THE FIRST TEN YEARS OF ARTIFICIAL INTELLIGENCE RESEARCH AT STANFORD

STANFORD UNIVERSITY

PREPARED FOR
DEPARTMENT OF DEFENSE
ADVANCED RESEARCH PROJECTS AGENCY

JULY 1973

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD 776 233

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER STAN-CS-74-409	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FINAL REPORT: THE FIRST TEN YEARS OF ARTIFICIAL INTELLIGENCE RESEARCH AT STANFORD		5. TYPE OF REPORT & PERIOD COVERED technical, 1973 6. PERFORMING ORG. REPORT NUMBER STAN-CS-74-409
7. AUTHOR(s) ed., Lester Earnest		8. CONTRACT OR GRANT NUMBER(s) SD-183
9. PERFORMING ORGANIZATION NAME AND ADDRESS Stanford University Computer Science Department Stanford, California 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order No. 457
11. CONTROLLING OFFICE NAME AND ADDRESS ARPA/IPT, Attn: Stephen D. Crocker, 1400 Wilson Blvd., Arlington, Va. 22209		12. REPORT DATE July 1973 13. NUMBER OF PAGES 120/double sided
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) ONR Representative: Jack Ducey Durand Aeronautics Bldg., Rm. 165 Stanford University Stanford, California 94305		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) releasable without limitations on dissemination.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

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 modelling of organic chemical processes. This report summarizes the accomplishments and provides bibliographies in each research area.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Stanford Artificial Intelligence Laboratory
Memo AIM-228

July 1973

Computer Science Department
Report No. STAN-CS-74-409

FINAL REPORT

The First Ten Years of Artificial Intelligence Research at Stanford

Edited by
Lester Earnest

ARTIFICIAL INTELLIGENCE PROJECT
John McCarthy, Principal Investigator

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

ABSTRACT

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.

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract SD-183. 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.

I. INTRODUCTION

Artificial Intelligence is the experimental and theoretical study of perceptual and intellectual processes using computers. Its ultimate goal is to understand these processes well enough to make a computer perceive, understand and act in ways now only possible for humans.

In the late 1950s John McCarthy and Marvin Minsky organized the Artificial Intelligence Project at M.I.T. That activity and another at Carnegie Tech (now Carnegie-Mellon University) did much of the pioneering research in artificial intelligence.

In 1962, McCarthy came to Stanford University and initiated another A. I. Project here. He obtained financial support for a small activity (6 persons) from the Advanced Research Projects Agency (ARPA) beginning June 15, 1963.

A Computer Science Department was formed at Stanford in January 1965. By that time there were 15 people on the Project, including Edward Feigenbaum who had just arrived. Shortly, a decision was made to expand the activities of the Project, especially in the area of hand-eye research. Additional support was obtained from ARPA and a PDP-6 computer system was ordered. Lester Earnest arrived in late 1965 to handle administrative responsibilities of the expanding Project.

By the summer of 1966, the Project had outgrown available campus space and moved to the D. C. Power Laboratory in the foothills above Stanford. The new computer system was delivered there. Arthur Samuel and Jerome Feldman arrived at about this time and D. Raj. Reddy joined the faculty, having completed his doctorate on speech recognition as a member of the Project. Several faculty members from other departments affiliated themselves with the Project, but without direct financial support: Joshua Lederberg (Genetics), John Chowning and Leland Smith (Music), and Anthony Hearn (Physics).

By early 1968, there were just over 100 people on the Project, about half supported by ARPA. Kenneth Colby and his group joined the Project that year, with separate funding from the National Institute of Mental Health. Other activities subsequently received some support from the National Science Foundation and the National Aeronautics and Space Administration. Zohar Manna joined the Faculty and the Project, continuing his work in mathematical theory of computation.

In June 1973, the Artificial Intelligence Laboratory (as it is now called) had 128 members, with about two-thirds at least partially ARPA-supported. Other Computer Science Faculty members who have received such support include Robert Floyd, Cordell Green, and Donald Knuth.

The Heuristic Dendral Project (later changed to Heuristic Programming) was formed in 1965 under the leadership of Edward Feigenbaum and Joshua Lederberg. It was initially an element of the A. I. Project and consisted of five or so people for several years.

The Heuristic Dendral Project became a separate organizational entity with its own ARPA budget in January 1970 and also obtained some support from the Department of Health, Education, and Welfare. It has had 15 to 20 members in recent months.

The following sections summarize accomplishments of the first 10 years (1963-1973). Appendices list external publications, theses, film reports, and abstracts of research reports produced by our staff.

2. ARTIFICIAL INTELLIGENCE PROJECT

The work of the Stanford Artificial Intelligence Project has been basic and applied research in artificial intelligence and closely related fields, including computer vision, speech recognition, mathematical theory of computation, and control of an artificial arm.

Expenditures from ARPA funds over the ten years beginning June 15, 1963 have amounted to $9.2 million. About 43% of this was for personnel (salaries, wages, benefits), 26% for computer and peripheral equipment (purchase, replacement parts, and rental), 8% for other operating expenses, and 23% for indirect costs.

Here is a short list of what we consider to have been our main accomplishments. More complete discussions and bibliographies follow.

Robotics

Development of vision programs for finding, identifying and describing various kinds of objects in three dimensional scenes. The scenes include objects with flat faces and also curved objects.

The development of programs for manipulation and assembly of objects from parts. The latest result is the complete assembly of an automobile water pump.

Speech Recognition

Development of a system for recognition of continuous speech, later transferred to Carnegie-Mellon University and now being expanded upon elsewhere.

Heuristic Programming

Our support of Hearn's work on symbolic computation led to the development of REDUCE, now being extended at the University of Utah and widely used elsewhere.

Work in heuristic programming resulting in Luckham's resolution theorem prover. This is currently about the best theorem prover in existence, and it puts us in a position to test the limitations of current ideas about heuristics so we can go beyond them.

Representation Theory

Work in the theory of how to represent information in a computer is fundamental for heuristic programming, for language understanding by computers and for programs that can learn from experience. Stanford has been the leader in this field.

Mathematical Theory of Computation

Our work in mathematical theory of computation is aimed at replacing debugging by computer checking of proofs that programs meet their specifications. McCarthy, Milner, Manna, Floyd, Igarashi, and Luckham have been among the leaders in developing the relevant mathematical theory, and the laboratory has developed the first actual proof-checking programs that can check proofs that actual programs meet their specifications. In particular, Robin Milner's LCF is a practical proof checker for a revised version of Dana Scott's logic of computable functions.

Research Facilities

We have developed a laboratory with very good computer and program facilities and special instrumentation for the above areas, including a timesharing system with 64 online display terminals.

We developed a mechanical arm well suited to manipulation research. It is being copied and used by other laboratories.

We designed an efficient display keyboard that has been adopted at several ARPAnet facilities. We invented a video switch for display systems that is being widely copied.

In the course of developing our facilities, we have improved LISP, developed an extended Algol compiler called SAIL, and created a document compiler called PUB (used to produce this report).

Our early Teletype-oriented text editor called SOS has become an industry standard and our new display editor "E" is much better.

We have written utility programs for the PDP-10 and made numerous improvements to time-sharing systems. Many of our programs, particularly LISP, SAIL, and SOS, are used in dozens of other computer centers.

We designed an advanced central processor that is about 10 times as fast as our PDP-10. In support of this work, we developed interactive design programs for digital logic that have since been adopted by other research and industrial organizations.

Training

In the 1963-1973 period, 27 members of our staff published Ph.D. theses as Artificial Intelligence Memos and a number of other graduate students received direct or indirect support from the A. I. Project.

The following subsections review principal activities, with references to published articles and reports.

2.1 Robotics

The project has produced several substantial accomplishments: an automatic manipulation system capable of assembling a water pump, a laser ranging apparatus, and programs which form symbolic descriptions of complex objects. We have now focused a major effort on robotics in industrial automation.

2.1.1 Manipulation

In 1966, we acquired and interfaced a prosthetic arm from Rancho Los Amigos Hospital. Although it had major mechanical shortcomings, the software experience was valuable. A program was written for point to point control of arm movements. Computer servoing was used from the beginning and has proven much more versatile than conventional analog servoing. A simple system that visually located blocks scattered on a table and sorted them into stacks according to size was operational by the spring of 1967 [Pingle 1968].

In order to move through crowded workspaces, a program was written to avoid obstacles while carrying out arm movements [Pieper 1968]. That program was fairly general but rather slow, since it used a local and not very smart search technique to find its way around obstacles.

A hydraulic arm was designed and built according to stringent criteria of speed and strength. Kahn developed a minimum-time servo scheme, which is close to a bang-bang servo [Kahn 1971]. The effect was impressive (even frightening) but hard on the equipment.

The next arm was designed with software in mind [Scheinman 1969]. It was completed in December 1970, and has proved a good research manipulator; several other groups have made copies.

Arm control software for the new arm was split into a small arm-control servo program, which ran in real time mode on our PDP-6 computer, and a trajectory planning program [Paul 197i], written in a higher level language and running on a timeshared PDP-10.

The arm servo software contained several new features: a) feedforward from a Newtonian dynamic model of the arm, including gravity and inertial forces; b) feedback as a critically damped harmonic oscillator including velocity information from tachometers; c) trajectory modification facility in the servo program, which allows considerable change from planned trajectories to accomodate contingency conditions. Compare this control mechanism with the usual analog servo; the kinematic model and computer servo allow changes of several orders of magnitude in the equivalent servo constants, to account for variations in gravity loads and inertial effects.

The arm was designed so that solution for joint angles from position and orientation is simple and rapid. The techniques apply to a wide range of arms; thus our work has significance for industrial and other robotics applications.

The MOVE_INSTANCE capability [Paul 1971] shows a simple form of automating some manipulation procedures. The routine chooses a best grasping and departure for a class of known objects. Models of these objects are stored in order to choose all possible grasping positions -- parallel faces, a face and a parallel edge, etc. The full range of possible departure and approach angles is investigated and a solution chosen, if possible, which allows manipulation in a single motion. Otherwise, a solution is chosen which uses an intermediate position to regrasp the object. Thus, this facility provides a method for grasping general polyhedra in arbitrary position and orientation, provided we have a model of the object.

Arm planning is based on making complete motions, e.g. picking up an object and putting it down. If we pick up an object arbitrarily, without thought for putting it down later, we may not be able to put it down as desired and may need to grasp it again. Complete trajectories are pieced together from various segments, e.g. grasp, departure, mid-segment, approach, release. Trajectories are specified by the user in an interpretive hand language (HAL) in terms of macros at the level of inserting a screw. Endpoint positions and orientations are often specified by positioning the arm itself in the proper position and recording joint angles, a form of "learning by doing". The language provides acilities for control using touch and force sensing.

Pump Assembly

Our first major task was assembly of an automobile water pump. A film which shows the process in detail is available for distribution [Pingle and Paul, "Automated Pump Assembly"]. We describe the task in some detail to show the level of programming.

The pump parts include pump base, cover, gasket and screws. We chose a plausible industrial environment with tools in fixed places, screws in a feeder, and pump body and cover on a pallet. It is located by vision, then moved by the arm to a standard position, up against some stops. Pins are inserted into screw holes in the pump body to guide alignment of gasket and cover. If necessary the hand searches to seat the pins.

The gasket is placed over the guide pins and visually inspected for correct placement. The cover is expected to be within about a quarter of an inch of its fixed place on the pallet. After locating the cover by touch, the

hand places the cover over the guide pins. The hand then picks up a hex head power screwdriver, picks up a screw from the feeder and inserts the screw and, if necessary, searches to seat the screw. A second screw is inserted. The hand then removes the two pins and inserts four more screws, completing the assembly. Finally, it tests that the pump impeller turns freely.

Three forms of feedback are used, visual, tactile, and force. The visual feedback is provided by strictly special purpose programs which have no general interest. We plan to generalize visual feedback capabilities and include them in a system like that used to program the rest of the task.

The manipulation parts of the assembly were programmed in the hand language, HAL. The actual program follows:

```
BEGIN PUMP
ALIGN           ;align pump base at stops
PIN P1 H1       ;put pin P1 at hole H1
PIN P2 H2       ;put pin P2 at hole H2
GASKET
TOP
SCREW1          ;put in first 2 screws
UNPIN H1 P1 H1A ;remove pin P1 from hole H1
UNPIN H2 P2 H2A ;remove pin P2 from hole H2
SCREW2          ;insert last 4 screws
TEST
END
```

Each of the commands is a macro. The task was performed in a general purpose hand language with a control program which could be readily adapted to other manipulators. Thus the system is more than a special purpose demonstration.

2.1.2 Vision

During the past two years our vision effort has shifted from scenes of blocks to outdoor scenes and scenes of complex objects. In both cases, interpretation has made use of world models.

A crucial part of our work with complex objects is our development of suitable representation of shape. This is an area closely connected with industrial applications of research, since representation of shape is important in programming of robots, design, display, visual feedback, assembly, and symbolic description for recognition.

Wichman assembled the first robotics visual feedback system [Wichman 1967]. It was deficient in several ways: only the outer edges of the blocks were observed, the hand had to be removed from view when visual checking was done, and the technique was limited in its domain of objects.

Gill then improved these aspects by recognizing hand marks and internal edges of blocks [Gill 1972]. The system was model driven in that specifying the projected appearance of a corner in a small window programmed the system for a new task. But the system was limited to tasks with polyhedra, e.g. stacking blocks and inserting blocks into holes.

A class of representations of shape has been applied to symbolic description of shapes of toys and tools [Agin 1972, Nevatia 1973]. The representation depends on a part/whole description of objects in terms of parts which may themselves be composed of parts. The success of any part/whole representation depends on the utility of the primitive parts; within the representation, dominant primitives are "generalized cones", originating from a formalization called "generalized translational invariance". These primitives are locally defined by an arbitrary

cross section and a space curve called the axis, along which the cross sections are translated normal to the axis.

We have developed laser depth ranging hardware which has been operative since January 1971. The device is very simple and can be replicated now for less than $1000, assuming that a suitably sensitive camera tube such as a silicon vidicon or solid state sensor is available. From that experimental data, we have obtained complete descriptions of a doll, a toy horse, a glove, etc., in terms of part/whole descriptions in the representation just described. The same techniques could be used for monocular images, with considerable benefit, in spite of the added difficulty of using only monocular information. We are now writing programs which match these graph descriptions for recognition.

The work on visual feedback depends on display techniques which have been developed here and elsewhere. An interactive program GEOMED [Baumgart 1972a] was developed to allow description of objects by building them up from a set of geometric primitives. The usually tedious task of input of descriptions of complex objects is much simplified. For our purposes, a symbolic line drawing output is necessary; a fast hidden line elimination program with symbolic line output structure has been written.

Working with outdoor scenes introduces new difficulties: surfaces are textured, line finders etc., have been specialized to scenes of polyhedra. We have made substantial progress by incorporating new visual modules such as color region finders with a world model. The first of these efforts [Bajcsy 1972] included a color region finder and Fourier descriptors of texture. Textures were described by directionality, contrast, element size and spacing. These interpretations from the Fourier transform are useful but not always valid, and a discussion of the limitations of Fourier descriptors was included. Depth cues were obtained from the gradient of texture. The results of color and texture region growing were shown, and a simulation made of combining these various modes with a world model. An interesting conclusion was that three dimensional interpretations and cues were the most semantic help in scene interpretation.

