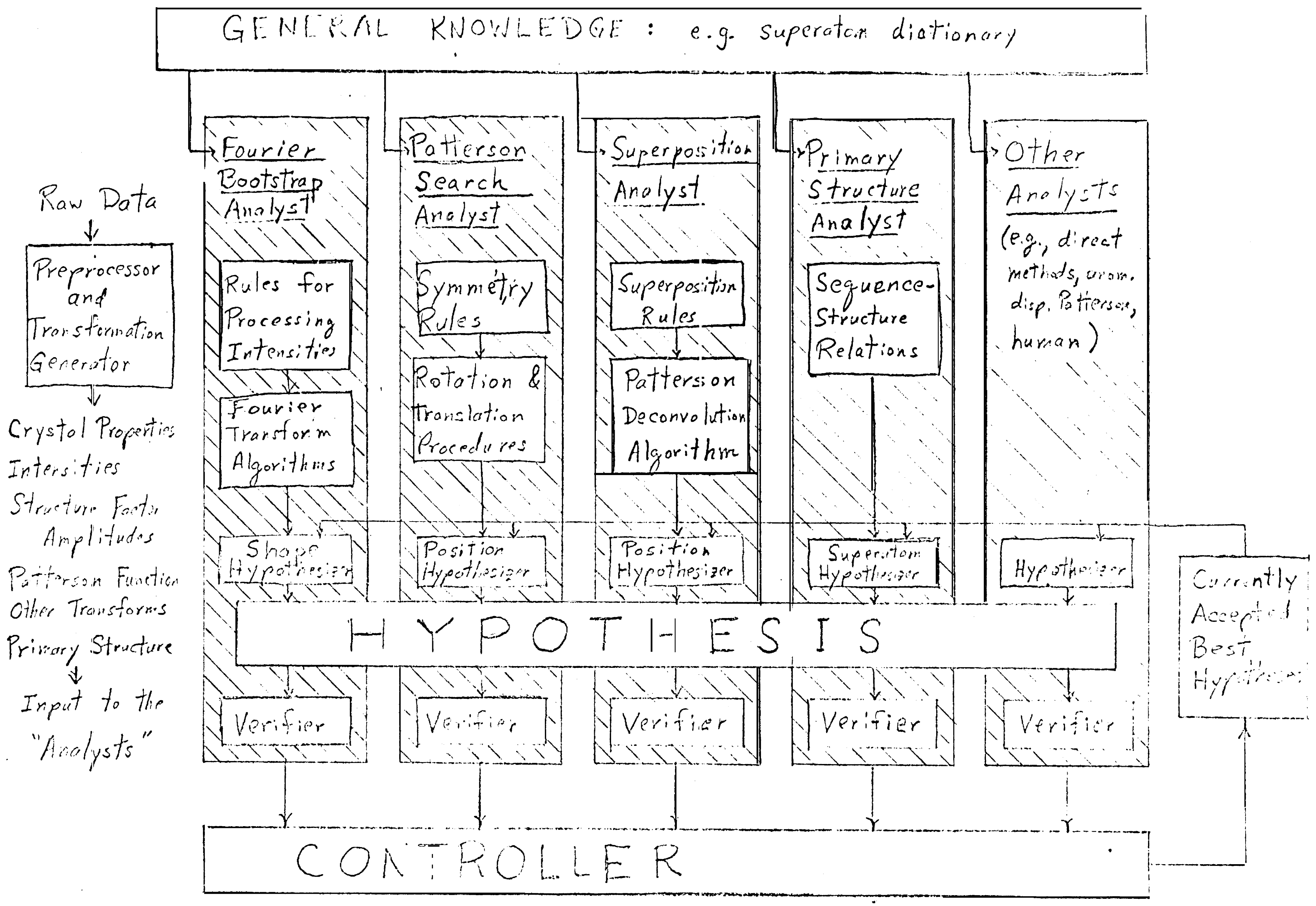

GENERAL KNOWLEDGE : e.g. superatom dictionary
Raw Data
Preprocessor and Transformation Generator
Crystal Properties
Intensities
Structure Factor Amplitudes
Patterson Function
Other Transforms
Primary Structure
Input to the "Analysts"
Fourier Bootstrap Analyst
Rules for Processing Intensities
Fourier Transform Algorithms
Shape Hypothesizer
Verifier
Patterson Search Analyst
Symmetry Rules
Rotation & Translation Procedures
Position Hypothesizer
Verifier
Superposition Analyst
Superposition Rules
Patterson Deconvolution Algorithm
Position Hypothesizer
Verifier
Primary Structure Analyst
Sequence-Structure Relations
Superatom Hypothesizer
Verifier
Other Analysts (e.g., direct methods, anom. disp. Patterson, human)
Hypothesizer
Verifier
HYPOTHESIS
CONTROLLER
Currently Accepted Best Hypothesis

Fig. 2. Detail of the Structure Inference Process in the Protein Structure Determination Program

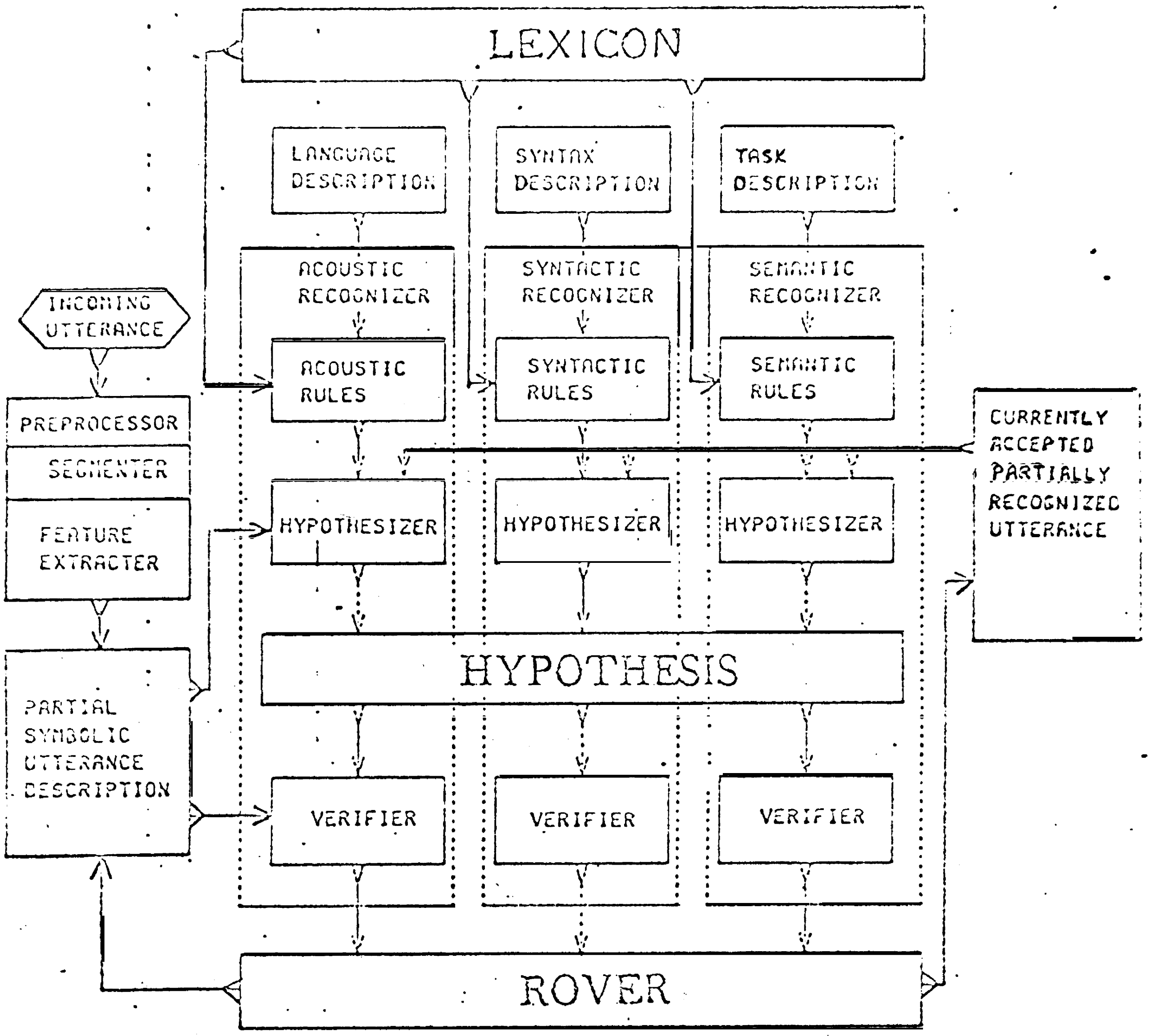


Fig. 1. Detail of the Recognition Process in the CMU-HEARSAY1 System

Bibliography

[1] D. R. Reddy, L. D. Erman, R. Neely, A Model and a System for Machine Recognition of Speech, *IEEE* Trans. *Audio & Electro-acoustics*, AU-? I, *1973*.

3.5 Knowledge Deployment Research: Inexact Reasoning

Often the knowledge acquired from Experts is intrinsically imprecise, i.e., although it accurately captures the Expert's best understanding of the situation, his understanding is imprecise. By what processes can a program be made to reason with such knowledge of a domain?

The intuitive and Inexact aspects of reasoning are described by Helmer and Rescher [1] who assert that the traditional concept of 'exact' versus 'Inexact science, with the social sciences accounting for most of the second class, has relied upon a false distinction usually reflecting the presence or absence of mathematical notation. They point out that only a small portion of natural science can be termed exact -- areas such as pure mathematics and subfields of physics in which some of the exactness "has even been put to the ultimate test of formal axiomatization". In several areas of applied natural science, on the other hand, decisions, predictions, and explanations are only made after exact procedures are mingled with unformalized expertise. Society's general awareness regarding these observations is reflected in the common references to the 'artistic' components in science.

In a recent paper (submitted to Mathematical Biosciences) we examine the nature of such nonprobabilistic and unformalized reasoning processes, consider their relationship to formal probability theory, and propose a model whereby such incomplete 'artistic' knowledge might be quantified. We have developed this model of inexact reasoning in response to the needs of AI Knowledge-based systems. That is, the goal has been to permit the opinion of experts to become more generally usable by programs and thus more available to nonexperts. The model is, in effect, an approximation to conditional probability. Although conceived with one problem area in mind, it is potentially applicable to any domain in which real world knowledge must be combined with expertise before an informed opinion can be obtained to explain observations or to suggest a course of action.

Although conditional probability in general, and Bayes' Theorem in particular, provides useful results in decision making, vast portions of technical experience suffer from so little data and so much imperfect knowledge that a rigorous probabilistic analysts, the ideal standard by which to judge the rationality of decisions, is not possible. It is nevertheless instructive to examine models for the less formal aspects of decision making. Experts seem to use an ill-defined mechanism for reaching decisions despite a lack of formal knowledge regarding the interrelationships of all the variables that they are considering. This mechanism is often adequate, in well-trained or experienced individuals, to lead to sound conclusions on the basis of a limited set of observations.

A conditional probability statement is, in effect, a statement of a decision criterion or rule. For example, the expression $P(H|E)=X$ can be read as a statement that there is a 100X% chance that hypothesis H is true given evidence E. The value of X for such rules may not be obvious (e.g., "y strongly suggests that z is true" is difficult to quantify), but an expert may be able to offer an estimate of this number based upon experience and general knowledge, even when such numbers are not readily available otherwise.

A large set of such rules obtained from textbooks and experts would clearly contain a large amount of task-specific knowledge useful to an intelligent program. It is conceivable that a computer program could be designed to consider all such general rules and to generate a final probability of each H based upon evidence gathered in a specific situation.

Programs that mimic the process of analyzing evidence incrementally often use this version of Bayes' Theorem:

Let El be the set of all observations to date, and e be some new piece of data. Furthermore, let E be the new set of observations once e has been added to El. Then the probability of hypothesis H on the combined evidence is expressed as:

$$P(H|E) = \frac{P(e \mid H \wedge E1)\ P(H \mid E1)}{\Sigma\ P(e \mid Hi \wedge E_i)\ P(Hi \mid E1)}$$

The successful programs that use Bayes' Theorem in this form require huge amounts of statistical data, not merely P(H | ek) for each of the pieces of data, ek, in E, but also the interrelationships of the ek within each hypothesis H,.