A second project has dealt with outdoor scenes [Yakimovsky 1973]. Yakimovsky made a region-based system which sequentially merges regions based on semantics of familiar scenes, using two-dimensional image properties. The system has an initial stage which is very much like other region approaches, merging regions based on similarity of color and other crude descriptors, except that it eliminates weakest boundaries first. The second stage introduces a world model and a means of estimating the best interpretation of the scene in terms of that model. The semantic evaluation provides much better regions than the same system without the semantics. It has been demonstrated on road scenes, for which rather good segmentations and labellings have been achieved. An application was also made to heart angiograms.

In human and animal peception, stereo and motion perception of depth are important. Several of these mechanisms have been programmed. Nevatia [1973] evaluated motion parallax of a moving observer for scenes of rocks. That program tracked by correlation subsequent images in a series of images with small angle between images. The technique allows large baselines for triangulation. Another approach to stereo has been carried out on outdoor scenes [Hannah unpublished].

An edge operator [Hueckel 1971] gives a good indication, based on local brightness, of

whether or not there is an edge near the center of a neighborhood. If there is an edge, its position and location are relatively well determined.

Differencing techniques have been used to find changes in scenes [Quam 1972]. Color region growing techniques have been used. Accomodation of camera parameters to optimize image quality for each task has been extensively used.

A focussing module allows automatic focussing. Color identification modules have been written, the latest incorporates the Retinex model of Land to achieve a certain color constancy, independent of the variation of illuminance spectrum from scene to scene (sunlight, incandescent, xenon arc, flourescent) or within a scene (reflections from colored objects). Calibration modules allow maintaining a system in a well-calibrated state to maintain reliability, and increase reliability by self-calibration.

Programs have been written to understand scenes of blocks given line drawings. An early version recognized only outlines of isolated objects. A later program, [Falk 1972], arrived at segmentations into objects by techniques which were extensions of Guzman's, but using lines rather than regions. This dealt more effectively with missing and extraneous edges. The system used very limited prototypes of objects, using size in an important and restrictive way to identify objects and their spatial relations. Missing edges were hypothesized for a line verifier program to confirm or reject.

Still another model-based program [Grape 1973] used models of a parallelipiped and a wedge to perform the segmentation into objects corresponding to models. It is able to handle a variety of missing and extraneous edges.

The thrust of all these efforts is the use of models for perception.

Bibliography

[Agin 1972] G. Agin, **Description and Representation of Curved Objects**, PhD Thesis in Computer Science, Stanford A. I. Memo AIM-173, October 1972.

[Bajcsy 1972] R. Bajcsy, **Computer Identification of Visual Texture**, PhD Thesis in Computer Science, Stanford A. I. Memo AIM-180, October 1972.

[Baumgart 1972a] B. Baumgart, **GEOMED - A Geometric Editor**, May 1972. Stanford A. I. Lab., SAILON-68, 1972.

[Baumgart 1972b] Baumgart, Bruce G., **Winged Edge Polyhedron Representation**, Stanford A. I. Memo AIM-179, October 1972.

[Binford 1973a] T. Binford, **Sensor Systems for Manipulation**, E. Heer (Ed), *Remotely Manned Systems*, Calif. Inst. Tech., Pasadena, 1973.

[Binford 1973b] T. O. Binford and Jay M. Tenenbaum, **Computer Vision**, Computer (IEEE), May 1973.

[Earnest 1967] L. D. Earnest, **Choosing an Eye for a Computer**, Stanford A. I. Memo AIM-51, April 1967.

[Falk 1971] G. Falk, **Scene Analysis Based on Imperfect Edge Data**, *Proc. IJCAI*, Brit. Comp. Soc., London, Sept. 1971.

[Falk 1972] G. Falk, **Interpretation of Imperfect Line Data as a Three-Dimensional Scene**, *J. Artificial Intelligence*, Vol 3, No. 2, 1972.

[Feldman 1969] J. Feldman, et al, **The Stanford Hand-Eye Project**, *Proc. IJCAI*, Washington D.C., 1969.

[Feldman 1970] J.A.Feldman, **Getting a**

Computer to see Simple Scenes, *IEEE Student Journal*, Sept. 1970.

[Feldman 1971a] J. Feldman and R. Sproull, System Support for the Stanford Hand-Eye System, Proc. IJCAI, British Computer Soc., London, Sept. 1971.

[Feldman 1971b] J. Feldman, et al, The Use of Vision and Manipulation to Solve the Instant Insanity Puzzle, *Proc. IJCAI*, Brit. Comp. Soc., London, Sept. 1971.

[Feldman 1972] J. Feldman, et al, Recent Developments in SAIL -- An Algol-Based Language for Artificial Intelligence, *Proc FJCC*, 1972.

[Gill 1972] Aharon Gill, Visual Feedback and Related Problems in Computer-controlled Hand-eye Coordination, PhD Thesis in EE, Stanford A. I. Memo AIM-178, October 1972.

[Grape 1973] Gunnar R. Grape Model-Based (Intermediate-Level) Computer Vision, PhD thesis in Comp. Sci., Stanford A. I. Memo AIM-201, May 1973.

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

[Kahn 1971] M. Kahn and B. Roth, The Near-Minimum-time Control of Open-Loop Articulated Kinematic Chains, *Trans. ASME*, Sept 1971.

[Kelly 1970] Michael D. Kelly, Visual Identification of People by Computer, PhD thesis in Comp. Sci., Stanford A. I. Memo AIM-130, July 1970.

[McCarthy 1970] J. McCarthy, Computer Control of a Hand and Eye, *Proc. 3rd All-Union Conference on Automatic Control (Technical Cybernetics)*, Nauka, Moscow, 1967 (Russian).

[Montanari 1969] U. Montanari, Continuous Skeletons from Digitized Images, *J. ACM*, October 1969.

[Montanari 1970a] U. Montanari, A Note on Minimal Length Polygonal Approximation to a Digitized Contour, *Comm. ACM*, January 1970.

[Montanari 1970b] U. Montanari, On Limit Properties in Digitization Schemes, *J. ACM*, April 1970.

[Montanari 1970c] U.Montanari, Separable Graphs, Planar Graphs and Web Grammars, *Information and Control*, May 1970.

[Montanari 1970d] U. Montanari, Heuristically Guided Search and Chromosome Matching, *Artificial Intelligence*, Vol 1, No. 4, December 1970.

[Montanari 1971] U. Montanari, On the Optimal Detection of Curves In Noisy Pictures, *Comm. ACM*, May 1971.

[Nevatia 1973] R.K.Nevatia and T.O.Binford, Structured Descriptions of Complex Objects, *Proc. IJCAI*, Stanford University, August 1973.

[Paul 1969] R.Paul, G.Falk, J.Feldman, The Computer Representation of Simply Described Scenes, *Proc 2nd Illinois Graphics Conf.*, Univ. Illinois, April 1969.

[Paul 1971] R. Paul, Trajectory Control of a Computer Arm, *Proc. IJCAI*, Brit. Comp. Sci., London, Sept. 1971.

[Paul 1972] R. Paul, Modelling, Trajectory Calculation and Servoing of a Computer Controlled Arm, PhD thesis in Comp. Sci., Stanford A. I. Memo AIM-177, Sept. 1972.

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

[Pingle 1968] K. Pingle, J. Singer, and W Wichman, **Computer Control of a Mechanical Arm through Visual Input**, *Proc. IFIP Congress 1968.*

[Pingle 1970] K. Pingle, **Visual Perception by a Computer**, in *Automatic Interpretation and Classification of Images*, Academic Press, New York, 1970.

[Pingle 1972] K. K. Pingle and J. M. Tenenbaum, **An Accomodating Edge Follower**, *Proc. IJCAI*, Brit. Comp. Soc., London, 1971.

[Quam 1972] L. H. Quam, et al, **Computer Interactive Picture Processing**, Stanford A. I. Memo AIM-166, April 1972.

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

[Schmidt 1971] R. A. Schmidt, **A study of the Real-Time Control of a Computer-Driven Vehicle**, PhD. thesis in EE, Stanford A. I. Memo AIM-149, August 1971.

[Sobel 1970] Irwin Sobel, **Camera Models and Machine Perception**, Stanford A. I. Memo AIM-121, May, 1970.

[Tenenbaum 1970] J. M. Tenenbaum, **Accommodation in Computer Vision**, Stanford A. I. Memo AIM-134, September 1970.

[Tenenbaum 1971] J. M. Tenenbaum, et al, **A Laboratory for Hand-Eye Research**, *Proc IFIP Congress 1971.*

[Wichman 1967] W. Wichman, **Use of optical Feedback in the Computer Control of an Arm**, Stanford A. I. Memo AIM-56, August 1967.

[Yakimovsky 1972] Y. Yakimovsky and J. A. Feldman, **A Semantics-Based Decision Theoretic Region Analyzer** *Proc. IJCAI*, Stanford U., August 1973.

FILMS

Gary Feldman, **Butterfinger**, 16mm color with sound. 8 min., March 1968.

Gary Feldman and Donald Pieper, **Avoid**, 16mm silent, color, 5 minutes, March 1969.

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

Suzanne Kandra, **Motion and Vision**, 16mm color, sound, 22 min., November 1972.

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

2.2 Theoretical Studies

Members of our project have pioneered in mathematical theory of computation, representation theory, and grammatical inference. This work is not a proper subcategory of artificial intelligence in that it deals with problems that are basic to all of computer science.

In addition to ARPA sponsorship of this work, the mathematical theory of computation activities received some support from the National Aeronautics and Space Administration and the grammatical inference group has received a grant from the National Science Foundation.

2.2.1 Mathematical Theory of Computation

The idea that computer scientists should study computations themselves rather then just the notion of computability (i.e. recursion theory) was suggested in 1963 by McCarthy [1, 2]. These early papers suggested that mathematical methods could be used to prove (or disprove) the following properties of programs:

1. a program is correct,
2. a program terminates,
3. two programs are equivalent,
4. a translation procedure between two languages is correct, (i.e. it preserves the meaning of a program),
5. optimized programs are equivalent to the original,
6. one program uses less resources than another and is therefore more efficient.

These are simply technical descriptions of a programer's day to day problems. The notion of correctness of a program is just -- "How do we know that a particular program solves the problem that it was intended to?" The usual way of putting it is: "Does my program have bugs in it". A correct mathematical description of what this means is a central problem in MTC and is a genuine first step in any attempt to mechanize the debugging of programs. The equivalence of programs is similar in that until there are clear ways of describing what a program does, saying that they "do" the same thing is impossible. These technical problems are now well enough understood so that serious attempts to apply the results to "real" programs are beginning.

Attempts to formalize these questions have proceeded along several lines simultaneously. In [4, 6] McCarthy and Mansfield discussed new languages for expressing these notions were considered. [4] considered a first order logic which contained an "undefined" truth-value. This was one way of explaining what was meant by computations which didn't terminate. [5] used a traditional first order logic to describe a subset of ALGOL.

In [3] McCarthy proposed that computers themselves might be used to check the correctness of proofs in formal systems, and was the first to actually construct a program to carry this out. This suggests that one could check or possibly look for solutions to the above problems (in the form of proofs in some formal system). As a result a series of proof checkers has been built. The first is reported in [7].

In 1966 Floyd [8] published his now well known method of assigning assertions to the paths in a flowchart, in order to find verification conditions the truth of which guarantee the "correctness" of the original program.

McCarthy, Painter and Kaplan [9, 10, 11, 12, 13, 14] used the ideas in [4, 8] to prove:

1) the correctness of a compiler for arithmetic expressions,
2) the correctness of several compilers for algol-like programs,
3) the equivalence of some algorithms.

Kaplan also gave some completeness results for a formal system which talks about assignment statements [10], and discussed the equivalence of programs [13, 14]. During this time another proof checker was written by W. Weiher [15].

In a series of articles Z. Manna extended and expanded Floyd's original ideas. With A. Pneuli [16, 17] he discussed the relationship between the termination, correctness and equivalence of recursively defined functions and the satisfiability (or unsatisfiability) of certain first order formulas. In [17] they work out an example using the 91 function. In [18] Manna extended his ideas to non-deterministic programs. E. Ashcroft and he did a similar thing for parallel programs in [21].

P. Hayes [18] again attacked the problem of a three-valued predicate logic, this time with a machine implementation in mind. This coincided with a paper of Manna and McCarthy [19], which used this logic.

About this time (1969) several important developments occurred which allowed the above questions to be reexamined from different points of view.

1. In [22] Z. Manna showed how to formulate the notion of partial correctness in second logic.
2. C. A. R. Hoare [24] published a paper describing a new formalism for expressing the meanings of programs in terms of input/output relations.
3. S. Igarashi [23] gave an axiomatic description of an ALGOL-like language.
4. D. Scott suggested using the typed lambda calculus for studying MTC and first described iN 1970 a mathematical model of Church's lambda calculus.

These together with McCarthy's axiomatic approach now represent the most important directions in MTC research. They express different points of view towards the meanings (or semantics) of programs.

Manna (following Floyd) describes the effects of a program by showing what kinds of relations must hold among the values of the program variables at different points in the execution of the program. In particular between the input and the output. In [31] Floyd suggests an interactive system for designing correct programs. These ideas are systematized and expanded by Manna in [34]. He and Ashcroft show how to remove GOTO statements from programs and replace them by WHILE statements in [33].

Hoare shows how properties (including the meaning) of a program can be expressed as rules of inference in his formal system and how these rules can be used to generate the relations described by Floyd and Manna. This puts their approach in a formal setting suitable for treatment on a computer. Work on this formal system is at present being aggressively pursued. Igarashi, London, and Luckham [39] have increased the scope of the original rules and have programed a system called VCG (for verification condition generator) which takes PASCAL programs together with assertions assigned to loops in the program and uses the Hoare rules to automatically generate verification conditions the proof of which guarantee the correctness of the original program. These sentences are then given to a resolution theorem prover [26] which tries to prove them.

There is also a project started by Suzuki under the direction of Luckham to develop programs to take account. of particular properties of arithmetic and arrays when trying to prove the verification conditions. London also produced an informal proof of two Lisp compilers [35].

Igarashi's formal system [23] differs from Hoare's in that the rules of inference act directly on the programs themselves rather than properties of such programs.

Scott's work assumes that the most suitable

meaning for a program is the function which it computes and essentially ignores how that computation proceeds. The other approaches are more intentional in that:

1) they may not necessarily mention that function explicitly although it might appear implicitly.
2) they can (and do) consider notions of meaning that are stronger than Scott's.

For example programs might have to have "similar" computation sequences before considering them equivalent [25].

A computer program LCF (for "logic for computable functions") has been implemented by Milner [26]. This logic uses the typed lambda calculus to defines the semantics of programs. Exactly how to do this was worked out by Weyhrauch and Milner [28, 29, 30]. In conjunction Newey worked on the axiomatization of arithmetic, finite sets, and lists in the LCF environment. This work is still continuing. In addition Milner and Weyhrauch worked with Scott on an axiomatization of the type free lambda calculus. Much of this work was informally summarized in [32].

McCarthy attempts to give an axiomatic treatment to a programming language by describing its abstract syntax in first order logic and stating properties of the programming language directly as axioms. This approach has prompted Weyhrauch to begin the design of a new first order logic proof checker based on natural deduction. This proof checker is expected to incorporate the more interesting features of LCF and will draw heavily on the knowledge gained from using LCF to attempt to make the new first order proof checker a viable tool for use in proving properties of programs.

This work is all being brought together by projects that are still to a large extent unfinished. They include

1. a new version of LCF including a facility to search for proofs automatically;
2. the description of the language PASCAL in terms of both LCF and in first order logic (in the style of McCarthy) in order to have a realistic comparison between these approaches and that of Floyd, Hoare, et al;
3. a continuation of Newey's work;
4. the discussion of LISP semantics in LCF and an attempt to prove the correctness of the London compilers in a formal way (this is also being done by Newey);
5. the design of both special purpose and domain independent proving procedures specifically with program correctness in mind;
6. the design of languages for describing such proof procedures;
7. the embedding of these ideas in the new first order checker.

In addition to the work described above, Ashcroft, Manna, and Pneuli [36], and Chandra and Manna [37] have published results related to program schemas. Manna's forthcoming book [37] will be the first general reference in this field.

Some of these references appeared first as A. I. memos and were later published in journals. In such cases both references appear in the bibliography.

Bibliography

[1] McCarthy, John, **A Basis for a Mathematical Theory of Computation**, in Biaffort, P., and Herschberg, D., (eds.), *Computer Programming and Formal Systems*, North-Holland, Amsterdam, 1963.

[2] McCarthy, John, **Towards a Mathematical Theory of Computation**, *Proc. IFIP Congress 62*, North-Holland, Amsterdam, 1963.

[3] McCarthy, John, **Checking Mathematical Proofs by Computer**, in

Proc. Symp. on Recursive Function Theory (1961), American Mathematical Society, 1962.

[4] McCarthy, John, **Predicate Calculus with 'UNDEFINED' as a Truth-value**, Stanford A. I. Memo AIM-1, March 1963.

[5] McCarthy, John, **A Formal Description of a Subset of Algol**, Stanford A. I. Memo AIM-24, September 1964; also in Steele, T., (ed.), *Formal Language Description Languages*, North Holland, Amsterdam, 1966.

[6] Mansfield, R., **A Formal System of Computation**, Stanford A. I. Memo AIM-25, September 1964.

[7] McCarthy, John, **A Proof-checker for Predicate Calculus**, Stanford A. I. Memo AIM-27, March 1965.

[8] Floyd, R. W., **Assigning Meanings to programs**, Vol. 19, American Mathematical Society, 19-32, 1967.

[9] McCarthy, John, and Painter, J., **Correctness of a Compiler for Arithmetic Expressions**, Stanford A. I. Memo AIM-40, April 1966; also in *Mathematical Aspects of Computer Science*, Proc. Symposia in Applied Mathematics, Amer. Math. Soc., New York, 1967.

[10] Painter, J., **Semantic Correctness of a Compiler for an Algol-like Language**, Stanford A. I. Memo AIM-44, March 1967.

[11] Kaplan, D., **Some Completeness Results in the Mathematical Theory of Computation**, Stanford A. I. Memo AIM-45, October 1966; also in *J. ACM*, January 1968.

[12] Kaplan, D., **Correctness of a Compiler for Algol-like programs**, Stanford A. I. Memo AIM-48, July 1967.

[13] Kaplan, D., **A Formal Theory Concerning the Equivalence of Algorithms**, Stanford A. I. Memo AIM-59, May 1968.

[14] Kaplan, D., **The Formal Theoretic Analysis of Strong Equivalence for Elemental Programs**, Stanford A. I. Memo AIM-60, June 1968.

[15] Weiher, William, **The PDP-6 Proof Checker**, Stanford A. I. Memo AIM-53, June 1967.

[16] Manna, Zohar, and Pnueli, Amir, **The Validity Problem of the 91-function**, Stanford A. I. Memo AIM-68, August 1968.

[17] Manna, Zohar, and Pneuli, Amir, **Formalization of Properties of Recursively Defined Functions**, Stanford A. I. Memo AIM-82, March 1969; also in *J. ACM*, Vol. 17, No. 3, July 1970.

[18] Hayes, Patrick J., **A Machine-oriented Formulation of the Extended Functional Calculus**, Stanford A. I. Memo AIM-86, June, 1969.

[19] Manna, Zohar, and McCarthy, John, **Properties of Programs and Partial Function Logic**, Stanford A. I. Memo AIM-100, October 1969; also in Meltzer, B. and Michie, D. (eds.), *Machine Intelligence 5*, Edinburgh University Press, Edinburgh, 1970.

[20] Manna, Zohar, **The Correctness of Non-deterministic Programs**, Stanford A. I. Memo AIM-95, August 1969; also in *Artificial Intelligence*, Vol. 1, No. 1, 1970.

[21] Ashcroft, Edward, and Manna, Zohar, **Formalization of Properties of Parallel**

Programs, Stanford A. I. Memo AIM-110, February 1970; also in *Machine Intelligence 6*, Edinburgh University Press, Edinburgh, 1971.

[22] Manna, Zohar, Second-order Mathematical Theory of Computation, Stanford A. I. Memo AIM-111, March 1970; also in *Proc. ACM Symposium on Theory of Computing*, May 1970.

[23] Igarashi, Shigeru, Semantics of Algol-like Statements, Stanford A. I. Memo AIM-129, June 1970.

[24] Hoare, C.A.R., An Axiomatic Basis for Computer Programming, *Comm. ACM 12*, No. 10, pp.576-580, 1969.

[25] Milner, Robin, An Algebraic Definition of Simulation between Programs, Stanford A. I. Memo AIM-142, February, 1971; also in *Proc. IJCAI*, British Computer Society, London, 1971.

[26] Allen, John and Luckham, David, An Interactive Theorem-proving Program, Stanford A. I. Memo AIM-103, October 1971.

[27] Milner, Robin, Logic for Computable Functions; Description of a Machine Implementation, Stanford A. I. Memo AIM-169, May 1972.

[28] Milner, Robin, Implementation and Applications of Scott's Logic for Computable Functions, *Proc. ACM Conf. on Proving Assertions about Programs*, ACM Sigplan Notices, January 1972.

[29] Milner, Robin and Weyhrauch, Richard, Proving Compiler Correctness in a Mechanised Logic, *Machine Intelligence 7*, Edinburgh University Press, Edinburgh, 1972.

[30] Weyhrauch, Richard and Milner, Robin, Program Semantics and Correctness in a Mechanized Logic, *Proc. USA-Japan Computer Conference*, Tokyo, 1972.

[31] Floyd, Robert W., Toward Interactive Design of Correct Programs, Stanford A. I. Memo AIM-150, September 1971; also in *Proc. IFIP Congress 1971*.

[32] Manna, Zohar, Ness. Stephen, and Vuillemin, Jean, Inductive Methods for Proving Properties of Programs, Stanford A. I. Memo AIM-154, November 1971; also in *ACM Sigplan Notices, Vol. 7, No. 1*, January 1972.

[33] Ashcroft, Edward, and Manna, Zohar, The Translation of 'GO-TO' Programs to 'WHILE' Programs, Stanford A. I. Memo AIM-138, January 1971 also in *Proc. IFIP Congress 1971*.

[34] Manna, Zohar, Mathematical Theory of Partial Correctness, Stanford A. I. Memo AIM-139, January 1971; also i *J. Comp. and Sys. Sci.*, June 1971.

[35] London, Ralph L., Correctness of Two Compilers for a LISP Subset, Stanford A. I. Memo AIM-151, October 1971.

[36] Ashcroft, Edward, Manna, Zohar, and Pneuli, Amir, Decidable Properties of Monadic Functional Schemas, Stanford A. I. Memo AIM-148, July 1971.

[37] Chandra, Ashok, and Manna, Zohar, Program Schemas with Equality, Stanford A. I. Memo AIM-158, December 1971.

[38] Manna, Zohar, *Introduction to Mathematical Theory of Computation*, McGraw-Hill, New York, 1974 (to appear).

[39] Igarashi, Shigeru, London, Ralph, Luckham, David, Automatic Program

Verification I: A Logical Basis and its Implementation, Stanford A. I. Memo AIM-200, May 1973.

2.2.2 Representation Theory

When we try to make a computer program that solves a certain class of problem, our first task is to decide what information is involved in stating the problem and is available to help in its solution. Next we must decide how this information is to be represented in the memory of the computer. Only then can we choose the algorithms for manipulating this information to solve our problem. *Representation theory* deals with what information we need and how it is represented in the computer. *Heuristics* is concerned with the structure of the problem solving algorithms.

In the past, much work in artificial intelligence has been content with a rather perfunctory approach to representations. A representation is chosen rather quickly for a class of problems and then all attention is turned to devising, programming, and testing heuristics. The trouble with this approach is that the resulting programs lack generality and are not readily modifiable to attack new classes of problems.

The first goal of representation theory is to devise a general way of representing information in the computer. It should be capable of representing any state of partial information necessary to solve it. In 1958, McCarthy posed the problem of making a program with "common sense" in approximately these terms and suggested using sentences in an appropriate formal language to represent what the program knows [1]. The advantage of representing information by sentences is that sentences have other sentences as logical consequences and the program can find consequences relevant to the goals at hand. Thus, representation of information by sentences allows the following.

1. A person can instruct the system without detailed knowledge of what sentences are already in memory. That is, the procedures for solving a problem using information in sentence form do not require that the information be in a particular order, nor even a particular grouping of information into sentences. All they require is that what to do is a logical consequence of the collection of sentences.

2. Similar considerations apply to information generated by the program itself.

3. Representing information by sentences seems to be the only clean way of separating that information which is common knowledge (and so should be already in the system) from information about a particular problem.

On the other hand, because each sentence has to carry with it much of its frame of reference, representation of information by sentences is very voluminous. It seems clear that other forms of information (e.g. tables) must also be used, but the content of these other forms should be described by sentences.

In the last ten years, considerable progress has been made in the use of the sentence representation. In the heuristic direction, theorem proving and problem solving programs based on J. Allen Robinson's resolution have been developed by both Green and Luckham, among others [see Section 2.3.1]. The theory of how to represent facts concerning causality, ability, and knowledge for artificial intelligence has been developed mainly by McCarthy and his students.

The early work in this project revolved around McCarthy's "advice taker" ideas [2-7]. McCarthy and Hayes [8] restructured the problem and connected this work to the subject of philosophical logic.

Later related work includes Becker's semantic memory system [9, 10], Sandewall's representation of natural language information in predicate calculus [11], and Hayes' study of the frame problem [12].

Bibliography

[1] John McCarthy, **Programs with Common Sense** in *Proc. 1958 Teddington Conf. on Mechanisation of Thought Processes*. Vol. 1, pp 77-84, H. M. Stationary Office, London, 1960; reprinted in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, Mass., 19[illegible]

[2] John McCarthy, **Situations, Actions, and Causal Laws**, Stanford A. I. Memo AIM-2, July 1963.

[3] F. Safier, **'The Mikado' as an Advice Taker Problem**, Stanford A. I. Memo AIM-3, July 1963.

[4] John McCarthy, **Programs with Common Sense**, Stanford A. I. Memo AIM-7, September 1963.

[5] M. Finkelstein and F. Safier, **Axiomatization and Implementation**, Stanford A. I. Memo AIM-15, June 1964.

[6] John McCarthy, **Formal Description of the Game of Pang-ke**, Stanford A. I. Memo AIM-17, July 1964.

[7] Barbara Huberman, **Advice Taker and GPS**, Stanford A. I. Memo AIM-33, June 1965.

[8] John McCarthy and Patrick Hayes, **Some Philosophical Problems from the Standpoint of Artificial Intelligence**, Stanford A. I. Memo AIM-73, November 1968; also in D. Michie (ed.), *Machine Intelligence 4*, American Elsevier, New York, 1969.

[9] Joseph Becker, **The Modeling of Simple Analogic and Inductive Processes in a Semantic Memory System**, Stanford A. I. Memo AIM-77, January 1969; also in *Proc. IJCAI*, Washington D. C., 1969.

[10] Joseph Becker, **An Information-processing Model of Intermediate-level Cognition**, Stanford A. I. Memo AIM-119, May 1970.

[11] Erik Sandewall, **Representing Natural-language Information in Predicate Calculus**, Stanford A. I. Memo AIM-128, July 1970.

[12] Patrick Hayes, **The Frame Problem and Related Problems in Artificial Intelligence**, Stanford A. I. Memo AIM-153, November 1971.

2.2.3 Grammatical Inference

Professor Feldman and a small group have been investigating the problem of grammatical inference. That is, given a set of strings which has been chosen in a random way from a formal language such as a context-free language, to make a "reasonable" inference of the grammar for the language.

Feldman has studied a very general class of complexity measures and shown that the least complex choice of grammar from an enumerable class of grammars (which are general rewriting systems), can be found, and he gives an algorithm for discovering it [3]. This approach is also being applied to the inference of good programs for producing specified input-output behavior.

Bibliography

[1] Jerome A. Feldman, **First Thoughts on Grammatical Inference**, Stanford A. I. Memo AIM-55, August 1967.

[2] Jerome A. Feldman, J. Gips, J. J. Horning, S. Reder, **Grammatical Complexity and Inference**, Stanford A. I. Memo AIM-89, June 1969.

[3] Jerome A. Feldman, **Some Decidability Results on Grammatical Inference and Complexity**, Stanford A. I. Memo AIM-93, August 1969; revised May 1970; also in *Information and Control*, Vol. 20, No. 3, pp. 244-262, April 1972.

[4] James Jay Horning, **A Study of Grammatical Inference**, Stanford A. I. Memo AIM-98, August 1969.

[5] Alan W. Biermann, J. A. Feldman, **On the Synthesis of Finite-state Acceptors**, Stanford A. I. Memo AIM-114, April 1970.

[6] Alan W. Biermann, **On the Inference of Turing Machines from Sample Computations**, Stanford A. I. Memo AIM-152, October 1971; also in *Artificial Intelligence J.*, Vol. 3, No. 3, Fall 1972.

[7] Alan W. Biermann, **On the Synthesis of Finite-state Machines from Samples of their Behavior**, *IEEE Trans. Computers*, Vol. C-21, No. 6, June 1972.

[8] Jerome A. Feldman, A. W. Biermann, **A Survey of Grammatical Inference**, *Proc. Int. Congress on Pattern Recognition*, Honolulu, January 1971; also in S. Watanabe (ed.), *Frontiers of Pattern Recognition*, Academic Press, 1972.

[9] Jerome A. Feldman, Paul Shields, **Total Complexity and Inference of Best Programs**, Stanford A. I. Memo AIM-159, April 1972.

[10] Jerome A. Feldman, **Automatic Programming**, Stanford A. I. Memo AIM-160, February 1972.