Bayes' Theorem would only be appropriate for such a program, however, if values for $P(e_1 \mid Hi)$ and $P(e_1 \mid Hi \wedge e_2)$ could be obtained. These requirements become unworkable, even if the subjective probabilities of experts are used, in cases where a large number of hypotheses must be considered. The first would require acquiring the inverse of every rule, and the second requires obtaining explicit statements regarding the Interrelationships of all rules in the system.

In short, we would like to devrse an approximate method that allows us to compute a value for P(Hi | E) solely in terms of $P(H_i \mid ek)$, where E is the composite of all the observed e's. Such a technique will not be exact, but since the conditional probabilities reflect judgmental (and thus highly subjective) knowledge, a rigorous application of Bayes' Theorem will not necessarily produce accurate cumulative probabilities either. Instead we have sought (a) ways to handle decision rules as discrete packets of knowledge, and (b) a quantification scheme that permits accumulation of evidence in a manner that adequately reflect; the reasoning process of an expert using the same or similar rules.

We believe that the proposed model is a plausible representation of the numbers an expert gives when asked to quantify the strength of his judgmental rules. We call these numbers "certainty factors," or CF's. He gives a positive number (CF>0) if the hypothesis is confirmed by observed evidence, suggests a negative number (CF<0) if the evidence lends credence to the negation of the hypothesis, and says there is no evidence at all (CF=0) if the observation is independent of the hypothesrs under consideration. The CF combines knowledge of both P(h) and P(h | e). Since the expert often has trouble stating P(h) and P(h|e) in quantitative terms, there is reason to believe that a CF that weights both the numbers into a single measure is actually a more natural intuitive concept (e.g., "I don't know what the probability is that all ravens are black, but 1 DO know that every time you show me an additional black raven my belief is increased by X that all ravens are black.")

In accordance with subjective probability theory, we assert that the expert's personal probability P(h) reflects his belief in h at any given time. Thus 1-P(h) can be viewed as an estimate of the expert's Disbelief regarding the truth of h. If P(h | e) is greater than P(h), the observation of e increases the expert's Belief in h while decreasing his Disbelief regarding the truth of h. In fact, the proportionate decrease in Disbelief is given by the ratio:

$$\frac{P(h \mid e) - P(h)}{1 - P(h)}$$

We call this ratio the measure of increased Belief in h resulting from the observation of e, i.e., M B[h,e].

Suppose, on the other hand, that P(h|e) were less than P(h). Then the observation of e would ciecrease the expert's Belief in h while increasing his Disbelief regarding the truth of h. The proportionate decrease in Belief is in this case given by the ratio:

$$\frac{P(h \mid e) - P(h)}{1 - P(h)}$$

We call this ratio the measure of increased Disbelief in h resulting from the observation of e, i.e., MD[h,e].

In addition, we define a third measure, termed a certainty factor (CF) that combines the MB and MD in accordance wrth the followrng definition:

$$\frac{P(h) - P(h \mid e)}{P(h)}$$

The certainty factor thus is an artifact for combining degrees of Belief and Disbelief into a single number. Such a number is needed in order to facilitate comparisons of the evidential strength of competing hypotheses. The use of this composite number is described in greater detail in the paper.

Bibliography

[I] Helmer, O., N. Rescher, On the Epistemology of the Inexact Sciences, Project RAND Report R-353, RAND Corp., Santa Monica, Cal., February 1960.

3.6 Knowledge Deployment Research for Real-World Applications: The Problem of Explaining a Program's Reasoning

As AI's research in knowledge-based systems moves toward application to real-world problems, it becomes essential for the intelligent agents so constructed to be able to explain to their users the knowledge and inference paths used in solving problems (or suggesting solutions). Without this ability, it is our belief that AI systems will not receive widespread acceptance. Nor will it be possible adequately to "debug" the knowledge in the systems' knowledge bases.

To conduct this research, we turned once more to the specific task domain and program (MYCIN) for diagnosis and treatment of infectious disease (an NIH-sponsored effort).

Recent progress in the development of the MYCIN system has included the development of a capability for providing sophisticated explanations of the program's reasoning steps.

Several aspects of the implementation of MYCIN facilitate the accomplishment of this goal -- the modularity of the program's rules simplifies the task of maintaining a record of the program's chain of reasoning, while the use of an interpretive language like LISP makes feasible the examination by the program of its own knowledge base, as well as the translation of the rules into English for display to the user. This ability of the program to keep track of Its reasoning and to examine its own knowledge and data is the central component in its ability to explain itself.

MYCIN normally takes the initiative during a consultation session; the system asks q uestions and uses the answers to determine the applicability of the decision rule it has retrieved. The user who desires an explanation of the program's motivation for a particular question has available to him a set of commands designed to make the examination of the program's reasoning both simple and effective.

3.6.1 WHY Questions -- Looking at Goals

Since any question is the result of an attempt to determine the truth of preconditions of a given subgoal, the simplest explanation of the motivation for a question is a statement of the current subgoal. By typing WHY, the user will get a detailed explanation from the system of the type of conclusion it is trying to draw, and how the current rule is to be applied in this case to establish that conclusion. Note that the system first examines its current reasoning chain to determine the "purpose" of the question, then examines the current rule to determine how it applies in this particular case, and finally translates all of this information from its Internal LISP representation Into unclerstandable English.

The user may understand why any particular question was asked, but may be unsure as to the program's reason for seeking the conclusion mentioned. He can examine this next step in the reasoning by simply repeating "WHY". This process can be repeated as often as desired, until the entire current reasoning chain has been displayed.

One problem we anticipated in the use of the WHY command, and one that is common with explanations in general, is the issue of presenting an explanation with the appropriate level of sophistication. Depending on the user, we might want to (a) display explicitly all steps in the chain of reasoning, (b) omit those which are definitional or trivial, or perhaps, or the most sophisticated user, (c) display only the highlights and allow him to supply the details.

We have provided this capability by allowing the user to indicate his level of sophistication with an optional argument to the WHY command. This parameter indicates how large a step in the reasoning process must be before it is to be displayed. Once again, this can be repeated as often as necessary, allowing the user to follow the reasoning chain in step sizes of his own choosing.

3.62 HOW questions: Looking at Preconditions

We have seen that as the user examines the current reasoning chain, he is informed of the various subgoals the system needs to achieve in order to accomplish the main goal. At some point he may wish to examine all the ways any subgoal may be achieved. For this examination of additional reasoning chains, he can use the HOW command.

The query allows the user to find out: (a) how a rule WAS used in the reasoning, i.e., what was known at the time the rule was Invoked and what conclusions were drawn as a result; (b) how a rule WILL BE used in the reasoning, i.e., what will have to be known for the rule to be invoked and what conclusion will be drawn, and (c) how a fact was determined that allowed some inference to be made.

Two points should be noted about the design of the program which generates these explanations. First, consistent with the general philosophy of MYCIN, the approach is quite domain-Independent. Although we have written programs with explicit knowledge of what is required for acceptable explanations, all task-specific knowledge is obtained by referring to the information stored in the separate knowledge base of rules.

Second, in attempting to supply information to the user, the system examines its own actions and knowledge base in order to discover what in fact it is "trying to do". The explanation program thus "keeps watch" over the actions of the consultation program by keeping a record of all of its past actions and mimicking its normal control structure when examining possible future actions.

3.7 Knowledge Representation: Production System

"The problem of representation of knowledge for AI systems is this: if the user has a fact about the world, or a problem to be stated, in what form does this become represented symbolically in the computer for immediate or later use?"

The formal structures and techniques for representing knowledge being explored by our project are production rules and production systems, a topic also being pursued vigorously by the ARPA project at Carnegie-Mellon University. Production systems offer a "natural", highly flexible, and modular way of representing knowledge.__ The approach is highly promising but much more work by us and others needs to be done.

Judgmental rules of the form 'If A then B' are commonly found in text and handbooks of scientific and technical disciplines. These rules not only describe formal relationships of a discipline but rules of accepted practice and hints of suggestive relations as well. For these reasons, production systems are important vehicles for encoding an expert's inferential knowledge in Intelligent programs. They have been studied and used in such programs as Newell's PSC, Waterman's poker-learning program, Shortliffe's MYCIN program and the Heuristic DENDRAL hypothesis formation program.

In the Heuristic DENDRAL program, a table of productlon rules holds the knowledge needed to explain empirical data from a subfield of analytical chemistry (mass spectrometry in particular). Part of the sophistication of this program comes from separating the program's knowledge base from the routines that use the knowledge for problem solving. Also, the productions themselves are more general than simple conditional sentences, 'If A then B'. The antecedents may be Boolean combinations of complex predicates and the consequents may be either (1) one or more processes to execute when the antecedent is true, (2) default processes to execute when no special-case antecedents are true ("else" conditions), or (3) another production. Making the productlon schema recursive (I.e., allowing any consequent to be another production decreases the run time and increases the storage efficiency of the program because the more general predicates separating large classes of special cases need to be stated and checked only once. For example, the schema might be written as:

```
If A then
     If A1 then    (If A11 then B11)
                   (If A12 then B12)
          else B1
     If A2 then B2
     else B.
```

Or, less efficiently, it could be equivalently represented as a set of simple productions in the form:

```
If A ∧ A1 ∧ A11     then B11;
If A ∧ A1 ∧ A12     then B12;
If A ∧ A1           then B1; - [A11 ∧ A12
          are implicit in the ordering]
If A ∧ A2           then B2;
If A                then B.
```

The knowledge representation ideas developed in the context of Heuristic DENDRAL have been successfully mapped into another AI program, using a different domain of knowledge. E. Shortliffe, in consultation with members of the Heuristic Programming Project (but under other financial support), developed the MYCIN program for reasoning about antimicrobial therapy.

The knowledge base of the program is a table of productions supplied by an expert, containing definitions, "common sense" pieces of knowledge, general scientific knowledge and highly task-specific inference rules. MYCIN is another successful application of AI to a scientific discipline whose sophistication is derived partly from the flexibility of a production rule representation of knowledge.

Bibliography

[1] E. H. Shortliffe, S. C. Axline, B. G. Buchanan, T. C. Merigan, and S. N. Cohen, **An Artificial Intelligence Program to Advise Physicians Regarding Antimicrobial Therapy**, *Computers and Biomedical Research, 6, 544-560, 1973.*

[2] E. H. Shortliffe, S. G. Axline, B. G. Buchanan, S. N. Cohen, Design **Considerations** for a Program to Provide **Consultations in** Clinical **Therapeutics**, *Proc. Biomedical Symposium*, San Diego, February 1974.

[3] E. H. Shortliffe, B. G. Buchanan, A Model of **Inexact Reasoning in Medicine**. (submitted to *Mathematical Biosciences*).

[4] E. H. Shortliffe, R. Davis, S. G. Axline, B. G. Buchanan, C. C. Green, S. N. Cohen, **Computer-Based Consultations in Clinical Therapeutics: Explanation and** Rule **Acquisition** Capabilities of the MYCIN System, (submitted to *Computers and Biomedical Research).*

3.8 **Application** of AI Techniques to a Programmer's Task: Automatic Debugging

We regard the tasks confronting computer programmers to be especially interesting as potential applications of our AI techniques. In such tasks, the "Expert" and the "Computer Scientist" usually merge into one person, facilitating the development of complex knowledge bases.

The work on Automatic Programming has been done in the context of a Ph.D. Thesis on Automatic Debugging of Logical Program Errors. The long-term goal of this research is to provide a system which will detect, diagnose, and treat logical programming errors. The Detection phase of debugging involves the process of deciding that a problem Indeed exists in the program. The Diagnosis phase involves the process of isolating the problem. The Treatment phase involves the process of determining what correction is to be made in the program and making it. We make a distinction between three classes of errors: (A) Syntactic errors, (B) Semantic errors, and (C) Logical errors. A syntactic error occurs when the text of the program does not conform to the syntax rules of the programming language. A semantic error occurs when a syntactically correct program attempts during its execution operations whose results are not defined by the semantics of the language. A logical error occurs when a program which is syntactically and semantically correct gives results which are "wrong". A prototype system is now up which is capable of detecting a small but interesting class of bugs, and diagnosing and treating a subset of the detected bugs. The prototype has correctly repaired a 'real' bug in a 'real' program. (See Brown, 'Internal Progress Memo', available from Heuristic Programming Project).