2.3 Heuristic Programming

Heuristic programming techniques are a core discipline of artificial intelligence. Work on theorem proving, program generation, board games, and symbolic computation make particularly heavy use of these techniques. An excellent general reference is the book by Nilsson [1]. Nilsson, of SRI, was supported in part by our project while writing it.

2.3.1 Theorem Proving

The basis of the theorem-proving effort has been the proof procedure for first-order logic with equality, originally developed by Allen and Luckham [2]. This has been extensively modified by J. Allen in recent months; the basic theorem-proving program has been speeded up by a factor of 20, an input-output language allowing normal mathematical notation has been added, and the program will select search strategies automatically if the user wishes (we refer to this as automatic mode). As a result it is possible for the program to be used by a person who is totally unfamiliar with the theory of the Resolution principle and its associated rules of inference and refinements. A user's manual is now available [10].

This program has been used to obtain proofs of several different research announcements in the Notices of the American Mathematical Society, for example, [7, 8, and 9]. More recently (July 1972), J. Morales learned to use the program essentially by using it to obtain proofs of the results stated in [9] as an exercise. In the course of doing this he was able to formulate simple ways of using the prover to generate possible new theorems in the same spirit as [9], and did in fact succeed in extending the results of [9]. Furthermore, he was able to send the authors proofs of their results before they had actually had time to write them up [R. H. Cowen, private correspondence, August 9, 1972]. Currently, Morales has been applying

the theorem-prover to problems in geometry [11, 12] that have been the subject of recent publications in the proceedings of the Polish National Academy of Sciences. He has been able to give elementary proofs of some results to clarify inaccuracies and omissions. This work is continuing in correspondence with the authors. The prover is also being used as part of the program verification system [see Section 2.2.1].

A version of the prover, with user documentation [10], has been prepared for distribution to other research groups. The addition of a language in which the user can specify his intuition about how a proof of a given statement might possibly be obtained, is in progress. J. Allen has already programmed a very preliminary version of this "HUNCH" language, and has completed the systems debugging necessary to get a compiled version of Sussman's CONNIVER language running here. HUNCH language may be implemented in CONNIVER, but discussions on this point are not yet complete. Initially, it is expected that HUNCH will be useful in continuing with more difficult applications in mathematics.

An alternative approach to theorem proving was developed in Cordell Green's Thesis on QA3. His work was done at SRI and his dissertation was published here [5].

Bibliography

[1] Nils Nilsson, *Problem-solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.

[2] John Allen and David Luckham, **An Interactive Theorem-proving Program**, *Machine Intelligence 5*, 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 and Nils J. Nilsson, **Extracting Information from Resolution Proof Trees**, *Artificial Intelligence*, Vol. 2, No. 1, pp. 27-54, Spring 1971.

[5] Cordell Green, **The Application of Theorem Proving to Question Answering Systems**, Ph.D. Thesis in E. E., Stanford A. I. Memo AIM-96, August 1969.

[6] J. Sussman and T. Winograd, **Micro Planner Manual**, Project MAC Memo, MIT.

[7] Chinthayamma, **Sets of Independent Axioms for a Ternary Boolean Algebra**, *Notices Amer. Math. Soc.*, 16, p. 654, 1969.

[8] E. L. Marsden, **A Note on Implicative Models**, *Notices Amer. Math. Soc.*, No. 682-02-7, p. 89, January 1971.

[9] Robert H. Cowen, Henry Frisz, Alan Grenadir, **Some New Axiomatizations in Group Theory**, Preliminary Report, *Notices Amer. Math. Soc.*, No. 72T-112, p. 547, June 1972.

[10] J. R. Allen, **Preliminary Users Manual for an Interative Theorem-Prover**, Stanford Artificial Intelligence Laboratory Operating Note SAILON-73, 1973.

[11] L. Szcerba and W. Szmielew, **On the Euclidean Geometry Without the Pasch Axiom**, *Bull. Acad. Polon. Sciences*, Ser. Sci. Math., Astronm, Phys., 18, pp. 659-666, 1970.

[12] Szcerba, L. W., **Independence of Pasch's Axiom**, *ibid.* [3], pp. 491-498.

2.3.2 Automatic Program Generation

This work is an outgrowth of an attempt to extend the applicability of the theorem-prover to problems in artificial intelligence, and makes use of a particular notion of a problem environment (called a "semantic frame") which was designed for that original purpose. However, the system as it now stands, is independent of the theorem-prover, and is best thought of as a heuristic problem-solver for a subset of Hoare's logical system. It has been implemented in LISP by J. Buchanan using the backtrack features of Micro-Planner.

It accepts as input an environment of programming methods and a problem and, if successful, gives as output a program for solving the problem. At the moment, the output programs are composed of the primitive operators of the environment, assignments, conditional branches, while loops, and non-recursive procedure calls. This system has been used to generate many programs for solving various problems in robot control, everyday advice-taking and planning, and for computing arithmetical functions. It is an interactive facility and incorporates an Advice language which allows the user to state preferences affecting the output program and heuristics for speeding the problem-solving process, and also to make assumptions.

The system is intended to permit a user to oversee the automatic construction of a program according to the current principles of structured programming. There is also a library facility and the system will incorporate library routines into the program under construction. The details of its implementation will be available in Jack Buchanan's Ph.D. Thesis (in preparation).

2.3.3 Board Games

Computer programs that play games such as chess or checkers are deservedly important as research vehicles. A Russian computer scientist has said [1] that chess plays the role in Artificial Intelligence that the fruit fly plays in genetics. Just as the genetics of *Drosophila* are studied not to breed better flies but to study the laws of heredity, so we write chess programs, not because it is important that computers play good chess, but because chess provides a rather clear basis for comparing our ideas about reasoning processes with human performance. Weaknesses in the performance of the program tell us about human mental processes that we failed to identify

John McCarthy supervised the development of a chess program at MIT. He brought it here when he came and subsequently improved it a bit. In 1966, a match was held with a program at what was then Carnegie Tech. Neither program played especially well, but Carnegie eventually resigned. The Stanford program played several games in 1967 with a program at the Institute for Theoretical and Experimental Physics in Moscow. Again, both blundered frequently, but ours made some of the worst moves. By prior agreement, the games were not completed because both programs were known to have essentially no end-game capability.

Barbara Huberman developed a program that handled certain important end game situations and published her dissertation in 1968 [5].

Some early work on the game of Kalah produced a program that played rather well [2, 3].

Arthur Samuel continued his long-standing development of a checkers program when he

arrived in 1966 [4]. His program is apparently still the best in the world. It does not defeat the best human players, but plays fairly well against them. Samuel subsequently decided to apply the "signature table learning" scheme that he had developed for checkers to speech recognition problems [see Section 2.4.1].

Jonathan Ryder developed what was (and probably still is) the best Go program to date [6]. It plays better than a beginning human, but it is still quite weak by the standards of experienced players.

Since 1971, we have done very little work on board games.

Bibliography

[1] A. Kronrod, **The Computer becomes More Intelligent**, *Isvestiya*, March 15, 1967; translated in *Soviet Cybernetics: Recent News Items*, No. 3, Rand Corp., Santa Monica, Calif., April 1967.

[2] R. Russell, **Kalah -- the Game and the Program**, Stanford A. I. Memo AIM-22, September 1964.

[3] R. Russell, **Improvements to the Kalah Program**, Stanford A. I. Memo AIM-23, September 1964.

[4] Arthur Samuel, **Some Studies in Machine Learning using the Game of Checkers, II -- Recent Progress**, Stanford A. I. Memo AIM-52, June 1967; also in *IBM Journal*, November 1967.

[5] Barbara J. Huberman, **A Program to Play Chess End Games**, Ph.D. Dissertation in Computer Science, Stanford A. I. Memo AIM-65, August 1968.

[6] Jonathan L. Ryder, **Heuristic Analysis of Large Trees as Generated in the Game of Go**, Ph.D. Dissertation in Computer Science, Stanford A. I. Memo AIM-155, December 1971.

2.3.4 Symbolic Computation

The use of computers to manipulate algebraic expressions and solve systems of symbolic equations potentially offers substantial improvements in speed and reduced error rate over pencil-and-paper methods. As a consequence, it becomes possible to tackle problems that are out of the range of practical human capabilities.

Beginning in 1963, Enea and Wooldridge worked on the central problem of algebraic simplification [1, 2]. By 1965, Korsvold had developed a working system [3].

At about this time, Hearn became interested in the problem because of potential application to problems in particle physics. He developed a system called REDUCE, which was written in LISP [4, 5]. Its initial capabilities included:

a) expansion and ordering of rational functions of polynomials,
b) symbolic differentiation,
c) substitutions in a wide variety of forms,
d) reduction of quotients of polynomials by cancellation of common terms,
e) calculation of symbolic determinates.

REDUCE has been used for analysis of Feynman Diagrams and a number of other problems in physics and engineering [6, 7, 12]. It has been extended in a number of ways [8-11, 13, 14] and is still under development by Hearn at the University of Utah.

George Collins spent a sabbatical year here (1972-3) developing his computational system [15, 16].

Bibliography

[1] Enea, H. and Wooldridge, D. **Algebraic Simplication**, Stanford A. I. Memo AIM-5, August 1963.

[2] Wooldridge, D., **An Algebraic Simplify Program in LISP**, Stanford A. I. Memo AIM-11, December 1963.

[3] Korsvold, K., **An On Line Algebraic Simplification Program**, Stanford A. I. Memo AIM-37, November 1965.

[4] Hearn, A., **Computation of Algebraic Properties of Elementary Particle Reactions Using a Digital Computer**, *Comm. ACM* 9, August 1966.

[5] Hearn, A., **REDUCE Users' Manual**, Stanford A. I. Memo AIM-50, February 1967.

[6] Brodsky, S. and Sullivan, J., **W-Boson Contribution to the Anomalous Magnetic Moment of the Muon**, *Physics Review*, 156, 1644, 1967.

[7] Campbell, J., **Algebraic Computation of Radiative Corrections for Electron-Proton Scattering**, *Nuclear Physics* Vol. B1, 1967.

[8] Hearn, A., **REDUCE, A User-Oriented Interactive System for Algebraic Simplification**, *Proceedings for ACM Symposium on Interactive Systems for Experimental Applied Mathematics*, August 1967.

[9] Hearn, A., **REDUCE, A User-Oriented Interactive System for Algebraic Simplification**, Stanford A. I. Memo AIM-57, October 1967.

[10] Hearn, A., **The Problem of Substitution**, *Proceedings of IBM Summer Institute on Symbolic Mathematics by Computer*, July 1968.

[11] Hearn, A., **The Problem of Substitution**, Stanford A. I. Memo AIM-170, December 1968.

[12] Hearn, A. and Campbell, J.A., **Symbolic Analysis of Feynman Diagrams by Computer**, Stanford A. I. Memo AIM-91, August 1969; also in *Journal of Computational Physics* 5, 1970.

[13] Hearn, A., **Applications of Symbol Manipulation in Theoretical Physics**, *Comm. ACM*, August 1971.

[14] Hearn, A., **REDUCE 2**, Stanford A. I. Memo AIM-133, October 1970.

[15] Collins, George, **The Computing Time of the Euclidean Algorithm**, Stanford A. I. Memo AIM-187, January 1973.

[16] Collins, George, and Horowitz, Ellis, **The Minimum Root Separation of a Polynomial**, Stanford A. I. Memo AIM-192, April 1973.

2.4 Natural Language

We have worked on two aspects of natural language understanding that are still quite distinct, but may be expected to interconnect eventually.

2.4.1 Speech Recognition

Efforts to establish a vocal communication link with a digital computer have been underway at Stanford since 1963. These efforts have been primarily concerned with four areas of research:

1) basic research in extracting phonemic and linguistic information from speech waveforms,
2) the application of automatic learning processes,
3) the use of syntax and semantics to aid speech recognition, and
4) speech recognition systems have been used to control other processes.

These efforts have been carried out in parallel with varying emphasis at different times.

The fruits of Stanford's speech research program were first seen in October 1964 when Raj Reddy published a report describing his preliminary investigations on the analysis of speech waveforms [1]. His system worked directly with a digital representation of the speech waveform to do vowel recognition.

By 1966 Reddy had built a much larger system which obtained a phonemic transcription and which achieved segmentation of connected phrases utilizing hypotheses testing [2]. This system represented a significant contribution towards speech sound segmentation [3]. It operated on a subset of the speech of a single cooperative speaker.

In 1967 Reddy and his students had refined several of his processes and published papers on phoneme grouping for speech recognition [4], pitch period determination of speech sounds [5], and computer recognition of connected speech [6].

1968 was an extremely productive year for the Speech group. Pierre Vicens developed an efficient preprocessing scheme for speech analysis [7]. Reddy and his students published papers on transcription of phonemic symbols [8], phoneme-to-grapheme translation of English [9], segmentation of connected speech [10], and consonantal clustering and connected speech recognition [11]. A general paper by John McCarthy, Lester Earnest, Raj Reddy, and Pierre Vicens described the voice-controlled artificial arm developed at Stanford [12].

By 1969 the speech recognition processes were successfully segmenting and parsing continuous utterances from a restricted syntax [13, 14, 15]. A short film entitled "Hear Here" was made to document recent accomplishments.

In mid 1970, Prof. Reddy left Stanford to join the faculty of Carnegie-Mellon University and Arthur Samuel became the head of the Stanford speech research efforts. Dr. Samuel had developed a successful machine learning scheme which had previously been applied to the game of checkers [16], [17]. He resolved to apply them to speech recognition.

By 1971 the first speech recognition system utilizing Samuel's learning scheme was reported by George White [18]. This report was primarily concerned with the examination of the properties of signature trees and the heuristics involved in their application to an optimal minimal set of features to achieve recognition. Also at this time, M. M. Astrahan described his hyperphoneme method [19], which attempted to do speech recognition by mathematical

classifications instead of the traditional phonemes or linguistic categories. This was accomplished by nearest-neighbor classification in a hyperspace wherein cluster centers, or hyperphonemes, had been established.

In 1972 R. B. Thosar and A. L. Samuel presented a report concerning some preliminary experiments in speech recognition using signature tables [20]. This approach represented a general attack on speech recognition employing learning mechanisms at each stage of classification.

The speech effort in 1973 has been devoted to two areas. First, a mathematically rigorous examination and improvement of the signature table learning mechanism has been accomplished by R. B. Thosar. Second, a segmentation scheme based on signature tables is being developed to provide accurate segmentation together with probabilities or confidence values for the most likely phoneme occuring during each segment. This process attempts to extract as much information as possible and to pass this information to higher level processes.

In addition to these activities, a new, high speed pitch detection scheme has been developed by J. A. Moorer and has been submitted for publication [22].

Bibliography

[1] D. Raj Reddy, **Experiments on Automatic Speech Recognition by a Digital Computer**, Stanford A. I. Memo AIM-26, October 1964.

[2] D. Raj Reddy, **An Approach to Computer Speech Recognition by Direct Analysis of the Speech Waveform**, Stanford A. I. Memo AIM-43, September 1966.

[3] D. Raj Reddy, **Segmentation of Speech Sounds**, *J. Acoust. Soc. Amer.*, August 1966.

[4] D. Raj Reddy, **Phoneme Grouping for Speech Recognition**, *J. Acoust. Soc. Amer.*, May, 1967.

[5] D. Raj Reddy, **Pitch Period Determination of Speech Sounds**, *Comm. ACM*, June, 1967.

[6] D. Raj Reddy, **Computer Recognition of Connected Speech**, *J. Acoust. Soc. Amer.*, August, 1967.

[7] Pierre Vicens, **Preprocessing for Speech Analysis**, Stanford A.I. Memo AIM-71, October 1968.

[8] D. Raj Reddy, **Computer Transcription of Phonemic Symbols**, *J. Acoust. Soc. Amer.*, August 1968.

[9] D. Raj Reddy, and Ann Robinson, **Phoneme-To-Grapheme Translation of English**, *IEEE Trans. Audio and Electroacoustics*, June 1968.

[10] D. Raj Reddy, and P. Vicens, **Procedures for Segmentation of Connected Speech**, *J. Audio Eng. Soc.*, October 1968.

[11] D. Raj Reddy, **Consonantal Clustering and Connected Speech Recognition**, *Proc. Sixth International Congress of Acoustics*, Vol. 2, pp. C-57 to C-60, Tokyo, 1968.

[12] John McCarthy, Lester Earnest, D. Raj Reddy, and Pierre Vicens, **A Computer With Hands, Eyes, and Ears**, *Proc. FJCC*, 1968.

[13] Pierre Vicens, **Aspects of Speech Recognition by Computer**, Stanford A. I. Memo AIM-85, April 1969.

[14] D. Raj Reddy, **On the Use of Environmental, Syntactic and Probabilistic Constraints in Vision and Speech**, Stanford A. I. Memo AIM-78, January 1969.

[15] D. Raj Reddy and R. B. Neely, **Contextual Analysis of Phonemes of English**, Stanford A. I. Memo AIM-79, January 1969.

[16] A. L. Samuel, **Some Studies in Machine Learning Using the Game of Checkers**, *IBM Journal* 3, 211-229, 1959.

[17] A. L. Samuel, **Some Studies in Machine Learning Using the Game of Checkers, II - Recent Progress**, *IBM Jour. of Res. and Dev.*, 11, pp. 601-617, November 1967.

[18] George M. White, **Machine Learning Through Signature Trees. Applications to Human Speech**, Stanford A. I. Memo AIM-136, October 1970.

[19] M. M. Astrahan, **Speech Analysis by Clustering, or the Hyper-phoneme Method**, Stanford A. I. Memo AIM-124, June 1970.

[20] R. B. Thosar and A. L. Samuel, **Some Preliminary Experiments in Speech Recognition Using Signature Table Learning**, ARPA Speech Understanding Research Group Note 43, 1972.

[21] R. B. Thosar, **Estimation of Probability Densities using Signature Tables for Application to Pattern Recognition**, ARPA Speech Understanding Research Group Note 81.

[22] J. A. Moorer, **The Optimum-Comb Method of Pitch Period Analysis in Speech**, Stanford A. I. Memo AIM-207, July 1973.

FILM

23 Raj Reddy, Dave Espar, Art Eisenson, **Hear Here**, 16mm color with sound, 1969.

2.4.2 Semantics

We have two sub-groups working within the general area of machine translation and natural language understanding. They are the Preference Semantics group (Wilks et al.) and the Conceptual Dependency group (Schank et al.)

Both groups stress the centrality of meaning representation and analysis, and the inferential manipulation of such representations, for the task of natural language understanding. Again, both assume the importance of context, and of the expression of meaning in terms of a formal system of meaning primitives, while at the same time denying the centrality of syntactical analysis in the conventional linguist's sense. Both have capacity for combining linguistic knowledge with knowledge of the real world, in order to solve problems of interpretation during the course of analysis. The two basic formalisms were set out definitively in [11 and 14], and the theoretical basis for systems of this sort has been argued in [9, 11, 13].

The differences between the sub-groups are largely concerned with the ordering of research goals: the PS [Preference Semantics] group emphasises an immediate task as a criterion of successful understanding, interlingual machine translation in the present case, while the CD [Conceptual Dependency] group emphasises independence from any particular task. A particular example of the difference is provided by machine translation, where, from a given semantic representation a PS program always constructs some particular output string for that representation, whereas a CD program would want the freedom to construct one of a number of possible representations.

Another, more philosophically motivated difference of approach between the groups, concerns the time at which inferences are to be made from semantic representations. In the PS approach, no more inferences are made than is necessary to solve the problem in hand, and the representation is never made any "deeper" than necessary. In the CD approach, a large number of inferences is made spontaneously.

Lastly, the two approaches both aim essentially at machine translation as the fundamental test of a language understander, but the CD approach emphasizes the treatment of human dialogue while the PS approach emphasizes the treatment of small texts as the definition of a context.

We believe that these approaches are complementary rather than exclusive, and that the parallel testing of different sub-hypotheses in this way, within an overall semantics-oriented project, can be stimulating and productive.

Both sub-groups have had pilot systems running on the computer since 1970, but in the last year both have implemented much stronger running programs. During the last year the PS implementation has been enlarged from a basic mode to an extended mode of the on-line English-French translator. It accepts small paragraphs of English, outputs the semantic representation it derives, and then the French translation. Since it is semantics, rather than syntax, based, the English input need not even be grammatically correct.

The basic mode of Preference Semantics will cope with the problems presented by sentences like "Give the bananas to the monkeys although they are not ripe, because they are very hungry"; that is to say, of attaching the two "they"s correctly on the basis of what it knows about bananas and monkeys (which have different genders in French, so the choice is important).

The extended mode is called when referential problems in a paragraph require the manipulation of stronger partial information about the real world, as in "The soldiers fired at the women and I saw several fall", where the referent of "several" depends on our relatively unreliable knowledge of how the world works. This is done by extracting inferential information from the basic representation with common-sense inference rules, that are themselves expressed within the semantic notation.

During the last year, the CD group has brought together three programs to implement MARGIE: Meaning Analysis, Response Generation and Inference on English. MARGIE is a prototype system, presently undergoing concurrent development here at Stanford, and at the Istituto per gli studi semantici e cognitivi in Castagnola, Switzerland.

MARGIE's three component programs (analyzer, memory/inferencer, generator) (a) analyze English utterances in context into graphs in the deep conceptual base prescribed by Conceptual Dependency, (b) generate inferences spontaneously from the graphs and build relational networks which demonstrate an ability to understand what was said, and (c) generate responses, constructed in this deep conceptual base, back into surface (syntax) networks, and finally back into syntactically correct English sentences. In addition, the analyzer and generator may be run coupled directly together, providing a paraphraser which paraphrases the meaning of an English sentence rather than simply its syntactic form.

For example, from the input "John killed Mary by choking Mary", the paraphraser will return with a number of sentences examples of which are "John strangled Mary" and "John choked Mary and she died because she could not breathe".

In the "Inference mode" the MARGIE system takes input such as "John gave Mary a beating with a stick" and produces a number of inferred output sentences such as "A stick touched Mary", "Mary became hurt " and so on.

Bibliography

1. Goldman, N., Riesbeck, C., A Conceptually Based Sentence Paraphraser, Stanford Artificial Intelligence Memo AIM-196, May 1973.

2. Goldman, N., Computer Generation of Natural Language from a Deep Conceptual Base, Ph.D. Thesis in Computer Science, Stanford University (forthcoming)

3. Herskovits, A., The generation of French from a semantic representation, Stanford Artificial Intelligence Memo AIM-212, October 1973.

4. Rieger, C., Conceptual Memory: A Theory for Processing the Meaning Content of Natural Language Utterances, Ph.D. Thesis in Computer Science, Stanford University (forthcoming).

5. Riesbeck, C., Computer Analysis of natural Language in Context, Ph.D. Thesis in Computer Science, Stanford University (forthcoming).

6. Schank, R., Goldman, N., Rieger, C., Riesbeck, C., Primitive Concepts Underlying Verbs of Thought, Stanford Artificial Intelligence Memo AIM-162, Feb. 1972.

7. Schank, R., Goldman, N., Rieger, C., Riesbeck, C., MARGIE: Memory, Analysis, Response Generation and Inference on English, *Advance Papers of the Third International Joint Conference on Artificial Intelligence*, Stanford U., August 1973.

8. Schank, R., Rieger, C., Inference and the Computer Understanding of Natural Language, Stanford Artificial Intelligence Memo AIM-197, May 1973.

9. Schank, R., Wilks, Y., The Goals of Linguistic Theory Revisited, Stanford Artificial Intelligence Memo AIM-202, May 1973.

10. Schank, R., Finding the Conceptual Content and Intention in an Utterance in Natural Language Conversation, *Advance Papers of the Second International Joint Conference On Artificial Intelligence*, British Comp. Soc., London, 1971.

11. Schank, R., Conceptual Dependency: a Theory of Natural Language Understanding, *Cognitive Psychology*, Vol. 3, No. 4, 1972.

12. Schank, R., The Fourteen Primitive Actions and Their Inferences, Stanford Artificial Intelligence Memo AIM-183, February 1973.

13. Wilks, Y., Decidability and Natural Language, *Mind*, December 1971.

14. Wilks, Y., *Grammer, Meaning and the Machine Analysis of Language*, Routledge, London, 1972.

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

16. Wilks, Y., An Artificial Intelligence Approach to Machine Translation, Stanford Artificial Intelligence Memo AIM-161, February 1972; to appear in Schank and Colby (eds.), *Computer*

Models of Thought and Language, W. H. Freeman, San Francisco, 1973.

17. Wilks, Y., **Understanding Without Proofs**, *Advanced Papers of the Third International Joint Conference on Artificial Intelligence*, Stanford U., August 1973.

18. Wilks, Y., and Herskovits, A., **An Intelligent Analyser and Generator for Natural Language**, *Proceedings of the International Conference on Computational Linguistics*, Pisa, 1973.

19. Wilks, Y., **Preference Semantics**, Stanford Artificial Intelligence Memo AIM-206, July 1973; also in E. Keenan (ed.), *Proc. 1973 Colloquium on Formal Semantics of Natural Language*, Cambridge, U.K., 1974 (to appear).

2.5 Programming Languages

There is a strong connection between what you can say and the language in which you say it. Advances in artificial intelligence, as well as other branches of computer science, are frequently linked to advances in programming languages. We have found it advantageous to invest a part of our effort in language development and the corresponding compilers and run-time packages.

We have already discussed the development of a "hand language" in support of robotics research [Section 2.1.1]. This section discusses more general programming language developments that have taken place in our laboratory.

Feldman and Gries published an analytical survey of translator (e.g. compiler) writing systems in 1968 [3]. Donald Knuth continued publication of a series of texts on computer programming that has become the standard reference in the field [4, 5, 6]. He also published reports on language definition schemes [7], a study of "real world" programming practices [8, summarized in 9] and an interesting historical study [10].

Bibliography

[1] J. Hext, **Programming Languages and Translation**, Stanford A. I. Memo AIM-19, August 1964.

[2] D. Raj Reddy, **Source Language Optimization of FOR-loops**, Stanford A. I. Memo AIM-20, August 1964.

[3] Jerome Feldman and David Gries, **Translator Writing Systems**, *Comm. ACM*, February 1968.

[4] Donald E. Knuth, *The Art of Computer Programming, Vol. 1, Fundamental Algorithms*, Addison-Wesley, Menlo Park, Calif., 1968.

[5] Donald E. Knuth, *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*, Addison-Wesley, Menlo Park, Calif., 1969.

[6] Donald E. Knuth, *The Art of Computer Programming, Vol. 3, Sorting and Searching*, Addison-Wesley, Menlo Park, Calif., 1973.

[7] Donald E. Knuth, **Examples of Formal Semantics**, Stanford A. I. Memo AIM-126, July 1970.

[8] Donald E. Knuth, **An Empirical Study of Fortran in Use**, Stanford A. I. Memo AIM-137, November 1970.

[9] Donald E. Knuth, **An Empirical Study of Fortran Programs**, *Software -- Practice and Experience*, Vol. 1, 105-133, 1971.

[10] Donald E. Knuth, **Ancient Babylonian Algorithms**, *Comm. ACM*, July 1972.

2.5.1 LISP

LISP is the most widely used language in artificial intelligence research. The overall design of this programming system was defined in John McCarthy's 1960 article [11]. LISP 1.5, developed initially at MIT [12], became available on nearly every major computer and remains so today.

Of course, the various versions of "LISP 1.5" turned out to be not quite compatible. Even so, Tony Hearn devised a LISP subset that is fairly portable [17]. This facilitated distribution of his REDUCE system for symbolic computation [see Section 2.3.4].

There was an early attempt to design an advanced language called LISP 2 [13, 16]. The implementation of a compiler was to be done by a group at System Development Corperation. Unfortunately, they elaborated it to death.

Stanford LISP 1.6 was initially developed in 1966 from an MIT PDP-6 system. Our staff essentially rewrote the system several times since [18] and have distributed it through DECUS. It is currently in use at dozens of PDP-6 and PDP-10 installations.

A group at U. C. Irvine has substantially augmented Stanford LISP in the direction of BBN LISP. The result is a very convenient and powerful system [19].

One of our groups developed the MLISP compiler [20, 21, 22, 23], which offers an ALGOL-like syntax on top of all the standard LISP features. This language is in use by investigators in artificial intelligence in various parts of this country and at least one place in Europe.

More recently, the same group has been working on a powerful new language called LISP70 [24]. It allows pattern-directed computation and extensibility, i.e. the user can add his own rewrite rules for new functions, which will automatically be merged and ordered in the existing rules.

Bibliography

[11] John McCarthy, **Recursive Functions of Symbolic Expressions**, *Comm. ACM*, April 1960.

[12] John McCarthy, et al, *LISP 1.5 Programmer's Manual*, MIT Press, 1962.

[13] John McCarthy, **Storage Conventions in LISP 2**, Stanford A. I. Memo AIM-8, September 1963.

[14] S. R. Russell, **Improvements in LISP Debugging**, Stanford A. I. Memo AIM-10, December 1963.

[15] J. Hext, **An Expression Input Routine for LISP**, Stanford A. I. Memo AIM-18, July 1964.

[16] R. W. Mitchell, **LISP 2 Specifications Proposal**, Stanford A. I. Memo AIM-21, August 1964.

[17] Anthony C. Hearn, **Standard LISP**, Stanford A. I. Memo AIM-90, May 1969.

[18] Lynn Quam, Whitfield Diffie, **Stanford LISP 1.6 Manual**, Stanford A. I. Lab. Operating note SAILON-28.7, 1973 (originally published in 1966).

[19] R. J. Bobrow, R. R. Burton, D. Lewis, **UCI LISP Manual**, U. C. Irvine Information and Computer Science Technical Report No. 21, October 1972.

[20] David C. Smith, **MLISP Users' Manual**, Stanford A. I. Memo AIM-84, January 1969.

[21] David C. Smith, **MLISP**, Stanford A. I. Memo AIM-135, October 1970.

[22] David C. Smith, Horace Enea, **MLISP2**, Stanford A. I. Memo AIM-195, May 1973.