The prototype system operates by requesting information from the user (programmer) describing features of the execution of his program. This description is then used to detect errors. More precisely, during the execution of the program, checks are made on the consistency of the actual program execution with the programmer's expressed intentions. If a discrepancy is detected, the diagnosis phase of the system is invoked. An attempt is then made to recognize what sort of error has occured, and to determine its probable source. If this is successful, the treatment phase uses this information to repair both the current executing environment and the source program. The execution of the program is then resumed in the new, hopefully correct, environment. The goals to be achieved in the short term are:

1. to discover a language for describing programs which:
   a. is easy and reasonably error-free.

b. is able to describe those features of programs most useful in debugging programs.

2. to provide a system which interacts with the user (programmer) to obtain a description of a particular program.
3. to provide a system which, given a program, its description, and sample data, can debug (or at least significantly help to debug) the program.

The effort in part 1 in designing a language for describing programs is closely related to other Automatic Programming research. This language (and its recognizer) need not be as rich or complete as a full-blown Automatic Programming language, since a dialog need only describe features of an existing program, instead of describing a program to be created. But the primitives of this descriptive language should be useful in a more complete language. The effort in part 2 is related to the "diagnosis and therapy" paradigm of the MYCIN system (mentioned earlier). In both systems, a dialog between the user and the system generates information about an Individual (patient/program). This information is then used to diagnose problems in the individual. The effort in part 3 involves creation of a knowledge base of common program pathologies. This work has a traditional AI flavor. The knowledge base must be structured in such a way so as to be easily modified (changes and additions), but yet be effective in accomplishing its given task. A DENDRAL-like production system is being considered for knowledge representation in the system being built.

3.9 Tools and Techniques

The major work during this period falls into the categories of maintenance and development of basic utilities. Various projects that were completed are:

TRANSOR

A TRANSOR package in INTERLISP for translating LISP/ 360 programs to INTERLISP was completed. Emphasis was on making the package complete (almost all programs would translate without user intervention) and in increasing the efficiency of the resultant code (often a straight-forward translation is not the most efficient).

COMPARE

A LISP program for comparing two INTERLISP files and discovering the differences. Similar to SRCCOM but tailored to LISP expressions. The algorithm involves heuristics about the most common types of transformations made in editing a program (extraction, switching expressions, insertions, changing function calls, etc) in a tree search for the set of "minimal" transformations from one file to the next.

SYSTEM DEBUGGING

An incompatibility between KA-10 and KI-10 TENEX was discovered which resulted in obscure problems in INTERLISP on the KI-10. The cause was determined after much investigation. Other similar problems with the interface of LISP and TENEX have also been investigated and fixed.

TENEX utilities

The following utilities were produced: READ, a simple minded READMAIL; SYSIN, a program for starting up INTERLISP sysout files without going through LISP first (later modified at BBN); OPSAMP, a program for monitoring the different instructions a program is executing.

HPRINT, a program for printing and reading back in circular structures, including user datatypes, arrays, hash tables, along with the interface to the INTERLISP file package and read table facility.

System maintenance

Solving the problems of transfering files between one TENEX site and another via tape where each site has a different convention for formatting information on tape involved a significant amount of time. Utilities from other TENEX sites were eventually brought over and put up on the SUMES-TENET facility.

3.10 Technology Transfer: Chemistry and Mass Spectrometry

The past year has seen heavy involvement by other granting agencies in research which was initiated by ARPA funding, or supported in part by this funding. This demonstrates the programs' high levels of performance in the task areas of chemistry and mass spectrometry. These programs are, or soon will be, available to members of the Stanford Chemistry Department and to a nationwide community of collaborators via the SUMEX computer facility and TYMNET and ARPANET.

Applications of and further research into programs arising from activities of rhe DENDRAL project have been funded by the Biotechnology Resources Branch, National Institutes of Health for a period of three years beginning May 1, 1974. In addition, two smaller grants have been awarded which support ancillary areas of research, again begun with ARPA support. There are (1) an NSF grant to Dr. Harold Brown to support further research into graph theory as applied to chemical problems, and (2) an NIH award to Prof. C. Djerassi and Dr. R. Carhart to support applications of DENDR AL programs to carbon- 13 nuclear magnetic resonance.

Three major AI programs have been planned, developed and transferred to working chemists. These are outlined below.

PLANNER - our efforts at modelling the processes of scientific inference resulted in a program for analysis of mass spectral data. This program has been successfully demonstrated in applications to important chemical problems in the steroid field. With NIH support we are extending the generality of this program and adding an interactive user interface.

STRUCTURE GENERATOR - As in every heuristic search program, an ex haustive legal move generator is central to our applications program. We have finished a complete and irredundant generator of chemical structures and recently added mechanisms for constraining the generation process.

The generator alone is a useful working tool for chemists with structure elucidation problems because it can build molecular graphs from inferred units much more thoroughly than a human chemist can. It has recently been successfully demonstrated to several groups of chemists, and is currently in use by members of the Stanford Chemistry Department. Work will continue under NIH support to improve the program's performance, as it represents a framework for more general applications of AI techniques to chemical structure problems.

DATA INTERPRETATION AND SUMMARY. The INTSUM program (for INTerpretation and SUMmary) was developed as the first stage of a program for modelling scientific theory formation. This program is applied to a large collection of mass spectral data in an attempt to uncover regular patterns of fragmentation across a series of related chemical compounds. It has proven, by itself, to be a powerful assistant to chemists in their interpretation of large quantities of data. As a result, an interactive version of this program is now available and is being applied to problems in the mass spectrometry laboratory.

3.11 Technology Transfer: to Biology and Medicine

For many years, ARPA contract support to this project for basic research in Intelligent Systems has been the seed from which has grown grant support from other federal agencies with different missions. For example, the research on Heuristic DENDRAL, initially supported by ARPA, was later supported by NIH (and recently renewed by NIH). The ARPA-supported work on knowledge-based systems led to NIH support for the development for a program to diagnose bacterial infections and advise treatment (the MYCIN program). NSF has Indicated considerable interest in funding the hypothesis formation program in protein crystallography, begun under ARPA support.

The most significant event of this type, involving the transfer of concepts and techniques developed under ARPA support to another area of science and another source of support, occurred during this period. The Co-principal Investigators of this project were successful in obtaining grant funds from the Biotechnology Resources Branch of NIH to establish a computing facility to satisfy not only the expanding needs of this project, the NIH-sponsored DENDRAL project, and the other · NIH-sponsored activity; but also the needs of an embryonic but rapidly growing national community of scientific work in the application of artificial intelligence techniques to Biology and Medicine. A computing facility (with staff) called SUMEX has been established at the Stanford Medical School. Its complement of equipment is similar to that at the ARPA-sponsored AI laboratories -- the main frame is a PDPIOI, operating under the TENEX operating system. It is currently connected to the TYMnet, and in the near future will be connected to the ARPAnet by a VDH connection.

SUMEX, as mentioned, will serve not only Stanford interests but also the interests of AIM (Artificial Intelligence in Medicine), a name given to a national community of investigators interested in applying the techniques of AI research to Medicine and Biology. Such investigators include, for example, professors and students at the Computer Science Department of Rutgers University; and Dr. K. Colby, now at UCLA. AIM is constituted to have its own Advisory Committee to allocate its portion of the SUMEX resource, and to advise NIH and the SUMEX Principal Investigator, Professor Lederberg, on the needs of the national community and on how best to satisfy those needs.

In considering the technology transfer aspects of SUMEX-AIM, it is important to note:

1. that a federal science funding institution that has traditionally been very conservative in its funding of advanced computer science research (NIH) -- certainly much more conservative than ARPA and other DOD agencies -- has been persuaded to take this major step at the computer science research frontier. The credit for this is in no small measure due to the massive evidence developed with ARPA support that such a step would have great payoff to the medical science community;
2. that the previous NIH computer funding policy -- of funding computer facilities for geographically local communities of interest (like "researchers at the Baylor University Medical School") -- has been changed to one that supports facilities for scientific communities of interest not necessarily geographically local. The credit for this is due primarily to the ARPAnet, and the networking concepts developed in conjunction with ARPAnet development.

3.12 Technology Transfer: to Military Problems

At ARPA's request one of the co-principal investigators was asked to investigate the applicability of the concepts and techniques developed in the DENDRAL project to a surveillance signal interpretation problem of considerable importance to the Defense Department. Since this work is of a classified nature, it is being performed not at Stanford University but at a local research company. However, the Heuristic Programmrng Project's work is of key Importance in shaping the development of that military application of artificial intelligence. Further details concerning this application can be obtained from Professor Feigenbaum or from Dr. J.C.R. Licklider of the ARPA IPT Office.

3.13 Publications of the Project, 1972/1974

Research Supported by ARPA and by NIH

Bibliography

[1] D.H. Smith, B. G. Buchanan, R.S. Engelmore, A.M. Duffield, A. Yeo, E.A. Feigenbaum, J. Lederberg, and C. Djerassi, Applications of Artificial Intelligence for Chemical Inference VIII, An approach to the Computer Interpretation of the High Resolution Mass Spectra of Complex Molecules. . Structure Elucidation of Estrogeuic Steroids, *Journal of the American Chemical Society, 94, 5962-5971* (1972).

[2] B.C. Buchanan, E.A. Feigenbaum, and N.S. Srid haran, Heuristic Theory Formation: Data Interpretation and Rule Formation, in *Machine Intelligence 7*, Edinburgh University Press (1972).

[3] Lederberg, J., Rapid Calculation of Molecular Formulas f rom Mass Values, *J. Chemical Education, 49, 6* 13 (1972).

[4] Brown, H., Masinter, L., Hjelmeland, L., Constructive Graph Labeling Using Double Cosets, *Discrete Mathematics, 7* (1974), I-30; also Computer Science Memo 318, (1972).

[5] B.C. Buchanan, Review of Hubert Dreyfus' What Computers Can't Do: A Critique of Artificial Reason, *Computing Reviews*, (January, 1973); also Stanford A. I. Memo AIM-181.

[6] D. H. Smith, B. G. Buchanan, R. S. Engelmore, H. Aldercreutz and C. Djerassi, Applications of Artificial Intelligence for Chemical Inference IX. Analysis of Mixtures Without Prior Separation as Illustrated for Estrogens, *J. American Chemical Society, 95, 6078* (1973).

[7] D. H. Smith, B. G. Buchanan, W. C. White, E. A. Feigenbaum, C. Djerassi and J. Lederberg, Applications of Artificial Intelligence for Chemical Inference X. Intsum. A Data Interpretation Program as Applied to the Collected Mass Spectra of Estrogeuic Steroids, *Tetrahedron, 29,* 3117 (1973).

[8] B. G. Buchanan and N. S. Sridharan, Rule Formation on Non-Homogeneous Classes of Objects, *Proc. Third International Joint Conference on Artificial Intelligence,* Stanford, California, August (1973); also Stanford A. I. Memo AIM-215.

[9] D. Michie and B. G. Buchanan, Current Status of the Heuristic DENDRAL Program for Applying Artificial Intelligence to the Interpretation of Mass Spectra, to appear in *Computers for Spectroscopy,* (ed. R.A.G. Carrington), Adam Hilger, London; also: University of Edinburgh, School of Artificial Intelligence, Experimental Programming Report No. 32 (1973).

[10] H. Brown and L. Masinter, An Algorithm for the Construction of the Graphs of Organic Molecules, *Discrete Mathematics*, (in press); also Stanford Computer Science Dept. Memo STAN-CS-73-361, May 1973.

[11] D. H. Smith, L. M. Masinter and N. S. Srldharan, Heuristic DENDRAL: Analysis of Molecular Structure, *Proc. NATO/CNNA Advanced Study Institute on Computer Representation and Manipulation of Chemical Information*, (W. T. Wipke, S. Heller, R. Feldmann and E. Hyde, eds.), John Wiley and Sons, inc., 1974.

[12] R. Carhart and C. Djerassi, Applications of Artificial Intelligence for Chemical Inference XI: The Analysis of C13 NMR Data for Structure Elucidation of Acyclic Amines, *J. Chem. Soc.* (Perkin II), 1753 (1973).

[13] L. Masinter, N. S. Sridharan, R. Carhart and D. H. Smith, Applications of Artificial Intelligence for Chemical Inference XII: Exhaustive Generation of Cyclic and Acyclic Isomers, submitted to *Journal of the American Chemical Society;* also Stanford A. I. Memo AIM-216.