[23] David C. Smith, Horace Enea, **Backtracking in MLISP2**, *Advance Papers, International Joint Conf. on Artificial Intelligence*, Stanford U., August 1973.

[24] L. G. Tesler, Horace Enea, David C. Smith, **The LISP70 Pattern Matching System**, *Advance Papers, International Joint Conf. on Artificial Intelligence*, Stanford U., August 1973.

2.5.2 FAIL

FAIL is a fast one-pass assembler for PDP-10 and PDP-6 machine language. It was developed initially by Phil Petit in 1968 [25] and the fourth revision of the manual will be published soon [26].

Compared with MACRO-10, the standard DEC assembler, FAIL uses substantially more memory, but assembles most programs in about one-fifth the time. FAIL assembles the entire Stanford A. I. timesharing system (two million characters) in less than 4 minutes of CPU time on a KA-10 processor.

FAIL provides more powerful macros than MACRO and has ALGOL-like block structure, permitting local names to be reused in various parts of a program without conflict.

Nearly all our system and utility programming is done in FAIL. It is distributed through DECUS and is widely used elsewhere.

Bibliography

[25] P. M. Petit, **FAIL**, Stanford A. I. Lab. Operating Note SAILON-26, 1968.

[26] F. H. G. Wright, **FAIL**, Stanford A. I. Memo, in preparation.

2.5.3 SAIL

Members of our staff began work on an ALGOL compiler for the PDP-6 in 1967. Work continued intermittently through December 1968, when Dan Swinehart put up GOGOL III [28].

Feldman subsequently guided the development of an extensively revised version that included his LEAP constructs [29, 30]. The resulting system, now called SAIL, has undergone a number of

subsequent revisions (e.g. [31]) and a new manual has been written [32]. SAIL is now a rather sophisticated programming system, offering coroutining, backtracking, context switching, and a number of other features.

The development of SAIL has been heavily influenced by the needs of our robotics research. SAIL is in use in a number of activities here and at many other installations on the ARPA network.

Bibliography

[27] John McCarthy (with 12 others), ALGOL 60, *Comm. ACM*, May 1960 and Jan. 1963; *Numerische Mathematik*, March 1960.

[28] Dan Swinehart, GOGOL III, Stanford A. I. Lab. Operating Note SAILON-48, December 1968.

[29] Jerome Feldman, Paul Rovner, The Leap Language Data Structure, *Proc. IFIP Congress*, 1968.

[30] Jerome Feldman, Paul Rovner, An ALGOL-based Associative Language, Stanford A. I. Memo AIM-66, August 1968; also in *Comm. ACM*, August 1969.

[31] Jerome Feldman, J. Low, D. C. Swinehart, R. H. Taylor, Recent Developments in SAIL, an ALGOL-based language for Artificial Intelligence, *Proc. FJCC*, 1972.

[32] Kurt VanLehn (ed.), SAIL User Manual, Stanford A. I. Memo AIM-204, July 1973.

2.6 Computer Facilities

In support of our research program, we have developed a very efficient display-oriented timesharing system. The basic hardware is a PDP-10/PDP-6 dual processor system with 256K words of core memory, a Librascope swapping disk, an IBM 3330 disk file with 85 million word capacity, and 64 display terminals.

The system is up essentially 24 hours per day, every day; it is not taken down for preventive maintenance. It comfortably supports forty-some jobs with a monitor that is functionally similar to DEC's TOPS-10, though quite different inside.

The high performance of the computer facility results in part from a number of technical innovations, both in hardware and software, and a great deal of hard work. We have consistently invested about one-third of our total funding in development of the computer system, and consider that ratio to be about right.

Equipment acquisition costs have been kept relatively low by not restricting choices to devices that are "compatible" with our existing hardware. Instead, we have selected on a performance/cost basis and usually done our own interfacing. The resulting total cost, including engineering and technician time, is often as low as one-third the price of "turn-key" devices, particularly for memories. Of course, the resulting system has a "mosaic" quality, with elements representing nearly every manufacturer. Nevertheless, it works rather well.

Our current system has the following features.

1. It successfully combines real time service (e.g. control of mechanical arms) with general timesharing [9, 13, 14]. We were the first to do this.

2. Everyone in our laboratory has a display terminal with full graphics capability in his office. This was made possible by their low average cost (about $2K per terminal). We employ a video switch to share a limited number of display generators among a larger number of displays [19]. This idea is being copied in a number of new display systems elsewhere.

3. Our display keyboard design uses multiple shift and control keys to provide both touch-typing of a large character set and powerful interactive control of editors and other system functions. This keyboard is now in use at MIT, Carnegie-Mellon University, and the University of Utah, and other groups are considering it.

4. Our teletype text editor, called SOS, was written some time ago (1968), but still is unsurpassed in its class [11]. It was recently adopted as a standard by the DECUS PDP-10 users group.

5. Our new display editor, called "E", is even better. Words can't describe it, but see [21, 22].

6. We have a geometric editor, called GEOMED, for drawing 3-D objects interactively and viewing them from any perspective [18].

7. Another family of drawing editors are used to design and check digital logic circuits and the corresponding printed circuit cards [23]. This family of programs has been adopted by MIT, CMU, and DEC.

8. We have developed a news information retrieval service called APE [30]. Our computer has a connection to an Associated Press newswire and maintains a file of storys from the last 24 hours. These storys are indexed under the words they contain, so that a person can ask for all the stories that mention, say, "Egypt" and have them displayed immediately. This service has proven to be very popular with people on the ARPA Network.

9. We have a document compiler called PUB that handles text justification with multiple variable-width fonts [20]. If asked, it will automatically handle section, page, and figure numbering, keep cross-references straight, generate a table of contents and a sorted index, and other wonderful things. Pub is in use at nearly all sites that have acquired Xerox Graphics Printers.

2.6.1 Early Development

The concept of a general purpose timesharing system was first proposed by John McCarthy when he was at MIT. That proposal led to the systems developed in Project MAC. McCarthy also participated in the development of an early timesharing system at BBN [1].

Shortly after arriving at Stanford in 1962, McCarthy undertook the development of a display-oriented timesharing system, with support from the National Science Foundation. That system had 12 display terminals connected to a PDP-1, with a link to an IBM 7090 [6]. An entertaining film report summarizes the capabilities of the system [7].

The PDP-1 timesharing system was used for some early hand-eye experiments with a rather crude television camera interface [3,4] and a donated prosthetic arm. The staff who developed the PDP-1 system became the nucleus of the A. I. Project computer group in 1966, which gave us a head start on the new system.

2.6.2 Hardware

Over ten years, total direct expense for computer and experimental equipment (not including salaries for construction or maintenance) was $2.4 million. Of this, about $1.6 million was for capital equipment purchases, $400K for interfacing and spare parts, $200K for IBM disk rental, and $200K for outside computer time.

Equipment acquisition highlights are as follows.

Date online		Equipment
1966	June	PDP-6, 64K core, 8 Dectape drives, 8 Teletypes
	Oct.	Vidicon camera, Rancho Arm
1967	Nov.	Librascope Disk
1968	Jan.	6 III Displays
	Aug.	Ampex Core (64K)
	Sept.	PDP-10 (KA-10 processor)
1969	Feb.	IBM 2314 Disk
1970	Jun.	3 IMLAC displays
1971	Mar.	Data Disc displays (58 eventually)
	May	Scheinman (Stanford) Arm
	July?	BBN IMP
1972	Jan.	IBM 3330 replaces 2314
	April	Ampex core (128K)
	May	Video Switch
1973	Jan.	Xerox Graphics Printer

Beginning in the summer of 1970, we also undertook the design of a new high speed processor, known locally as "Foonly", which was to be ten times as fast as our PDP-10. The design was completed in early 1973 and featured a 2K word cache memory, a microcode store that was to be accessible by timeshared users, and a "Console Computer", which was to be a minicomputer with display and keyboard. The Console Computer was to take the place of the traditional console lights and switches and, since it was to control the registers and clock of the big machine, could be used to monitor and debug the latter.

The processor was not fabricated, but some of the ideas in it may be expected to appear in commercial equipment. The digital logic design programs mentioned earlier were developed in support of this effort.

2.6.3 Software

Our first PDP-6 timesharing monitor was the DEC 1.3 Monitor. Since then, we have rewritten essentially all of the system, some parts more than once. We have mostly added features; a number of these have found their way back into later DEC monitors. Thus, we are semi-compatible with current TOPS-10 monitors.

An area of major revision in our system is keyboard and display service. We devised a "line editor" for displays, that facilitates rapid entry and modification of lines of text. Using the keyboards, we can also control various formatting parameters of the display and can even switch to television, to watch televised seminars on the Stanford campus. This service is a fringe benefit of using standard TV monitors for displays together with a video switch.

In support of the display service, we had to restructure the monitor to provide free storage allocation for display data. This free storage feature has proven useful for a number of other system functions.

We developed a display-oriented debugging package called RAID [16]. Its purpose is similar to DDT, but the display permits more powerful interaction, such as maintaining displays of selected locations.

When we received the PDP-10 in 1968, we devised a dual processor system that permits the PDP-10 to be used for both realtime and general timesharing service and the PDP-6 to be used for realtime only, such as control of arms and (recently) raster generation for the Xerox Graphics Printer.

We completely rewrote the disk file system in the Monitor, adding redundancy and read-before-write checking. Disk files are backed up by a tape archiving system that gives excellent protection against both operating and programming errors [24]. We have never had a major file loss attributable to system hardware or software failure since our file system was installed in 1969.

We have developed a resource reservation and allocation system, called RSL [29], that allows users to "purchase" reservations for subsystems and a *service level* (a percentage of CPU cycles). Reservation costs vary with the time of day and are stated in a reusable pseudo-currency, *bams*, whose abundance is controlled administratively. The reservation transactions, enforcement, and accounting are all performed automatically by programs. This permits persons who need assured services to have them without worrying about how many other people climb on the system.

An IMP was installed fairly early in the life of the ARPA Network and we wrote a telnet program to talk to it. At this point we discovered that the extra core memory needed to support this service caused serious system performance degradation. As a consequence, chose not to support network services until after we had a full 256K words of memory, in the spring of 1972 [25, 26].

Overall we think that our system has yielded excellent returns on the investment, though it suffers from chronic overloading. CPU time is the principal bottleneck.

Bibliography

Note that SAILONs listed below are *Stanford A. I. Lab. Operating Notes*, which are generally written for laboratory internal use. The text of most SAILONs is kept in disk files in our [S,DOC] area. *Working Notes* are even less formal than SAILONs and usually are kept only in the form of text files on the disk. Even so, these files are public and can be accessed over the ARPA Network [see Appendix A].

[1] John McCarthy, S. Boilen, E. Fredkin, J.C.R. Licklider, **A Time-Sharing Debugging System for a Small Computer**, *Proc. AFIPS Conf.* (SJCC), Vol. 23, 1963.

[2] John McCarthy, F. Corbato, M. Daggett, **The Linking Segment Subprogram Language and Linking Loader Programming Languages**, *Comm. ACM*, July 1963.

[3] P. Carah, **A Television Camera Interface for the PDP-1**, Stanford A. I. Memo AIM-34, June 1965.

[4] J. Painter, **Utilization of a TV Camera on the PDP-1**, Stanford A. I. Memo AIM-36, September 1965.

[5] John McCarthy, **Time-sharing Computer Systems**, in W. Orr (ed.), *Conversational Computers*, Wiley, 1966.

[6] John McCarthy, D. Brian, G. Feldman, J. Allen, **THOR -- A Display Based Time-sharing System**, *Proc. AFIPS Conf.* (FJCC), Vol. 30, Thompson, Washington D. C., 1967.

[7] Art Eisenson, Gary Feldman, **Ellis D. Kropotechev and Zeus, his Marvelous Time-sharing System**, 16mm film, black and white with sound, 15 minutes, March 1967.

[8] Richard Gruen, W. Weiher, **Rapid Program Generation**, *Proc. DECUS Symposium*, Fall 1968.

[9] J. A. Moorer, **Dual Processing for the PDP-6/10**, *Decuscope*, Vol 8, No. 3, 1969.

[10] William Weiher, **Loader Input Format**, SAILON-46, October 1968.

[11] William Weiher, S. Savitzky, **Son of Stopgap**, SAILON-50.3, also in disk file SOS.LES[S,DOC].

[12] William Weiher, R. Gruen, **RPG -- Rapid Program Generator**, SAILON-51, 1968.

[13] J. A. Moorer, **Stanford A-I Project Monitor Manual: Chapter I - Console Commands**, SAILON-54.2, September 1970 (revision in preparation).

[14] J. A. Moorer, **Stanford A-I Project Monitor Manual: Chapter II - User Programming**, SAILON-55.2, September 1970 (revision in preparation).

[15] Ted Panofsky, **Stanford A-I Facility Manual**, SAILON-56, May 1973.

[16] Phil Petit, **RAID**, SAILON-58.1, February 1970.

[17] Richard P. Helliwell, **Copy**, SAILON-61.1, May 1971.

[18] Bruce Baumgart, **GEOMED -- A Geometric Editor**, SAILON-68, May 1972.

[19] Lester Earnest, **Video Switch**, SAILON-69, May 1972.

[20] Larry Tesler, **FUB, the Document Compiler**, SAILON-70, September 1972.

[21] Dan Swinehart, **TV -- A Display Text Editor**, Working Note in disk file TVED.DCS[UP,DOC], December 1971.

[22] Fred Wright, **Differences between 'E' and 'TV'**, Working Note in disk file TV2E.FW[UP,DOC].

[23] Richard Helliwell, **Stanford Drawing Program**, Working Note in disk file W[F,RPH].

[24] Ralph Gorin, **DART -- Dump and Restore Technique**, Working Note in disk file DART.REG[UP,DOC], November 1972.

[25] Andy Moorer, **Telnet Program**, Working Note in disk file NET.JAM[UP,DOC].

[26] Dan Swinehart, **FTP -- File Transfer Protocol**, Working Note in disk file FTP.DCS[UP,DOC].

[27] Ralph Gorin, **Spooler System Documentation**, Working Note in disk file SPOOL.REG[UP,DOC]. February 1971.

[28] Ralph Gorin, **Spelling Checker**, Working Note in disk file SPELL.REG[UP,DOC].

[29] Jim Stein, **Automatic Service Level Reservation**, Working Note in disk file RSL.JHS[UP,DOC].

[30] Martin Frost, **Reading Associated Press News**, SAILON-72, July 1973.

2.7 Associated Projects

Our ARPA-sponsored research program has benefited from interaction with several associated but separately supported projects. In addition to intellectual interchange, there has been some resource sharing among the projects. For example, some computer equipment purchases and rentals have been shared on a *quid pro quo* basis.

2.7.1 Higher Mental Functions

The Higher Mental Functions Project, under the leadership of Kenneth Colby, is supported by the National Institutes of Health. The overall objectives are to aid in the solution of certain problems in psychiatry and psychotherapy.

One line of research involves a computer model of paranoia, called PARRY, which responds to natural language inquiries [6]. Another involves computer-based treatment of language difficulties in nonspeaking children [9]. (For this research, the principal investigator received the 1973 Frieda Fromm-Reichmann Award from the American Academy of Psychoanalysis.)