[14] L. Masinter, N. S. Sridharan, R. Carhart and D. H. Smith, Applications of Artificial Intelligence for Chemical Inference XIII: An Algorithm for Labelling Chemical Graphs, submitted to *Journal of the American Chemical Society*

[15] N. S. Sridharan, Computer Generation of Vertex Graphs, Stanford CS Memo STAN-U-73-38 1, July 1973.

[16] N. S. Sridharan, et.al., A Heuristic Program to Discover Syntheses for Complex Organic Molecules, Stanford CS Memo STAN-CS-73-370, June 1973; also Stanford A. I. Memo AIM-205.

[17] N. S. Sridharan, Search Strategies for the Task of Organic Chemical Synthesis, Stanford CS Memo STAN-CS-73-39 I, . October 1973; also Stanford A. I. Memo AIM-217

[18] D. H. Smith, Applications of Artificial Intelligence for Chemical Inference XIV: The Number of Structural Isomers of $C*xN*yO*Z$, $x + y + z = 6$. *An Investigation of Some Intuitions of Chemists.*

[19] D. H. Smith, Applications of Artificial Intelligence for Chemical Inference XV, in preparation.

[20] D. H. Smith and R. E. Carhart, Applications of Artificial Intelligence for Chemical Inference XVI: On Structural Isomerism of Tricyclodecanes, to be submitted to *Journal of the American Chemical Society.*

[21] R. G. Dromey, B. G. Buchanan, D. H. Smith, J. Lederberg and C. Djerassi, Applications of Artificial Intelligence for Chemical Inference XVII: A General Method for Predicting Molecular Ions in Mass Spectra, submitted to *Journal of Organic Chemistry.*

Other references, relating to the MYCIN system, under NIH support.

[22] E. H. Shortliffe, S. G. Axline, B. G. Buchanan, T. C. Merigan, and S. N. Cohen, An Artificial Intelligence Program to Advise Physicians Regarding Antimicrobial Therapy, *Computers and Biomedical Research*, 6 (1973), 544-560.

[23] E. H. Shortliffe, S. G. Axline, B. G. Buchanan, S. N. Cohen, Design Considerations for a Program to Provide Consultations in Clinical Therapeutics, *Proc. Biomedical Symposium,* San Diego, February 1974.

[24] E. H. Shortliffe and B. C. Buchanan, A Model of Inexact Reasoning in Medicine, submitted *to Mathematical Biosciences.*

[25] E. H. Shortliffe, R. Davis, S. G. Axline, B. G. Buchanan, C. C. Green and S. N. Cohen, Computer-Based Consultations in Clinical Therapeutics: Explanation and Rule Acquisition Capabilities of the MYCIN System, submitted to *Computers and Biomedical Research.*

4. NETWORK PROTOCOL DEVELOPMENT PROJECT

4.1 Internetwork Protocol Design

During this period, a design for an experimental internetwork protocol was completed [1] and has been circulated both to IFIP WG 6.1 and to other interested ARPA research centers. In addition, an article describing the basic concepts was published in May 1974 [2]. An updated and more detailed design was prepared and circulated only to the sites participating in ARPA sponsored internetworking and is now undergoing further revision.

The participants in the internetworking experiment include the University College London under the direction of Prof. Peter Kirstein, Bolt Beranek and Newman under the direction of Dr. Jerry Burchfiel, and the Stanford Digital Systems Laboratory under the direction of Prof. V. Cerf. Plans were laid to connect a TENEX system at BEN with a PDP-9 at UCLA and with a PDP-11 at SU-DSL, all running the proposed Transmission Control Program (internetwork protocol). Concurrently an experiment was outlined between the National Physical Laboratory in England under the direction of Dr. Donald Davies and the IRIA research center neat Paris under the direction of Mr. Louis Pouzin. In the latter experiment, a Modula-I computer at NPL is to be connected to a CII 100-70 at IRIA running a protocol proposed by H. Zimmerman and M. Elie of IRIA.

An agreement was reached regarding a common basic addressing format for both protocols [3] and it is intended that the results of these two experiments will be used to settle on a final protocol which could be used to connect all 5 sites.

In a concurrent effort, plans were made to study the problem of connecting the TYMNET with the ARPANET using the protocol proposed in [1]. During the period of this report, only modest progress has been made in this effort, but enthusiasm for the project remained high. It is expected that more concrete progress will be made during the second year.

IFIP Working Group 6.1 met in June 1973 and the National Computer Conference in New York, in September of 1973 in Sussex as the NATO Conference on Computer Networks, and in January 1974 at the Seventh Hawaii International Conference on Systems Science. Plans were made to meet again at IFIP 74 in August 1974. WG 6.1 was reorganized into four subcommittees to make working together easier:

Committee	**Chairman**
Experiments	Prof. P. Kirstein
Protocols	Mr. L. Pouzin
Legal and Political Issues	Prof. F. Kuo
Social Issues	Dr. C. D. Shepard

In another step to make W.G. 6.1 as self supporting as possible, and in the wake of the reduced NIC services offered by ARPA after 1 July 1974, all W.G. 6.1 members were to pay for the cost of reproducing and mailing of committee notes and reports. It was expected that this move would also shrink the size of the group down to those who were seriously interested in the work.

4.2 PDP-11 Expansion

During January through March 1974, the PDP-11/20 installation was expanded using funds from the Joint Services Electronics Program sponsored jointly by the Army, Navy and Air Force. The PDP-11 facility now includes:

a) PDP-11/20 CPU with 28 K 16-bit words of memory (maximum allowed).
b) 5.6 M word Diablo 44 moving head dual platter disk. One disk is removable; each will hold 2.8 M words.
c) Unibus repeater to expand the number of Unibus slots available.
d) Four asynchronous terminal interfaces, two for hard-wired use and two for dial up modems. Two Anderson- Jacobsen modems and two Direct Access Arrangement telephone lines also installed.
e) One OMRON microprogrammed CRT terminal with 4K byte buffer memory.
f) One card reader (not new).
g) One upper case only printer (not new).
h) Two Dectape drives (not new).
i) One RS64 64K byte fixed head disk.
j) One 1024::: 1024 CRT (not new) with SU-DSL designed controller and two joysticks (latter two are new).
k) Three Texas Instruments Silent 700 portable terminals.
1) One 16 bit general purpose digital Interface for experimental device attachments.
m) One 50 Kbit/second modem with ARPANET VDH Interface for use with the ELF operating system (PDP-11 is connected by VDH to SRI IMP).

The ARPA contract pays for the rental of the Modems, TI terminals, and maintenance on the PDP-11 during the summer months; the Electrical Engineermg Department of Stanford University pays for maintenance during the rest of the academic year.

4.3 Software Systems

ELF

In January, an ELF I system was installed. It proved to be fairly reliable although it had a few bugs left. It did not support the Diablo Disk or the dial-up facilities. Nor did it have much of a File Transfer Protocol (text files from the net could be printed on the line printer). The ELF system was used intermittently during this period for access to the ARPANET, but owing to shared use of the equipment for academic projects, the ELF system was not up much of the time.

An attempt was made to integrate ELF with the Disk Operating System (DOS), but this proved impossible since DOS is configured for single user function and simultaneous use of DOS with ELF caused ELF to lose control of it critical interrupts. We investigated the possibrlity of a Virtual Machine system, but the PDP-11/20 does not have adequate hardware to support virtual memory or privileged instruction trapping needed for Virtual Machine Monitors. We concluded that only a PDP-11/40 with hardware modifications similar to those on the UCLA system would serve for such a Virtual Machine system and gave up that approach as too costly and time consuming. Consequently, the system still alternated between DOS and ELF usage.

File Transfer Protocol

During the summer of 1974, an FTP was written which would accept MAIL files from the network and print them on the line printer. The program was documented [4] and plans were made to extend the system to full FTP capability.

Simple Minded File System

As an aid to the ELF user community, we proposed to implement a simple minded file system which would permit ELF to read or write contiguous files on the disk. The detailed specification and implementation of this package was seriously delayed owing to lack of documentation of the new ELF II system to which SMFS was to be interfaced. ELF II did not arrive during this period, so only the basic SMFS design specification was written using DOS I/O calls as the model foı user level interface.

Bibliography

[1] Cerf, V. G. and R. E. Kahn, Towards Protocols for Internetwork Communication *IFIP W.G. 6.1 Document 33*, September 1973.

[2] Cerf, V. C. and R. E. Kahn, A Protocol for Packet Network Intercommunication, *IEEE Transactions on Communication*, Volume COM-22, No. 5, May 1974.

[3] Pouzin, L. (Revised V. Cerf), A Packet Format Proposal, *IFIP W.G. 6.1 Document 48*, January 1974.

[4] Haugland, T., An Implementation of the ARPANET File Transfer Protocol for ELF, *Stanford University Digital Systems Laboratory Technical Note* 46, July 1974.

[5] Cerf, V. and C. Sunshine, Protocols and Gateways for Interconnection of Packet Switching Networks, *Proceedings of the Seventh Hawaii International Conference on Systems Science,* January 1974.

[6] Cerf, V. An Assessment of Arpanet Protocols, *Proceedings of the Second Jerusalem Conference on Information Technology,* July 1974.

Appendix A

ACCESS TO DOCUMENTATION

This is a description of how to get copies of publications referenced in this report.

External Publications

For books, journal articles, or conference papers, first try a technical library. If you have difficulty, you might try writing the author directly, requesting a reprint. Appendix D lists recent publications alphabetically by lead author.

Artificial Intelligence Memos

Artificial Intelligence Memos, which carry an "AIM" prefix on their number, are used to report on research or development results of general Interest, including all dissertations published by the Laboratory. Appendix B lists the titles of dissertations; Appendix E gives the abstracts of recent A. I. Memos and instructions for how to obtain copies. The texts of some of these reports are kept in our disk file and may be accessed via the ARPA Network (see below).

Computer Science Reports

Computer Science Reports carry a "STAN-CS" prefix and report research results of the Computer Science Department. (All A. I. Memos published since July 1970 also carry Computer Science numbers.) To request a copy of a CS report, write to:

Documentation Services
Computer Science Department
Stanford University
Stanford, California 94306

The Computer Science Department publishes a monthly abstract of forthcoming reports that can be requested from the above address.

Film Reports

Several films have been made on research projects. See Appendix C for a list of films and procedures for borrowing prints.

Operating Notes

Reports that carry a SAILON prefix (a strained acronym for *Stanford A. I. Laboratory Operating Note*) are semi-formal descriptions of programs or equipment in our laboratory that are thought to be primarily of internal interest. The texts of most SAILONS are accessible via the ARPA Network (see below), Printed copies may be requested from:

Documentation Services
Artificial Intelligence Laboratory
Stanford University
Stanford, California 94306

Working Notes

Much of our working documentation is not stocked in hard copy form, but is maintained in the computer disk file. Such texts that are in public areas may be accessed from the ARPA Network (see below). Otherwise, copies may be requested from the address given just above.

Public File Areas

People who have access to the ARPA Network are welcome to access our public files. The areas of principal interest and their contents are a follows:

[BIB,DOC]	bibliographies of various kinds,
[AIM,DOC]	texts of some A. I. Memos,
[S,DOC]	texts of most SAILONs,
[UP,DOC]	user program documentation,
[H,DOC]	system hardware descriptions,
[11,DOC]	PDP-11 subsystem descriptions,
[P,DOC]	"people-oriented" files, including the lab phone directory.

Network Access

On the ARPA Network, our system is site 11 (decimal), alias SU-AI. We use a heavily modified DEC monitor. It types "." whenever it is ready to accept a command. All commands end with <carriage return>. To halt a program that is running, type <Control>C twice.

It is possible to examine the contents of public files without logging in. For example, if you wish to know the names of all files in an area called [P,DOC]. Just type:

```
DIR [P,DOC]
```

The system will then type the names of all such files, their sizes, etc.

To type out the contents of a text file, say "TYPE <file name>". For example, to type the contents of our telephone directory, say

```
TYPE PHONE.LST[P,DOC]
```

and be prepared for 18 pages of output.

There may be difficulty in printing files that use the full Stanford character set, which employs some of the ASCII control codes (1 to 37 octal) to represent special characters.

If your terminal has both upper and lower case characters, let. the monitor know by saying "TTY FULL". If you are at a typewriter terminal, you may also wish to type "TTY FILL", which causes extra carriage returns to be inserted so that the carriage has time to return to the left margin before the next line begins.

To get information on other services that are available, say "HELP ARPA" or just plain "HELP".

File Transfer

Files can also be transferred to another site using the File Transfer Protocol. Documentation on our FTP program is located in diskfile FTP.DCS[UP,DOC]. No passwords or account numbers are needed to access our FTP from the outside.

Appendix B

THESES

Theses that have been published by the Stanford Artificial Intelligence Laboratory are listed here. Several earned degrees at institutions other than Stanford, as noted. This list is kept in our system in diskfile THESES[BIB,DOC].

D. Raj Reddy, AIM-43
An Approach to Computer Speech Recognition by Direct Analysis of the Speech Wave,
Ph. D. in Computer Science,
September 1966. --

S. Persson, AIM-46
Some Sequence Extrapolating Programs: a Study of Representation and Modeling in Inquiring Systems,
Ph.D. in Computer Science, University of California, Berkeley,
September 1966.

Bruce Buchanan, AIM-47
Logics of Scientific Discovery,
Ph.D. in Philosophy, University of California, Berkeley,
December 1966.

James Painter, AIM-44
Semantic Correctness of a Compiler for an Algol-like Language,
Ph.D. in Computer Science,
March 1967.

William Wichman, AIM-56
Use of Optical Feedback in the Computer Control of an Arm,
Eng. in Electrical Engineering,
August 1967.

Monte Callero, AIM-58
An Adaptive Command and Control System Utilizing Heuristic Learning Processes,
Ph.D. in Operations Research,
December 1967.

Donald Kaplan, AIM-60
The Formal Theoretic Analysis of Strong Equivalence for Elemental Properties,
Ph.D. in Computer Science,
July 1968.

Barbara Huberman, AIM-65
A Program to Play Chess End Games,
Ph.D. in Computer Science,
August 1968.

Donald Pieper, AIM-72
The Kinematics of Manipulators under Computer Control,
Ph.D. in Mechanical Engineering,
October 1968.

Donald Waterman, AIM-74
Machine Learning of Heuristics,
Ph.D. in Computer Science,
December 1968.

Roger Schank, AIM-83
A Conceptual Dependency Representation for a Computer Oriented Semantics,
Ph.D. in Linguistics, University of Texas,
March 1969.

Pierre Vicens, AIM-85
Aspects of Speech Recognition by Computer,
Ph.D. in Computer Science,
March 1969.

Victor D. Scheinman, AIM-92
Design of Computer Controlled Manipulator,
Eng. in Mechanical Engineering,
June 1969.

Claude Cordell Green, AIM-96
The Application of Theorem Proving to Question-answering Systems,
Ph.D. in Electrical Engineering,
August 1969.

James J. Horning, AIM-98
A Study of Grammatical Inference,
Ph.D. in Computer Science,
August 1969.

Michael E. Kahn, AIM-106
The Near-minimum-time Control of Open-loop Articulated Kinematic Chains,
Ph.D. in Mechanical Engineering,
December 1969.

Joseph Becker, AIM-119
An Information-processing Model of Intermediate-Level Cognition,
Ph.D. in Computer Science,
May 1972.

Irwin Sobel, AIM-121
Camera Models and Machine Perception,
Ph. D. in Electrical! Engineering,
May 1970.

Michael D. Kelly, AIM-130
Visual Identification of People by Computer,
Ph.D. in Computer Science,
July 1970.

Gilbert Falk, AIM-132
Computer Interpretation of Imperfect Line Data as a Three-dimensional Scene,
Ph. D. in Electrical Engineering,
August 1970.

Jay Martin Tenenbaum, AIM-134
Accommodation in Computer Vision,
Ph.D. in Electrical Engineering,
September 1970.

Lynn H. Quam, AIM-144
Computer Comparison of Pictures,
Ph.D. in Computer Science,
May 1971.

Robert E. Kling, AIM-147
Reasoning by Analogy with Applications to Heuristic Problem Solving: a Case Study,
Ph.D. in Computer Science,
August 197 I.

Rodney Albert Schmidt Jr., AIM-149
A Study of the Real-time Control of a Computer-driven Vehicle,
Ph.D. in Electrical Engineering,
August 1971.

Jonathan Leonard Ryder, AIM-155
Heuristic Analysis of Large Trees as Generated in the Game of Go,
Ph.D. in Computer Science,
December 197 l.

Jean M. Cadiou, AIM-163
Recursive Definitions of Partial Functions and their Computations,
Ph.D. in Computer Science,
April 1972.

Gerald Jacob Agin, AIM-173
Representation and Description of Curved Objects,
Ph.D. in Computer Science,
October 1972.

Francis Lockwood Morris, AIM-174
Correctness of Translations of Programming Languages -- an Algebraic Approach,
Ph.D. in Computer Science,
August 1972.

Richard Paul, AIM-177
Modelling, Trajectory Calculation and Servoing of a Computer Controlled Arm,
Ph.D. in Computer Science,
November 1972.

Aharon Gill, AIM-178
Visual Feedback and Related Problems in Computer Controlled Hand Eye Coordination,
Ph.D. in Electrical Engineering,
October 1972.

Rutena Bajcsy, AIM-180
Computer Identification of Textured Visiual Scenes,
Ph.D. in Computer Science,
October 1972.

Ashok Chandra, AIM-188
On the Properties and Applications of Prograininirig Schenias,
Ph.D. in Computer Science,
March 1973.

Gunnar Rutger Grape, AIM-201
Model Based (Intermediate Level) Computer Vision,
Ph.D. in Computer Science,
May 1973.

Yoram Yakimovsky, AIM-209
Scene Analysis Using a Semantic Base for Region Growing,
Ph.D. in Computer Science,
July 1973.

Jean E. Vuillemin, AIM-2 18
Proof Techniques for Recursive Programs,
Ph.D. in Computer Science,
October 1973.

Daniel C. Swinehart, AIM-230
COPILOT: A Multiple Process Approach to Interactive Programming Systeins,
Ph.D. in Computer Science,
May 19'74.

James Gips, AIM-23 1
Shape Grammars and their Uses
Ph.D. in Computer Science,
May 1974.

Charles J. Rieger III, AIM-233
Conceptual Memory: A Theory and Computer Program for Processing the Meaning Content of Natural Language Utterances,
Ph.D. in Computer Science,
June 1974.

Christopher K. Riesbeck, AIM-238
Computational Understanding: Analysis of Sentences and Contest,
Ph.D. in Computer Science,
June 1974.

Marsha Jo Hannah, AIM-239
Computer Matching of Areas in Stereo Images,
Ph.D. in Computer Science,
July 19'74.

James R. Low, AIM-242
Automatic Coding: Choice of Data Structures,
Ph.D. in Computer Science,
August 1974.

Appendix C

FILM REPORTS

Prints of the following films are available for short-term loan to Interested groups without charge. They may be shown only to groups that have paid no admission fee. To make a reservation, write to:

Film Services
Artificial Intelligence Lab.
Stanford University
Stanford, California 94305

Alternatively, prints may be purchased at cost (typically $40 to $60) from:

Cine-Chrome Laboratories
4075 Transport St.
Palo Alto, California
(415) 321-5678

This list is kept in diskfile FILMS[BIB,DOC].

1. Art Eisenson and Gary Feldman, Ellis D. Kroptechev and Zeus, his Marvelous Time-sharing System, 16mm B&W with sound, 15 minutes, March 1967.

The advantages of time-sharing over standard batch processing are revealed through the good offices of the Zeus time-sharing system on a PDP- 1 computer. Our hero, Ellis, is saved -from a fate worse than death. Recommended for mature audiences only.

2. Gary Feldman, Butterfinger, 16mm color with sound, 8 minutes, March 1968.

Describes the state of the hand-eye system at the Artificial Intelligence Project in the fall of 1967. The PDP-6 computer getting visual Information from a television camera and controlling an electrical-mechanrcal arm solves simple tasks involving stacking blocks. The techniques of recognizing the blocks and their positions as well as controlling the arm are briefly presen ted. Rated "G".

3. Raj Reddy, Dave Espar and Art Eisenson, Hear Here, 16mm color with sound, 15 minutes, March 1969.

Describes the state of the speech recognition project as of Spring, 1969. A discussion of the problems of speech recognition is followed by two real time demonstrations of the current system. The first shows the computer learning to recognize phrases and second shows how the hand-eye system may be controlled by voice commands. Commands as complicated as 'Pick up the small block in the lower lefthand corner', are recognized and the tasks are carried out by the computer controlled arm.

4. Gary Feldman and Donald Peiper, Avoid, 16mm silent, color, 5 minutes, March 1969.

Reports on a computer program written by D. Peiper for his Ph.D. Thesis. The problem is to move the computer controlled electro-mechanical arm through a space filled with one or more known obstacles. The program uses heuristics for finding a safe path; the film demonstrates the arm as it moves through various cluttered environments with fairly good success.

5. Richard Paul and Karl Pingle, Instant Insanity, 16mm color, silent, 6 minutes, August, 1971.

Shows the hand/eye system solving the puzzle *Instant Insanity.* Sequences include finding and recognizing cubes, color recognition and object manipulation. This film was made to accompany a paper presented at the 1971 International Joint Conference on Artificial Intelligence in London and may be hard to understand without a narrator.

6. Suzanne Kandra, Motion and Vision, 16mm color, sound, 22 minutes, November 1972.

A technical presentation of three research projects completed in 1972: advanced arm control by R. P. Paul [AIM-177], visual feedback control by A. Gill [AIM-178], and representation and description of curved objects by G. Agin [AIM-173].

7. Larry Ward, **Computer Interactive Picture Processing**, (MARS Project), 16mm color, sound, 8 min., Fall 1972.

This film describes an automated picture differencing technique for analyzing the variable surface features on Mars using data returned by the Mariner 9 spacecraft. The system uses a time-shared, terminal oriented PDP- 10 computer. The film proceeds at a breath less pace. Don't blink, or you will miss an entire scene.

8. Richard Paul and Karl Pingle, Automated **Pump** Assembly, 16mm color, silent (runs at sound speed!), 7 minutes, April, 1973.

Shows the hand-eye system assembling a simple pump, using vision to locate the pump body and to check for errors. The parts are assembled and screws inserted, using some special tools designed for the arm. Some titles are Included to help explain the film.

9. Terry Winograd, Dialog **with** a robot, 16mm black and white, silent, 20 minutes, (made at MIT), 1971.

Presents a natural language dialog with a simulated robot block-manipulation system. The dialog is substantially the same as that in *Understanding Natural Language* (T. W inograd, Academic Press, 1972). No explanatory or narrative material is on the film.

10. Karl Pingle, Lou Paul and Bob Bolles, Automated Assembly, Three Short Examples, 1974 (forthcoming).

Appendix D

EXTERNAL PUBLICATIONS

Articles and books by Project members that have appeared in the last year are listed here alphabetically by lead author. Earlier publications are given in our ten-year report [Memo AIM-2281 and in diskfile PUBS.OLD[BIB,DOC]. The list below is kept in PUBS[BIB,DOC].

1. Agin, Gerald J., Thomas O. Binford, Computer Description of Curved Objects, *Proceetfings of the Third International Joint Conference on Artificial Intelligence*, Stanford University, August 1973.

2. Ashcroft, Edward, Zohar Manna, Amir Pnueli, Decidable Properties of Monodic Functional Schemas, *J. ACM*, July 1973.

3. Bajcsy, Ruzena, Computer Description of Textured Scenes, *Proc. Third Int. Joint Conf. on Artificial Intelligence*, Stanford U., 1973.

4. Brown, H., Masinter, L., Hjelmeland, L., Constructive Graph Labeling Using Double Cosets, *Discrete Mathematics*, 7, 1974.

5. Buchanan, Bruce, N. S. Sridharan, Analysis of Behavior of Chemical Molecules: Rule Formation on Non-Homogeneous Classes of Objects, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford University, August 1973.

6. Carhart, R., C. Djerassi, Applications of Artificial Intelligence for Chemical Inference XI: The Analysis of C13 NMR Data for Structure Elucidation of Acyclic Amines, *J. Chem. Soc.* (Perkin II), 1753, 1973.

7. Cerf, Vinton G., R. E. Kahn, A Protocol for Inter-network Communications, *IEEE Trans. Communications*, May 1974.

8. Cerf, V. G., D. D. Cowan, R. C. Mullin, R. G. Stanton, Networks and Generalized Moore Graphs, *Proc. Manitoba Conf. on Numerical Math., 1973*, (to appear).