Bibliography

[1] Colby, K. M., and Smith, D. C., **Dialogues between Humans and Artificial Belief Systems**, Stanford A. I. Memo AIM-97, August 1969; also in *Proc. International Joint Conference on Artificial Intelligence*, 1969.

[2] Colby, K. M., Tesler, L., and Enea, H. J., **Experiments with a Search Algorithm on the Data Base of a Human Belief Structure**, Stanford A. I. Memo AIM-94, August 1969.

[3] Colby, K. M., Hilf, F. D., and Hall, W. A., **A Mute Patient's Experience with Machine-Mediated Interviewing**, Stanford A. I. Memo AIM-113, March 1970.

[4] Colby, K. M., **Mind and Brain, Again**, Stanford A. I. Memo AIM-116, March, 1970; also in *Mathematical Biosciences* Vol. 11, 47-52, 1970.

[5] Colby, K. M., and Smith, D. C., **Computer as Catalyst in the Treatment of Nonspeaking Autistic Children**, Stanford A. I. Memo AIM-120, April 1970.

[6] Colby, K. M., Weber, S., and Hilf, F. D., **Artificial Paranoia**, Stanford A. I. Memo AIM-125, July 1970; also in *Artificial Intelligence Journal* Vol. 2, No. 1, 1972.

[7] Colby, K. M., Hilf, F. D., Weber, S., and Kraemer, H. C., **A Resemblance Test for the Validation of a Computer Simulation of Paranoid Processes**, Stanford A. I. Memo AIM-156, November 1971.

[8] Colby, K. M., Hilf, F. D., Weber, S., and Kraemer, H. C., **Turing-like Indistinguishability Tests for the Validation of a Computer Simulation of Paranoid Processes**, *Artificial Intelligence Journal* Vol. 3, No. 1, Fall 1972.

[9] Colby, K. M., **The Rationale for Computer-based Treatment of Language Difficulties in Nonspeaking Autistic Children**, Stanford A. I. Memo AIM-193, April 1973; also in *Journal of Autism and Childhood Schizophrenia*, Vol. 3, 254-260, 1973.

[10] Colby, K. M. and Hilf, F. D., **Multidimensional Analysis in Evaluating a Simulation of Paranoid Thought**, Stanford A. I. Memo AIM-194, May, 1973.

[11] Enea, H. J., and Colby, K. M., **Idiolectic Language-Analysis for Understanding Doctor-Patient Dialogues**, *Advance Papers of the Third International Joint Conference on Artificial Intelligence*, 278-284, 1973.

[12] Hilf, F. D., Colby, K. M., Smith, D. C., Wittner, W., and Hall, W. A., **Machine-Mediated Interviewing**, *Journal of Nervous and Mental Disease*, Vol. 152, No. 4, 1971.

[13] Hilf, F. D., **Non-Nonverbal Communication and Psychiatric Research**, *Archives of General Psychiatry*, Vol. 27, November 1972.

[14] Hilf, F. D., **Partially Automated Psychiatric Interviewing -- A Research Tool**, *Journal of Nervous and Mental Disease*, Vol. 155, No. 6, December 1972.

[15] Schank, R. C., **The Fourteen Primitive Actions and Their Inferences**, Stanford A. I. Memo AIM-183, February 1973.

[16] Schank, R. C., and Rieger, C. J. III, **Inference and Computer Understanding of Natural Language**, Stanford A. I. Memo AIM-197, May 1973.

[17] Schank, R. C., and Wilks, Y., **The Goals of Linguistic Theory Revisited**, Stanford A. I. Memo AIM-202, May 1973.

[18] Schank, R. C., **The Development of Conceptual Structures in Children**, Stanford A. I. Memo AIM-203, May, 1973.

[19] Schank, R. C., Goldman, N., Rieger, C. J. III, and Riesbeck, C., **Margie: Memory, Analysis, Response Generation and Inference on English**, *Advance Papers of the Third International Joint Conference on Artificial Intelligence*, 255-261, August 1973.

[20] Schank, R. C., **Identification of Conceptualizations Underlying Natural Language**, in R. Schank and K. Colby (eds.), *Computer Models of Thought and Language*, W. H. Freeman, San Francisco, 1973.

[21] Schank, R. C., and Colby, K. M., (eds.), *Computer Models of Thought and Language*, W. H. Freeman, San Francisco, 1973.

[22] Smith, D. C., and Enea, H. J., **MLISP2**, Stanford A. I. Memo AIM-195, May 1973.

[23] Smith, D. C., and Enea. H. J., **Backtracking in MLISP2**, *Advance Papers of the Third International Joint Conference on Artificial Intelligence*, 677-685, August 1973.

[24] Tesler, L., Enea, H. J., and Colby, K. M., **A Directed Graph Representation for Computer Simulation of Belief Systems**, *Mathematical Biosciences*, Vol. 2, 1968.

[25] Tesler, L. G., Enea, H. J. and Smith, D. C., **The LISP70 Pattern Matching System**, *Advance Papers of the Third International Joint Conference on Artificial Intelligence*, 671-676, August 1973.

2.7.2 Digital Holography

In a project lead by Joseph Goodman of the Electrical Engineering Department and sponsored by the Air Force, certain techniques for converting holograms into photographic images were investigated, with potential applications to satellite photography.

One interesting byproduct of this work was the creation of the world's first photograph taken without lens or pinhole [2]. A

hologram was formed directly on the surface of a vidicon television tube, which was connected to our computer through a digitizer. The digitized hologram was then converted into an image of the original object by computer methods and displayed on a CRT.

Bibliography

[1] Joseph Goodman, **Digital Image Formation from Electronically Detected Holograms**, in *Proc. SPIE Seminar on Digital Imaging Techniques*, Soc. Photo-Optical Instrumentation Engineering, Redondo Beach, California, 1967.

[2] Joseph Goodman, **Digital Image Formation from Electronically Detected Holograms**, *Applied Physics Letters*, August 1967.

[3] A. Silvestri and J. Goodman, **Digital Reconstruction of Holographic Images**, 1968, NEREM Record, IEEE, Vol. 10, pp. 118-119. 1968.

2.7.3 Sound Synthesis

John Chowning and Leland Smith of the Stanford Music Department and their students have developed computer techniques for generating stereophonic and quadraphonic sounds that can be programmed to move in two and three dimensions with respect to the listener. The method controls the distribution and amplitude of direct and reverberant signals between loudspeakers to provide the angular and distance information and introduces a Doppler shift to enhance velocity information [3].

Recently, Chowning made an interesting discovery that frequency modulation techniques provide a simple but effective way to synthesize certain kinds of sounds [6].

Leland Smith has developed a graphics editor capable of handling musical notation, among other things [7]. A number of commercial groups had previously tried and failed to solve this problem.

Bibliography

[1] James Beauchamp (with H. Von Foerster) (eds.), *Music by Computers*, John Wiley, New York, 1969.

[2] John M. Chowning, **The Simulation of Moving Sound Sources**, *Proc. Audio Engineering Soc. Convention*, May 1970.

[3] James A. Moorer, **Music and Computer Composition**, *Comm. ACM*, January 1972.

[4] Leland Smith, **SCORE -- A Musician's Approach to Computer Music**, *J. Audio Eng. Soc.*, Jan./Feb. 1972.

[5] James A. Moorer, **The Hetrodyne Method of Analysis of Transient Waveforms**, Stanford A. I. Memo AIM-208, June 1973.

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

[7] Leland Smith, **Editing and Printing Music by Computer**, *J. Music Theory*, Fall 1973.

2.7.4 Mars Picture Processing

As in so many areas, John McCarthy was among the first to examine potential applications of artificial intelligence to planetary exploration [1].

More recently, under the sponsorship of the National Aeronautics and Space Administration, Lynn Quam did a dissertation on picture differencing techniques [2]. The particular set of pictures he worked on was satellite photographs of Mars containing various geometric and photometric distortions as well as several kinds of noise. He successfully solved the problem of detecting small changes in the planet surface in the presence of all these extraneous factors.

His system was subsequently applied to pictures of Mars taken by the Mariner 9 spacecraft while the mission was in progress [3, 4, 5]. A short film shows the interactive display techniques [6].

Bibliography

[1] John McCarthy, **Computer Control of a Machine for Exploring Mars**, Stanford A. I. Memo AIM-14, January 1964.

[2] Lynn H. Quam, **Computer Comparison of Pictures**, Stanford A. I. Memo AIM-144, May 1971.

[3] Lynn H. Quam, et al, **Computer Interactive Picture Processing**, Stanford A. I. Memo AIM-166, April 1972.

[4] Carl Sagan, et al, **Variable Features on Mars: Preliminary Mariner 9 Television Results**, *Icarus* 17, pp. 346-372, 1972.

[5] Lynn H. Quam, et al, **Mariner 9 Picture Differencing at Stanford**, *Sky and Telescope*, August 1973.

[6] Larry Ward, **Computer Interactive Picture Procession**, 16 mm color film with sound, 8 min., 1972.

3. HEURISTIC PROGRAMMING PROJECT

The Heuristic Programming Project originated in 1965 under the name Heuristic DENDRAL. Its current title reflects a broadening of scope to include several areas of research related by the common theme of developing high-performance, intelligent programs for assisting scientific work.

3.1 Summary of Aims and Accomplishments

Heuristic DENDRAL is one of the few examples of a high-performance intelligent system, sometimes achieving levels of scientific problem solving not yet achieved by the best human experts, often achieving levels equal to that of good human performance. However, the attention given to the Heuristic DENDRAL performance program as a successful application of artificial intelligence research has tended to obscure the more general concerns of the project investigators. Our aims and accomplishments have been:

1. To study and construct detailed information processing models of processes of scientific inference. By scientific inference we mean the inferential process by which a model is constructed to explain a given set of empirical data. The Heuristic DENDRAL system is such a model.

2. To study experimentally the "operating characteristics" and the effectiveness of different designs (strategies) for the development of task-specific knowledge in a scientific area. The Planning Rule Generator, a program which takes the theory used in hypothesis verification and produces rules for guiding hypothesis generation, is a result of this concern [23]. The General Planning Program is another example [28].

3. To develop a method for eliciting from an expert the heuristics of scientific judgment and choice that he is using in the performance of a complex inference task. We have designed our problem solving systems so that the heuristics may be separated from the programs which use them. By restricting program design to this table-driven form [16] new heuristics can be easily incorporated. Heuristic DENDRAL, Meta-DENDRAL, and the General Planning Program employ this methodology.

4. To solve real problems in an area of significance to modern science, and to do so with a level of performance high enough to have a noticeable impact upon that area of science. Chemists will agree we have reached that stage. For example, the General Planning Program has been used to analyze mixtures of estrogenic steroids without the need for gas-chromatographic separation [32]. In the analysis of data for some classes of compounds Heuristic DENDRAL's performance matches or exceeds that of a post-doctoral chemist.

5. To discover the heuristics that form the basis of expert performance. The significant problem is not so much tuning a specialist with new sets of heuristics for new problems as learning how to acquire these heuristics. The problem of knowledge acquisition and structuring by problem solving systems is crucial, and is perhaps the central problem of AI research today. In recent years we have made it the main concern of our project. The work on automatic theory formation is focussed on the development of a program called Meta-DENDRAL for automatic acquisition of a theory of fragmentation processes in mass spectrometry [33, 34].

6. To study the representation of knowledge. Much of Computer Science can be viewed as a series of programming innovations by means of which we are moving gradually from a position of having to tell a computer

precisely how we want a problem to be solved to a position of being able to tell a problem-solving program what we wish done by the computer. But, what the user wants done always concerns some specific task environment--some piece of the real world. For the problem-solving program to interpret for itself the what, it must have knowledge of the specific task environment and its behavior. In other words, the program needs some kind of theory (formal, informal, heuristic, etc.) of that environment in terms of which to do its problem solving. We have seen in our work that the form in which knowledge about the (DENDRAL) world is represented is crucial to effective problem solving and to augmenting the knowledge structure for improved performance. We have found the production rule form of knowledge representation to be flexible, easily understood, manipulable by a computer program, and capable of driving a complex problem solving system [16, 17, 25, 26].

Survey Articles

The research leading to the implementation of the Heuristic DENDRAL and Meta-DENDRAL systems has been documented in over thirty publications; a bibliography is included at the end of this section. In particular, [17] and [25] give fairly concise summaries of the Heuristic DENDRAL research up through 1970, and reference 29 reports on the status of Meta-DENDRAL as of the middle of 1972.

Most Recent Accomplishments

Since 1970, the most significant accomplishments of the DENDRAL research effort have been the design and implementation of

1) an exhaustive and irredundant generator of topological graphs, thereby extending Heuristic DENDRAL to cyclic structures [30],
2) a General Planning Program which interprets high resolution mass spectral data from complex, biologically interesting molecules having a known skeletal substructure [28, 32],
3) the initial parts of a theory formation program which has already been used to infer a theory of mass spectrometry for a particular class of molecules [33].

3.2 Current Activity

At the present time the Project members are working in the following task areas:

1) Heuristic DENDRAL - Extensions to the Heuristic DENDRAL program are aimed at increasing its utility to practicing chemists by extending its domain of applicability to a wider class of molecules. This work is now funded by the Biotechnology Resources Branch of the National Institutes of Health (Grant No. RR-612-01).

2) Meta-DENDRAL - Within the domain of mass spectrometry, a theory formation program is under development. The goal is to assist in the inference of the theory of fragmentation of molecules in a mass spectrometer.

3) Intelligent Control of Scientific Instruments - The aim of this research is to develop programs for intelligent control of a data-collecting instrument. The program makes control decisions by reasoning in terms of a theory of the instrumental and physical process involved. This is a problem of importance when time, band width, or other constraints limit the amount of data which can be gathered for analysis. Candidate instruments are mass spectrometers and nuclear magnetic resonance spectrometers.

4) Application of AI to the task of computer programming (Automatic Programming) -

One of the aspects of programming being worked upon is debugging. In a DENDRAL-like fashion, evidence of program malfunction (bugs) is "explained" in terms of a model of commonly-observed program pathologies. In another effort, the synthesis of portions of systems programs is the subject of a Ph.D. thesis project now being formulated.

5) Application of AI to a new complex task domain of scientific interest, viz. protein structure determination from x-ray crystallographic data - Work has recently begun to develop heuristic programs that will formulate 3-space models of the structure of proteins. The expertise of protein chemists in "fitting" complex structures to poorly resolved and/or incomplete data will be extracted from experts and used by the program.

3.3 Views Expressed by Others Concerning DENDRAL

The DENDRAL publications have engendered considerable discussion and comment among computer scientists and chemists. Professor H Gelernter (SUNY, Stony Brook), at an SJCC 1970 panel of the use of computers in science gave an extended discussion of the program, in which he said that it was the first important scientific application of artificial intelligence. Dr. W. H. Ware (RAND Corporation) in a recent article entitled "The Computer in Your Future" in the collection *Science and Technology in the World of the Future* (A. B. Bronwell, ed., Wiley Press, 1970) said:

> "Thus, much of engineering will be routinized at a high level of sophistication, but what about science? An indication of what is coming at a higher level of intellectual performance is a computer program called Heuristic DENDRAL, which does a task that a physical chemist or biologist concerned with organic chemistry does repeatedly."