9. Cerf, V. C., D. Cowan, R. C. Mullin, R. G. Stanton, Topological Design Considerations in Computer-Communication Networks, in R. L. Grimsdale, F. F. Kuo (eds.), *Computer Communication Networks*, Academic Book Services Holland, Netherlands, 1974.

10. Cerf, V., C. Sunshine, Protocols and Gateways for Interconnection of Packet Switching Networks, *Proc. 7th Hawaii International Conf. on System Sciences*, Western Periodicals Co., Hawaii, January 1974.

11. Cerf, V. G., R. E. Kahn, A Protocol for Packet Network Intercommunication, *IEEE Trans. Communication*, Vol. COM-22, No. 5, May 1974.

12. Cerf, V. G., D. D. Cowan, R. C. Mullin, R. G. Stanton, A Partial Census of Generalized Moore Graphs, *Proc. Australian National Combinitorics Conference*, May 1974.

13. Cerf, V. C., An Assessment of ARPANET Protocols, *Proc. Jerusalem Conf. on Information Technology*, July 1974.

14. Chowning, John M., The Synthesis of Complex Audio Spectra by means of Frequency Modulation, *J. Audio Engineering Society*, September 1973.

15. Colby, Kenneth M., *Artificial Paranoia: A Computer Simulation of the Paranoid Mode*, Pergamon Press, N.Y., 1974,

16. Colby, K.M. and Parkison, R.C. Pattern-matching rules for the Recognition of Natural Language Dialogue Expressions, *American Journal of Computational Linguistics*, 1, 1974.

17. Dobrotin, Boris M., Victor D. Schernman, Design of a Computer Controlled Manipulator for Robot Research, *Proc. Third Int. Joint Conf. on Artificial Intelligence,* Stanford U., 1973.

18. Enea, Horace, Kenneth Mark Colby, Idiolectic Language-Analysis for Understanding Doctor-Patient Dialogues, *Proceedings oj the Third International Joint Conference on Artificial Intelligence,* Stanford University, August 1973.

19. Feldman, Jerome A., James R. Low, Comment on Brent's Scatter Storage Algorithm, *Comm. ACM,* November 1973.

20. Hieronymus, J. L., N. J. Miller, A. L. Samuel, The Amanuensis Speech Recognit inn System, Pros. *IEEE Symposium on Speech Recognition,* April 1974.

21. Hieronymus, J. L., Pitch Synchronous Acoustic Segmentation, *Proc. IEEE Symposium on Speech Recognition,* April 1974.

22. Hilf, Franklin, Use of Computer Assistance in Enhancing Dialog Based Social Welfare, Public Health, and Educational Services in Developing Countries, *Proc. 2nd Jerusalem Conj. on Info. Technology*, July 1974.

23. Hueckel, Manfred 1-1., A Local Visual Operator which Recognizes Edges and Lines, *J. ACM,* October 1979.

24. Igarashi, S., R. L. London, D. C. Luckham, Interactive Program Verification: A Logical System and its Implementation, Acta Informatica, (to appear).

25. Katz, Shmuel, Zohar Manna, A Heuristic Approach to Program Verification, *Proceedings of the Third International Joint Conference on Artificial Intelligence,* Stanford University, August 1973.

26. Luckham, David C., Automatic Problem Solving, *Proceedings of the Third International Joint Conference on Artificial Intelligence,* Stanford University, August 1973.

27. Luckham, David C., Jack R. Buchanan, Automatic Generation of Programs Containing Conditional Statements, *Proc. AISB Summer Conference,* U. Sussex, July 1974.

28. Manna, Zohar, Program Schemas, in *Currents in the Theory of Computing* (A. V. Aho, Ed.), Prentice-Hail, Englewood Cliffs, N. J., 1973.

29. Manna, Zohar, Stephen Ness, Jean Vuillemin, Inductive Methods for Proving Properties of Programs, *Comm. ACM,* August 1973.

30. Manna, Zohar, Automatic Programming, *Proceedings of the Third International Joint Conference on Artificial intelligence,* Stanford University, August 1973.

31. Manna, Zohar, *Introduction to Mathematical Theory of Computation,* McGraw-Hill, New York, 1974.

32. Masinter, L., N. S. Sridharan, R. Carhart, D. H. Smith, Applications of Artificial Intelligence for Chemical Inference XII: Exhaustive Generation of Cyclic and Acyclic Isomers, *J. Amer. Chem. Soc.*, (to appear).

33. Masinter, L., N. S. Sridharan, R. Carhart, D. H. Smith, Applications of Artificial Intelligence for Chemical Inference XIII: An Algorithm for Labelling Chemical Graphs, *J. Amer. Chem. Soc.*, (to appear).

34. Michie, D., Bruce G. Buchanan, Current Status of the Heuristic DENDRAL Program for Applying Artificial Intelligence to the Interpretation of Mass Spectra, in R. A. G. Carrington (ed.), *Computers for Spectroscopy*, Adam Hilger, London, (to appear).

35. Miller, N. J., Pitch Detection by Data Reduction, *Proc. IEEE Symposium on Speech Recognition*, April 1974.

36. Moorer, James A., The Optimum Comb Method of Pitch Period Analysis of Continuous Speech, *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. ASSP-22, No. 5, October 19'74.

37. Jorge J. Morales, Interactive Theorem Proving, *Proc. ACM National Conference*, August 1973.

38. Nevatia, Ramakant, Thomas O. Binford, Structured Descriptions of Complex Objects, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford University, August 1973.

39. Quam, Lynn, Robert Tucker, Botond Eross, J. Veverka and Carl Sagan, Mariner 9 Picture Differencing at Stanford, *Sky and Telescope*, August 1973.

40. Sagan, Carl, J. Veverka, P. Fox, R. Dubrsch, R. French, P. Gierasch, L. Quam, J. Lederberg, E. Levinthal, R. Tucker, B. Eross, J. Pollack, Variable Features on Mars II: Mariner 9 Global Results, *J. Geophys. Res.*, 78, 4 163-4 196, 1973.

41. Veverka, J., Carl Sagan, Lynn Quam, R. Tucker, B. Eross, Variable Fcaturcs on Mars III: Comparison of Mariner 1969 and Mariner 19'71 Photography, *Icarus*, 21, 3 17-368, 1974.

42. Schank, Roger C., Neil Goldman, Charles J. Rieger III, Chris Riesbeck, MARGIE: Memory, Analysis, Response Generation and Inference on English, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford University, August 1973.

43. Schank, Roger C., Kenneth Colby (eds), *Computer Models of Thought and Language*, W. H. Freeman, San Francisco, 1973.

44. Shortliffe, E. H., S. C. Axline, B. G. Buchanan, T. C. Merigan, S. N. Cohen, An Artificial Intelligence Program to Advise Physicians Regarding Antimicrobial Therapy, *Computers and Biomedical Research 6, 544-560,* 1973.

45. Shortliffe, E. H., S. C. Axline, B. G. Buchanan, S. N. Cohen, Design Consultations in Clinical Therapudics, *Proc. Biomedical Symposium*, San Diego, February 1974.

46. Smith, D. H., B. G. Buchanan, R. S. Engelmore, H. Aldercruetz, C. D jerassi, Applications of Artificial Intelligence for Chemical Inference IX. Analysis of Mixtures without Prior Separation as Illustrated for Estrogens, *J. American* Chem. *Soc.*, Vol. 95, No. 18, page 6078, 1973.

4'7. Smith, D. H., B. C. Buchanan, W. C. White, E. A. Feigenbaum, J. Lederberg, C. Djerassi, Applications of Artificial Intelligence for Chemical Inference X. Intsum. A Data Interpretation Program as Applied to the Collected Mass Spectra of Estrogenic Steroids, *Tetrahedron*, Vol. 29, page 3117, 1973.

48. Smith, D. H., L. M. Masinter, N. S. Sridharan, Heuristic DENDRAL: Analysis of Molecular Structure, *Proc. NATO/CNNA Advanced Study Institute on Computer Representation and*

Manipulation of Chemical Information, John Wiley and Sons, 1974.

49. Smith, David Canfield, Horace J. Enea, Backtracking in MLISP2, *Proceedings of the Third International joint Conference on Artificial Intelligence,* Stanford University, August 1973.

50. Smith, Leland, Editing and Printing Music by Computer, *J. Music Theory,* Fall 1973.

51. Sobel, Irwin, On Calibrating Computer Controlled Cameras for Perceiving 3-D Scenes, *Proc. Third Int. Joint Conf. on Artificial Intelligence,* Stanford U., 1973; also in Artificial Intelligence J., Vol. 5, No. 2, Summer 1974.

52. Sridharan, N., Search Strategies for the Task of Organic Chemical Synthesis, *Proceedings of the Third International Joint Conference on Artificial Intelligence,* Stanford University, August 1973.

53. Tesler, Lawrence G., Horace J. Enea, David C. Smith, The LISP70 Pattern Matching System, *Proceedings of the Third International Joint Conference on Artificial Intelligence,* Stanford University, August 1973.

54. Wilks, Yorlck, The Stanford Machine Translation and Understanding Project, in Rustin (ed.) *Natural Language Processing, New* York, 1973.

55. . Wilks, Yorlck, Understanding Without Proofs, *Proceedings of the Third International Joint Conference on Artificial Intelligence,* Stanford University, August 1973.

56. Wilks, Yorlck, Annette Herskovits, An Intelligent Analyser and Generator of Natural Language, *Proc. Int. Conf. on Computational Linguistics,* Pisa, Italy, *Proceedings of the Third Internation Joint Conference on Artificial Intelligence,* Stanford University, August 1973.

57. Wilks, Yorick, The Computer Analysis of Philosophical Arguments, *CIRPHO,* Vol. 1, No. 1, September 1973

58. Wilks, Yorlck, An Artificial Intelligence Approach to Machine Translation, in Schank and Colby (eds.), *Computer Models of Thought and Language,* W. H. Freeman, San Francisco, 1973.

59. Wilks, Yorick, One Small Head -- Models and Theories in Linguistics, *Foundations of Language,* Vol. 10, No. 1, January 1974.

60. Wilks, Yorick, Preference Semantics, E. Keenan (ed.), *Proc. 1973 Colloquium on Formal Semantics of Natural Language,* Cambridge, U.K., 1974.

61. Wilks, Y. Semantic Procedures and Information, in *Studies in the Foundations of Communication,* R. Posner (ed.), Springer, Berlin, forthcoming.

62. Winograd, Terry, A Process Model of Language Understanding, in Schank and Colby (eds.), *Computer Models of Thought and Language,* W. H. Freeman, San Francisco, 1973.

63. Winograd, Terry, The Processes of Language Understanding in Benthall, (ed.), *The Limits of Human Nature,* Allen Lane, London, 1973.

64. Wmograd, Terry, Language and the Nature of Intelligence, in G.J. Dalenoort (ed.), *Process Models for Psychology,* Rotterdam Univ. Press, 1973

65. Wmograd, Terry, Breaking the Complexity Barrier (again), *Proc. SIGPLAN-SIGIR Interface Meeting,* 1973.

66. Winograd, Terry, Artificial Intelligence -- When Will Computers Understand People?, *Psychology Today*, May 1974.

67. Winograd, Terry, Parsing Natural Language via Recursive Transition Net, in Yeh (ed.) *Applied Computation Theory,* Prentice-Hall, 1974.

68. Yakimovsky, Yoram, Jerome A. Feldman, A Sewantics-Based Decision Theoretic Region Analyzer, *Proceedings* of *the Third International joint Conference on Artificial Intelligence,* Stanford University, August 1973.

Appendix E

A. I. MEMO ABSTRACTS

Abstracts are given here for Artificial Intelligence Memos that we have published in the last year. For earlier years, see our ten-year report [Memo AIM-2281 or diskfile AIMS.OLD[BIB,DOC]. The abstracts below are kept in diskfile AIMS[BIB,DOC] and the titles of both earlier and more recent A. I. Memos are in AIM LST[BIB,DOC].

In the listing below, there are up to three numbers given for each report: an "AIM" number on the left, a "CS" (Computer Science) number in the middle, and a NTIS stock number (often beginning "AD...") on the right. Special symbols preceding the "AIM" number indicate availability at this writing, as follows:

+ hard copy or microfiche,
⊗ microfiche only,
∗ out-of-stock.

If there is no special symbol, then it is available in hard copy only. Reports that are out-of-stock are likely to stay that way because of peculiar governmental contractual requirements. Reports that are in stock may be requested from:

Documentation Services
Artificial Intelligence Laboratory
Stanford University
Stanford, California 94305

Alternatively, reports may be ordered (for a nominal fee) in either hard copy or microfiche from:

National Technical Information Service
P. 0. Box 1553
Springfield, Virginia 22151

If there is no NTIS number given, then they may or may not have the report. In requesting copies in this case, give them both the "AIM-" and "CS-nnn" numbers, with the latter enlarged into the form "STAN-CS-yy-nnn", where "yy" is the last two digits of the year of publication.

Memos that are also Ph.D. theses are so marked below and may be ordered from:

University Microfilm
P. 0. Box 1346
Ann Arbor, Michigan 48106

For people with access to the ARPA Network, the texts of some A. I. Memos are stored online in the Stanford A. I. Laboratory disk file. These are designated below by "Diskfile: <file name>" appearing in the header.

∗AIM-211 CS-383 AD769673
Yorick Wilks,
Natural Language **Inference,**
24 pages, September 1973.

The paper describes the way in which a Preference Semantics system for natural language analysis and generation tackles a difficult class of anaphoric inference problems (finding th correct referent for an English pronoun in context): those requiring either analytic (conceptual) knowledge of a complex sort, or requiring weak inductive knowledge of the course of events in the real world. The method employed converts all available knowledge to a canonical template form and endeavors to create chains of non-deductive inferences from the unknowns to the possible referents. Its method of selecting among possible chains of inferences is consistent with the overall principle of 'semantic preference' used to set up the original meaning representation, of which these anaphoric inference procedures are a manipulation.

AIM-Z12 cs-3s4 AD769379
Annette Herskovits,
The **Generation** of **French** from a **Semantic Representation,**
20 pages, September 1973.

The report contains first a brief description of Preference Semantics, a system of representation and analysis of the meaning structure of natural language. The analysis algorithm which transforms phrases into semantic items called **templates** has been considered in detail elsewhere, so this report concentrates on the second phase of analysis, which binds templates together into a higher level semantic block corresponding to an English paragraph, and which, in operation, interlocks with the French generation procedure. During this phase, the semantic relations between templates are extracted, pronouns are referred and those word disambiguations are done that require the context of a whole paragraph. These tasks require items called *paraplates* which are attached to keywords such as prepositions, subjunctions and relative pronouns. The system chooses the representation which maximizes a carefully defined 'semantic density'.

A system for the generation of French sentences is described, based on the generation of French sentences is described, based on the recursive evaluation of procedural generation patterns called *stereotypes*. The stereotypes are semantically context sensitive, are attached to each sense of English words and keywords and are carried into the representation by the analysis procedure. The representation of the meaning of words, and the versatility of the stereotype format, allow for fine meaning distinctions to appear in the French, and for the construction of French differing radically from the English origin.

AIM-213 cs-385
Ravindra B. Thosar,
Recognition of Continuous Speech: Segmentation and Classification using Signature Table Adaptation,
37 pages, September 1973.

This report explores the possibility of using a set of features for segmentation and recognition of continuous speech. The features are not necessarily *distinctive* or minimal, in the sense that they do not divide the phonemes into mutually exclusive subsets, and can have high redundancy. This concept of feature can thus avoid apriori binding between the phoneme categories to be recognized and the set of features defined in a particular system.

An adaptive technique is used to find the probability of the presence of a feature. Each feature is treated independently of other features. An unknown utterance is thus represented by a feature graph with associated probabilities. It is hoped that such a representation would be valuable for a hypothesize-test paradigm as opposed to a one which operates on a linear symbolic Input.

AIM-214 CS-386
Walter A. Perkins, Thomas O. Binford,
A Corner Finder for Visual Feedback,
59 pages, September 1973.

In visual-feedback work often a model of an object and its approximate location are known and it is only necessary to determine its location and orientation more accurately. The purpose of the program described herein is to provide such Information for the case in which the model is an edge or corner. Given a model of a line or a corner with two or three edges, the program searches a TV window of arbitrary size looking for one or all corners which match the model. A model-driven program directs the search. It calls on another program to find all lines inside the window. Then it looks at these lines and eliminates lines which cannot match any of the model lines. It next calls on a program to form vertices and then checks for a matching vertex. If this simple procedure fails, the model-driver has two backup procedures. First it works with the lines that it has and tries to form a matching vertex (corner). If this fails, it matches parts of the model with vertices and lines that are present and then takes a careful look in a small region in which it expects to find a missing line. The program often finds weak contrast edges in this manner. Lines are found by a global method after the entire window has been scanned with the Hueckel edge operator.

⁎ AIM-215 CS-387 AD 769380
Bruce G. Buchanan, N. S. Sridharan,
Analysis of Behavior of Chemical Molecules: Rule Formation on Non-homogeneous Classes of Objects,
15 pages, September 1973.

An information processing model of some important aspects of Inductive reasoning is presented within the context of one scientific discipline. Given a collection of experimental (mass spectrometry) data from several chemical molecules the computer program described here separates the molecules into *well-behaved* subclasses and selects from the space of all explanatory processes the *characteristic* processes for each subclass. The definitions of *well-behaved* and *characteristic* embody several heuristics which are discussed. Some results of the program are discussed which have been useful to chemists and which lend credibility to this approach.

⁎ AIM-2 16 CS- 389
Larry Masinter, N.S. Sridharan, J. Lederberg, S. H. Smith,
Applications of Artificial Intelligence for Chemical Inference: XII. Exhaustive Generation of Cyclic and Acyclic Isomers,
60 pages, September 1973.

A systematic method of identification of all possible graph isomers consistent with a given empirical formula is described. The method, embodied in a computer program, generates a complete list of isomers. Duplicate structures are avoided prospectively.

⁎ AIM-2 17 cs-39 1 AD770610
N. S. Srldharan,
Search Strategies for the Task of Organic Chemical Synthesis,
32 pages, August 1973.

A computer program has been written that successfully discovers syntheses for complex organic chemical molecules. The definition of the search space and strategies for heuristic search are described in this paper.

AIM-2 18 cs-393
Jean Etienne Vuillemin,
Proof Techniques for Recursive Programs,
Thesis: Ph.D. in Computer Science,
97 pages, October 1973.

The concept of least fixed-point of a continuous function can be considered as the unifying thread of this dissertation. The connections between fixed-points and recursive programs are detailed in Chapter 2, providing some insights on practical implementations of recursion. There are two usual characterizations of the least fixed-point of a continuous function. To the first characterization, due to Knaster and Tarski, corresonds a class of proof techniques for programs, as described in Chapter 3. The other characterization of least fixed points, better known as Kleene's first recursion theorem, is discussed in Chapter IV. It has the advantage of being effective and it leads to a wider class of prrof techniques.

⁎ AIM-219 cs-394 AD769674
C. A. R. Hoare,
Parallel Programming: an Axiomatic Approach,
33 pages, October 1973.

This paper develops some ideas expounded in [1]. It distinguishes a number of ways of using parallelism, including disjoint processes, competition, cooperation, communication and "colluding". In each case an axiomatic proof rule is given. Some light is thrown on traps or ON conditions. Warning: the program structuring methods described here are not suitable for the construction of operating systems.

AIM-220 CS-396
Robert Bolles, Richard Paul,
The use of Sensory Feedback in a Programmable Assembly Systems,
26 pages, October 1973.

This article describes an experimental, automated assembly system which uses sensory feedback to control an electro-mechanical arm and TV camera. Visual, tactile, and force feedback are used to improve positional information, guide manipulations, and perform inspections. The system has two phases: a *planning* phase in which the computer is programmed to assemble some object, and a *working* phase in which the computer controls the arm and TV camera in actually performing the assembly. The working phase is designed to be run on a mini-computer.

The system has been used to assemble a water pump, consisting of a base, gasket, top, and six screws. This example is used to explain how the sensory data is incorporated into the control system. A movie showing the pump assembly is available from the Stanford Artificial Intelligence Laboratory.

AIM-221 cs-447
Luigia Aiello, Mario Aiello, Richard Weyhrauch,
The Semantics of PASCAL in LCF,
78 pages, October 1974.

We define a semantics for the arithmetic part of PASCAL by giving it an interpretation in LCF, a language based on the typed λ-calculus. Programs are represented in terms of their abstract syntax. We show sample proofs, using LCF, of some general properties of PASCAL and the correctness of some particular programs. A program implementing the McCarthy Airline reservation system is proved correct.

AIM-222 *CS-467*
Mario Aiello, Richard Weyhrauch,
Checking Proofs in the Metamathematics of First Order Logic,
55 pages, (forthcoming).

This is a report on some of the first experiments of any size carried out using the new first order proof checker FOL. We present two different first order axiomatizations of the metamathematics of the logic which FOL itself checks and show several proofs using each one. The difference between the axiomatizations is that one defines the metamathematics in a many sorted logic the other does not.

+ AIM-223 cs-400 AD772509
C. A. R. Hoare,
Recursive Data Structures,
32 pages, December 1973.

The power and convenience of a programming language may be enhanced for certain applications by permitting- data structures to be defined by recursion. This paper suggests a pleasing notation by which such structures can be declared and processed; it gives the axioms which specify their properties, and suggests an efficient implementation method, It shows how a recursive data structure may be used to represent another data type, for example, a set. It then discusses two ways in which significant gains in efficiency can be made by selective updating of structures, and gives the relevant proof rules and hints for implementation. It is shown by examples that a certain range of applications can be efficiently programmed, ithout introducing the low-level concept of a reference into a high-level programming language.

⊗ AIM-224 cs-403
C. A. R. Hoare,
Hints on Programming Language Design,
29 pages, December 1973.

This paper (based on a keynote address presented at the *SIGACT/SIGPLAN Symposium on Principles of Programming Languages,* Boston, October 1-3, 1973) presents the view that a programming language is a tool which should assist the programmer in the most difficult aspects of his art, namely program design, documentation, and debugging. It discusses the objective criteria for evaluating a language design, and illustrates them by application to language features of both high level languages and machine code programming. It concludes with an annotated reading list, recommended for all intending language designers.

⊗ AIM-225 CS-406
W. A. Perkins,
Memory Model For a Robot,
118 pages, January 1974.

A memory model for a robot has been designed and tested in a simple toy-block world for which it has shown clarity, efficiency, and generality. In a constrained psuedo-English one can ask the program to manipulate objects and query it about the present, past, and possible future states of its world. The program has a good understanding of its world and gives intelligent answers in reasonably good English. Past and hypothetical states of the world nre handled by changing the state the world in an imaginary contest. Procedures interrogate and modify two globabl databases, one which contains the present representation of the world and another which contains the past history of events, conversations, etc. The program has the ability to create, destroy, and even resurrect objects in its world.

+ AIM-226 cs-407
F.H.G. Wright II, R. E. Corin,
FAIL,
61 pages, April 1974.

This is a reference manual for FAIL, a fast, one-pass assembler for PDP-10 and PDP-6 machine language. FAIL statements, pseudo-operations, macros, and conditional assembly features are described. Although FAIL uses substantially more main memory than MACRO- 10, it assembles typical programs about five times faster. FAIL assembles the entire Stanford time-sharing operating system (two million characters) in less than four mrnutes of CPU time on a KA-10 processor. FAIL permits an ALGOL-style block structure which provides a way of localizing the usage of some symbols to certain parts of the program, such that the same symbol name can be used to mean different things in different blocks.

AIM-227 cs-408
A. J. Thomas, T. 0. Binford,
Information Processing Analysis of Visual Perception: A Review,
? pages, forthcoming.

We suggest that recent advances in the construction of artificial vision systems provide the beginnings of a framework for an information processing analysis of human visual perception. We review some pertinent Investigations which have appeared in the psychological literature, and discuss what we think t be some of the salient and potentially useful theoretical concepts which have resulted from the attempts to build computer vision systems. Finally we try to integrate these two sources of ideas to suggest some desireable structural and behavioural concepts which apply to both the natural and artificial systems.

⊗ AIM-228 cs-409 AD776233
Lester Earnest (ed.),
FINAL REPORT: The First Ten Years of Artificial Intelligence Research at Stanford,
118 pages, July 1973,

The first ten years of research in artificial intelligence and related fields at Stanford University have yielded significant results in computer vision and control of manipulators, speech recognition, heuristic programming, representation theory, mathematical theory of computation, and modeling of organic chemical processes. This report summarizes the accomplishments and provides bibliographies in each research area.

⊗ AIM-229 cs-4 11
D.B. Anderson, T.O. Binford, A.J. Thomas, R.W. Weyhrauch, Y.A. Wilks,
AFTER LEIBNIZ...: Discussions on Philosophy and Artificial Intelligence,
43 pages, April 1974.

This is an edited transcript of informal conversations which we have had over recent months, in which we looked at some of the issues which seem to arise when artificial intelligence and philosophy meet. Our aim was to see what might be some of the fundamental principles of attempts to build intelligent machines. The major topics covered are the relationship of AI and philosophy and what help they might be to each other; the machanisms of natural inference and deduction; the question of what kind of theory of meanrng would be Involved in a successful natural language understanding program, and the nature of models in AI resea rch.

⊗ AIM-230 CS-412
Daniel C. Swinehart,
COPILOT: A Multiple Process Approach to Interactive Programming Systems,
Thesis: Ph.D. in Computer Science,
2 13 pages, August 1974.

The addition of multiple processing facilities to a language used in an interactive computing environment requires new techniques. This dissertation presents one approach, emphasizing the characteristics of the Interface between the user and the system.

We have designed an experimental interactive programming system, COPILOT, as the concrete vehicle for testing and describing our methods. COPILOT allows the user to create, modify, investigate, and control programs written in an Algol-like language, which has been augmented with facilities for multiple processing. Although COPJ LOT is compiler-based, many of our solutions could also be applied to an Interpretive system.

Central to the design is the use of CRT displays to present programs, program data, and system status. This continuous display of information in context allows the user to retain comprehension of complex program environments, and to indicate the environments to be affected by his commands.

COPILOT uses the multiple processing facilities to its advantage to achieve a kind of interactive control which we have termed *non-preemptive.* The user's terminal is continuously available for commands of any kind: program editing, variable inquiry, program control, etc., independent of the execution state of the processes he is controlling. No process may unilaterally gain possession of the user's input; the user retains control at all times.

Commands in COPILOT are expressed as statements in the programming language. This single language policy adds consistency to the system, and permits the user to construct procedures for the execution of repetitive or complex command sequences. An abbreviation facility is provided for the most common terminal operations, for convenience and speed.

We have attempted in this thesis to extend the facilities of interactive programming systems in response to developments in language design and information display technology. The resultant system provides an interface which, we think, is better matched to the interactive needs of its user than are its predecessors.

⊗ AIM-231 *cs-4* 13
James Cips,
Shape Grammars and their Uses,
Thesis: Ph.D. in Computer Science,
243 pages, August 1974.

Shape grammars are defined and their uses are investigated. Shape grammars provide a means for the recursive specification of shapes. A shape grammar is presented that generates a new class of reversible figures. Shape grammars are given for some well known mathematical curves. A simple method for constructing shape grammars that simulate Turing machines is presented. A program has been developed that uses a shape grammar to solve a perceptual task involving the analysis and comparison of line drawings that portray three-dimensional objects of a restricted type. A formalism that uses shape grammas to generate paintings is defined, its Implementation on the computer is described, and examples of generated paintings are shown. The use of shape

⊗ AIM-232 cs-4 14
Bruce G. Baumgart,
GEOMED - A Geometric Editor,
45 pages, May 1974.

GEOM ED is a system for doing 3-D geometric model'ing; used from a keyboard, it is an Interactive drawing program; used as a package of SAIL or LISP accessible subroutines, it is a graphics language. With GEOMED, arbitrary polyhedra can be constructed; moved about and viewed in perspective with hidden lines eliminated. In addition to polyhedra; camera and image models are provided so that simulators relevant to computer vision, problem solving, and animation may be constructed.

⊗ AIM-233 cs-4 I9
Charles J. Rieger, III,
Conceptual Memory: A Theory and Computer Program for Processing the Meaning Content of Natural Language Utterances,
Thesis: Ph.D. in Computer Science,
393 pages, June 1974.

Humans perform vast quantities of spontaneous, subconscious computation in order to understand even the simplest natural language utterances. The computation is principally meaning-based, with syntax and traditional semantics playing insignificant roles. This thesis supports this conjecture by synthesis of a theory and computer program which account for many aspects of language behavior in humans. It is a theory of language and memory.

Since the theory and program deal with language in the domain of conceptual meaning, they are independent of language form and of any specific language. Input to the memory has the form of analyzed conceptual dependency graphs which represent the underlying meaning of language utterances. Output from the memory is also in the form of meaning graphs which have been produced by the active (inferential) memory processes which dissect, transform, extend and recombine the input graphs in ways which are dependent upon the meaning context in which they were perceived.

A memory formalism for the computer model is first developed as a basis for examining the inferential processes by which comprehension occurs. Then, the notion of inference space is presented, and sixteen classes of conceptual inference and their implementation in the computer model are examined, emphasizing the contribution of each class to the total problem of understanding. Among the sixteen inference classes are: causative/resultative inferences (those which explain and predict cause and effect relationships relative to the memory's model of the world), motivational inferences (those which infer the probable intentions of actors), enabling inferences (those which predictively fill out the circumstances which were likely to have obtained at the time of an action), action prediction inferences (those which make guesses about what a person might be expected to do in some situation), knowledge propagation inferences (those which predict what knowledge is available to a person, based on what the memory already knows or can infer he knows), normative Inferences (those which assess the "normality" of a given piece of information), and state duration inferences (those which predict the probable duration of specific states in the world). All inferences are probabilistic, and "backup" is deemphasized as a programming tool.

The idea of points of contact of information structures in inference space is explored. A point of contact occurs when an inferred unit of meaning from one starting point within one utterance's meaning graph either confirms (matches) or contradicts an Inferred unit of meaning from another point within the graph, or from within the graph of another utterance. The quantity and quality of points of contact serve as the primary definition of understanding, since such points provide an effective measure of the memory's ability to relate and fill in information.

Interactions between the Inference processes and (1) *word sense promotion* (how meaning contest influences the language analyzer's choice of lexical senses of words during the parse), and (2) the processes of reference (how memory pointers to tokens of real world entities are established) are examined. In particular, an important inference-reference relaxation cycle is Identified and solved.

The theory forms a basis for a computationally effective and comprehensive theory of language understanding by conceptual inference. Numerous computer examples are included to Illustrate key points. Most issues are approached from both psychological and computational points of view, and the thesis is intended to be comprehensible to people with a limited background in computers and symbolic computation.

AIM-294 CS431
Kenneth Mark Colby, Roger C. Parkison, Bill Faught,
Pattern-Matching Rules for the Recognition of Natural Language Dialogue Expressions,
23 pages, June 1974.

Mat-i-machine dialogues using everyday conversational English present difficult problems for computer processing of natural language. Grammar-based parsers which perform a word-by-word, parts-of-speech analysis are too fragile to operate satisfactorily in real time interviews allowing unrestricted English. In constructing a simulation of paranoid thought processes, we designed an algorithm capable of handling the linguistic expressions used by interviewers in teletyped diagnostic psychiatric interviews. The algorithm uses pattern-matching rules which attempt to characterize the input expressions by progressively transforming them into patterns which match, completely or fuzzily, abstract stored patterns. The power of this approach lies in its ability to ignore recognized and unrecognized words and still grasp the meaning of the message. The methods utilized are general and could serve any "host" system which takes natural language input.

AIM-235 CS-432
Richard W. Weyhrauch, Arthur J. Thomas,
FOL: A Proof Checker for First-order Logic,
57 pages, forthcoming.

This manual describes a machine Implementation of an extended version of the system of natural deduction described by Prawitz. This language, called FOL, extends Prawitz's formulation to a many-sorted logic allowing a partial order over sorts. FOL also allows deductions to be made in some intuitionistic, modal and strict-implication logics. It is intended to be a vehicle for the investigation of the metamathamatics of first-order systems, of problems in the theory of computation and of issues in representation theory.

AIM-236 cs-433
Jack R. Buchanan and David C. Luckham,
On Automating the Construction of Programs,
65 pages, May 1974.

An experimental system for automatically generating certain simple kinds of programs is described. The programs constructed are expressed in a subset of ALGOL containing assignments, function calls, conditional statements, while loops, and non-recursive procedure calls. The input is an environment of primitive programs and programming methods specified in a language currently used to define the semantics of the output programming language. The system has been used to generate programs for symbolic manipulation, robot control, everyday planning, and computing arithmetical functions.

AIM-237 CS-436
Yorick Wilks,
Natural Language Understanding Systems Within the AI Paradigm -- A Survey and Some Comparisons,
26 pages, forthcoming.

The paper surveys the major projects on the understanding of natural language that fall within what may now be called the artificial intelligence paradigm for natural language systems. Some space is devoted to arguing that the paradigm is now a reality and different in significant respects from the generative paradigm of present day linguistics. The comparisons between systems center around questions of the relative perspicuity of procedural and static representations; the advantages and disadvantages of developing systems over a per-rod to test their limits; and the degree of agreement that now exists on what are the sorts of information that must be available to a system that is to understand everyday language.

⊗ AIM-238 cs-437
Christopher K. Riesbeck,
Computational Understanding: Analysis of Sentences and Context,
Thesis: Ph.D. in Computer Science,
245 pages, forthcoming.

The goal of this thesis was to develop a system for the computer analysis of written natural language texts that could also serve a a theory of human comprehension of natural language. Therefore the construction of this system was guided by four basic assumptions about natural language comprehension. First, the primary goal of comprehension is always to find meanings as soon as possible. Other tasks, such as discovering syntactic relationships, are performed only when essential to decisions about meaning. Second, an attempt is made to understand each word as soon as it is read, to decide what it means and how it relates to the rest of the text. Third, comprehension means not only understanding what has been seen but also predicting what is likely to be seen next. Fourth, the words of a text provide the cues for finding the information necessary for comprehending that text.

+ AIM-239 cs-43s
Marsha Jo Hannah,
Computer Matching of Areas in Stereo Images,
Thesis: Ph.D. in Computer Science,
99 pages, July 1974.

This dissertation describes techniques for efficiently matching corresponding areas of a stereo pair of images. Measures of match which are suitable for this purpose are discussed, as are methods for pruning the search for a match. The mathematics necessary to convert a set of matchings into a workable camera model are given, along with calculations which use this mode1 and a pair of image points to locate the corresponding scene point. Methods are included to detect some types of unmatchable target areas in the original data and for detecting when a supposed match is invalid. Region growing techniques are discussed for extend matching areas into regions of constant parallax and for delimiting uniform regions in an image. Also, two algorithms are presented to show some of the ways in which these techniques can be combined to perform useful tasks in the processing of stereo images.

+ AIM-240 *cs-444*
C. Cordell Green, Richard J. Watdinger, David R. Barstow, Robert Elschlager, Douglas B. Lenat, Brian P. McCune, David E. Shaw, and Louis I. Steinberg,
Progress Report on Program-understanding Systems,
47 pages, August 1974.

This progress report covers the first year and one- half of work by our automatic programmtng research group at the Stanford Artificial Intelligence Laboratory. Majoi emphasis has been placed on methods of program specification, codification of programming knowledge, and implementation of pilot systems for program writing and understancltng. List processing has been used as the general problem domain for this work.

+ AIM-241 CS-446
Luigia Aiello, Richard W. Weyhrauch,
LCFsmall: an implementation of LCF,
45 pages, August 19'74.

This is a report on a computer program implementing a simplified verston of LCF. It is written (wtth mtnor exceptions) entirely in pure LISP and has none of the user oriented features of the implementation described by Milner. We attempt to represent directly in code the metamathematical notions necessary to describe LCF. We hope that the code is simple enough and the metamathematics is cleai enough so that properties of this particular program (e.g. its correctness) can eventually be proved. The program is reproduced in full.

⊛ AIM-242 cs-4 52
James R. Low,
Automatic Coding: Choice of Data **Structures**,
Thesis: Ph.D. in Computer Science,
1 10 pages, August 1974.

A system is described which automatically c hooscs representations foi high-level information structures, such as sets, sequences, and relations for a given computer program. Representations are picked from a fixed library of low-level data structures including linked-lists, binary trees and hash tables. The representations are chosen by attempting to minimize the predicted space-time integral of the user's program execution. Predictions are based upon statistics of information structure use provided directly by the user and collected by monitoring executions of the user program using default representations for the high-level structures. A demonstration system has been constructed. Results using that system are presented.