Professor J. Weizenbaum of MIT, in an undergraduate computer science curriculum proposal for MIT entitled "A First Draft of a Proposal for a New Introductory Subject in Computer Science (September 1970)", included Heuristic DENDRAL in his "group 4" series (three lectures) entitled Great Programs; and he said recently (personal communication), commenting on recent work and plans,

> "I see the work you are now beginning as a step in the direction of composing explicit models of just such programs (that build expertise in an area). The implications of a success in such an effort are staggering. I therefore believe your effort to be worthy of very considerable investment of human and financial resources."

In his paper presented at the Sixth International Machine Intelligence Workshop, Professor Saul Amarel (Rutgers University) used Heuristic DENDRAL to illustrate a point about programs which use theoretical knowledge. He wrote:

> "The DENDRAL system provides an excellent vehicle for the study of uses of relevant theoretical knowledge in the context for formation problems," from "Representations and Modeling in Problems of Program Formation", Saul Amarel, in *Machine Intelligence 6* (B. Meltzer and D. Michie, eds.) Edinburgh University Press (in press).

Professor Donald Michie of the University of Edinburgh includes a description of Heuristic DENDRAL in his recent paper on "Machines and the Theory of Intelligence" (*Nature*, 241, pp. 507-512, February 23, 1973):

> "Mass spectrogram analysis was proposed by Lederberg as a suitable task for machine intelligence methods. The heuristic DENDRAL program

> developed by him and Feigenbaum now outperforms post-doctoral chemists in the identification of certain classes of organic compounds. The program is a rich quarrying ground for fundamental mechanisms of intelligence, including the systematic conjecture of hypotheses, heuristic search, rote learning, and deductive and inductive reasoning."

And, in the March, 1973, issue of *Computing Reviews* (pp. 132-133), Robert Kling of the University of Wisconsin begins his review of reference 25 with this assessment of DENDRAL:

> "This brief paper provides an overview of one of the most sophisticated applications programs in artificial intelligence"

Dr. T. G. Evans (Air Force Cambridge Research Labs), President of the ACM SIGART, in introducing a talk on Heuristic DENDRAL at the 1970 FJCC meeting of SIGART, called the program "probably the smartest program in the world" (and followed this with the interesting observation that this program had probably received a more sustained and intense effort than any other single program in the history of the artificial intelligence field).

At a practical level, a mass spectrometry laboratory at the University of Copenhagen, headed by Dr. Gustav Schroll, adapted the program to his facilities there.

Bibliography

[1] J. Lederberg, **DENDRAL-64 - A System for Computer Construction, Enumeration and Notation of Organic Molecules as Tree Structures and Cyclic Graphs**, (technical reports to NASA, also available from the author and summarized in [12]).

(1a) Part I. Notational algorithm for tree structures (1964) CR.57029.

(1b) Part II. Topology of cyclic graphs (1965) CR.68898.

(1c) Part III. Complete chemical graphs; embedding rings in trees (1969).

[2] J. Lederberg, *Computation of Molecular Formulas for Mass Spectrometry*, Holden-Day, Inc. (1964).

[3] J. Lederberg, **Topological Mapping of Organic Molecules**, *Proc. Nat. Acad. Sci.*, 53:1, January 1965, pp. 134-139.

[4] J. Lederberg, **Systematics of organic molecules, graph topology and Hamilton circuits. A general outline of the DENDRAL system.** NASA CR-48899 (1965).

[5] J. Lederberg, **Hamilton Circuits of Convex Trivalent Polyhedra (up to 18 vertices)**, *Am. Math. Monthly*, May 1967.

[6] G. L. Sutherland, **DENDRAL - A Computer Program for Generating and Filtering Chemical Structures**, Stanford Artificial Intelligence Memo AIM-49, February 1967.

[7] J. Lederberg and E. A. Feigenbaum, **Mechanization of Inductive Inference in Organic Chemistry**, in B. Kleinmuntz (ed) *Formal Representations for Human Judgment*, Wiley, 1968; also Stanford Artificial Intelligence Memo AIM-54, August 1967.

[8] J. Lederberg, **Online Computation of Molecular Formulas from Mass Number** NASA CR-94977, 1968.

[9] E. A. Feigenbaum and B. G. Buchanan, **Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry**, in *Proceedings, Hawaii International Conference on System Sciences*, B. K. Kinariwala and F. F. Kuo (eds), University of Hawaii Press, 1968.

[10] B. G. Buchanan, G. L. Sutherland, and E. A. Feigenbaum, **Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry**, in *Machine Intelligence 4*, B. Meltzer and D. Michie (eds), Edinburgh University Press, 1969; also Stanford Artificial Intelligence Memo AIM-62, July 1968.

[11] E. A. Feigenbaum, **Artificial Intelligence: Themes in the Second Decade**. In *Final Supplement to Proceedings of the IFIP68 International Congress*, Edinburgh, August 1968; also Stanford Artificial Intelligence Memo AIM-67, August 1968.

[12] J. Lederberg, **Topology of Molecules**, in *The Mathematical Sciences - A Collection of Essays*, Committee on Support of Research in the Mathematical Sciences (COSRIMS), National Academy of Sciences - National Research Council, M.I.T. Press, 1969, pp. 37-51.

[13] G. Sutherland, **Heuristic DENDRAL: A Family of LISP Programs**, to appear in D. Bobrow (ed), *LISP Applications*; also Stanford Artificial Intelligence Memo AIM-80, March 1969.

[14] J. Lederberg, G. L. Sutherland, B. G. Buchanan, E. A. Feigenbaum, A. V. Robertson, A. M. Duffield, and C. Djerassi, **Applications of Artificial Intelligence for Chemical Inference I. The Number of Possible Organic Compounds: Acyclic Structures Containing C, H, O and N**. *Journal of the American Chemical Society*, 91:11, May 21, 1969.

[15] A. M. Duffield, A. V. Robertson, C. Djerassi, B. G. Buchanan, G. L. Sutherland, E. A. Feigenbaum, and J. Lederberg, **Application of Artificial Intelligence for Chemical Inference II. Interpretation of Low Resolution Mass Spectra of Ketones**. *Journal of the American Chemical Society*, 91:11, May 21, 1969.

[16] B. G. Buchanan, G. L. Sutherland, E. A. Feigenbaum, **Toward an Understanding of Information Processes of Scientific Inference in the Context of Organic Chemistry**, in *Machine Intelligence 5*, B. Meltzer and D. Michie, (eds), Edinburgh University Press 1970; also Stanford Artificial Intelligence Memo AIM-99, September 1969.

[17] J. Lederberg, G. L. Sutherland, B. G. Buchanan, and E. A. Feigenbaum, **A Heuristic Program for Solving a Scientific Inference Problem: Summary of Motivation and Implementation**, Stanford Artificial Intelligence Memo AIM-104, November 1969.

[18] C. W. Churchman and B. G. Buchanan, **On the Design of Inductive Systems: Some Philosophical Problems**, *British Journal for the Philosophy of Science*, 20, 1969, pp. 311-323.

[19] G. Schroll, A. M. Duffield, C. Djerassi, B. G. Buchanan, G. L. Sutherland, E. A. Feigenbaum, and J. Lederberg, **Application of Artificial Intelligence for Chemical Inference III. Aliphatic Ethers Diagnosed by Their Low Resolution Mass Spectra and NMR Data**, *Journal of the American Chemical Society*, 91:26, December 17, 1969.

[20] A. Buchs, A. M. Duffield, G. Schroll, C. Djerassi, A. B. Delfino, B. G. Buchanan, G. L. Sutherland, E. A. Feigenbaum, and J. Lederberg, **Applications of Artificial Intelligence For Chemical Inference. IV. Saturated Amines Diagnosed by Their Low Resolution Mass Spectra and Nuclear Magnetic Resonance Spectra**, *Journal of the American Chemical Society*, 92, 6831, 1970.

[21] Y.M. Sheikh, A. Buchs, A.B. Delfino, G. Schroll, A.M. Duffield, C. Djerassi, B.G. Buchanan, G.L. Sutherland, E.A. Feigenbaum and J. Lederberg, **Applications of Artificial Intelligence for Chemical Inference V. An Approach to the Computer Generation of Cyclic Structures. Differentiation Between All the Possible Isomeric Ketones of Composition C6H10O,** *Organic Mass Spectrometry*, 4, 493, 1970.

[22] A. Buchs, A.B. Delfino, A.M. Duffield, C. Djerassi, B.G. Buchanan, E.A. Feigenbaum and J. Lederberg, **Applications of Artificial Intelligence for Chemical Inference VI. Approach to a General Method of Interpreting Low Resolution Mass Spectra with a Computer,** *Chem. Acta Helvetica*, 53, 1394, 1970.

[23] E.A. Feigenbaum, B.G. Buchanan, and J. Lederberg, **On Generality and Problem Solving: A Case Study Using the DENDRAL Program,** in *Machine Intelligence 6*, B. Meltzer and D. Michie, (eds.), Edinburgh University Press, 1971; also Stanford Artificial Intelligence Memo AIM-131, August 1970.

[24] A. Buchs, A.B. Delfino, C. Djerassi, A.M. Duffield, B.G. Buchanan, E.A. Feigenbaum, J. Lederberg, G. Schroll, and G.L. Sutherland, **The Application of Artificial Intelligence in the Interpretation of Low-Resolution Mass Spectra,** *Advances in Mass Spectrometry*, 5, 314.

[25] B.G. Buchanan and J. Lederberg, **The Heuristic DENDRAL Program for Explaining Empirical Data,** *Proc. IFIP Congress 71*; also Stanford Artificial Intelligence Memo AIM-141.

[26] B.G. Buchanan, E.A. Feigenbaum, and J. Lederberg, **A Heuristic Programming Study of Theory Formation in Science,** *Advance Papers of the Second International Joint Conference on Artificial Intelligence*, Imperial College, London, September, 1971; also Stanford Artificial Intelligence Memo AIM-145.

[27] Buchanan, B. G., Duffield, A.M., Robertson, A.V., **An Application of Artificial Intelligence to the Interpretation of Mass Spectra,** *Mass Spectrometry Techniques and Appliances*, George W. A. Milne (ed), John Wiley & Sons, 1971, p. 121-77.

[28] 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 Estrogenic Steroids,** *Journal of the American Chemical Society*, 94, 5962-5971, 1972.

[29] B.G. Buchanan, E.A. Feigenbaum, and N.S. Sridharan, **Heuristic Theory Formation: Data Interpretation and Rule Formation,** in *Machine Intelligence 7*, Edinburgh University Press, 1972.

[30] Brown, H., Masinter L., Hjelmeland, L., **Constructive Graph Labeling Using Double Cosets,** *Discrete Mathematics* (in press); also Stanford Computer Science Memo 318, 1972.

[31] B. G. Buchanan, Review of Hubert Dreyfus' **What Computers Can't Do: A Critique of Artificial Reason,** *Computing Reviews*, January 1973; also Stanford Artificial Intelligence Memo AIM-181, November 1972.

[32] 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. *Journal of the American Chemical Society* (in press).

[33] 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 Estrogenic Steroids. *Tetrahedron*, (in press).

[34] B. G. Buchanan and N. S. Sridharan, Rule Formation on Non-Homogeneous Classes of Objects, *Advance Papers of the Third International Joint Conference on Artificial Intelligence*, Stanford, California, August, 1973.

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 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 all 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 CS 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. Lab. 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 Lab.
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 author(s) at the address given above for Operating Notes.

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 a few of our A. I. Memos,
[S,DOC] many of our SAILONs,
[UP,DOC] working notes (quite informal),
[P,DOC] "people-oriented" files, including the lab phone directory.

Network Access

To get into our system from the Network, say "L NET.GUE", which logs you in as a "network guest". All commands end with a carriage return. Our monitor types "." whenever it is ready to accept a command. To halt a program that is running, type <Control>C twice.

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 see the names of all the files in, say, the [S,DOC] area (where SAILONs are stored), type "DIR [S,DOC]". This will produce a list of files and the dates they were last written. Among others, it should list "INTRO.TES", which is an introduction to our timesharing system (usually obsolescent, alas), written by the programmer whose initials are TES.

To type out the contents of a given file, such as INTRO.TES, say

```
TYPE INTRO.TES[S,DOC]
```

and it will come spewing forth. To stop the typout, say <Control>C twice and it will stop after a few lines. To continue, type "CONT". If you wish to examine selected portions of text files, use the SOS editor in read-only mode, as described in SOS.LES[S,DOC].

To log out when you are done, type

```
K <carriage return>
```

There may be some difficulty with files that employ the full Stanford character set, which uses some 26 of the ASCII control codes (0 to 37 octal) to represent special characters.

File Transfer

Files can also be transferred to another site using the File Transfer Protocol. Documentation on our FTP program is located on our disk file in 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Robert E. Kling, AIM-147
Reasoning by Analogy with Applications to Heuristic Problem Solving: a Case Study
Ph.D. Thesis in Computer Science,
August 1971.

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

Jonathan Leonard Ryder, AIM-155
Heuristic Analysis of Large Trees as Generated in the Game of Go,
Ph.D. Thesis in Computer Science,
December 1971.

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

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

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

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

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

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

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

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

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

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 $30 to $50) from:

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

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-mechanical arm solves simple tasks involving stacking blocks. The techniques of recognizing the blocks and their positions as well as controlling the arm are briefly presented. 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 breathless 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. Winograd, Academic Press, 1972). No explanatory or narrative material is on the film.

Appendix D

EXTERNAL PUBLICATIONS

Articles and books by Project members are listed here alphabetically by lead author. Only publications following the individual's affiliation with the Project are given.

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

2. Allen, John, David Luckham, **An Interactive Theorem-Proving Program** in Bernard Meltzer and Donald Michie (eds.), *Machine Intelligence 5*, Edinburgh University Press, 1970.

3. Ashcroft, Edward, Zohar Manna, **Formalization of Properties of Parallel Programs**, *Machine Intelligence 6*, Edinburgh Univ. Press, 1971.

4. Ashcroft, Edward, Zohar Manna, **The Translation of 'Go To' Programs to 'While' Programs**, *Proc. IFIP Congress 1971*.

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

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

7. Beauchamp, James, H. Von Foerster (eds.), *Music by Computers*, John Wiley, New York, 1969.

8. Becker, Joseph, **The Modeling of Simple Analogic and Inductive Processes in a Semantic Memory System**, *Proc. International Conf. on Artificial Intelligence*, Washington, D.C., 1969.

9. Biermann, Alan, Jerome Feldman, **On the Synthesis of Finite-state Machines from Samples of Their Behavior**, *IEEE Transactions on Computers*, Vol. C-21, No. 6, pp. 592-596, June 1972.

10. Biermann, Alan, **On the Inference of Turing Machines from Sample Computations**, *Artificial Intelligence J.*, Vol. 3, No. 3, Fall 1972.

11. Binford, Thomas O., **Sensor Systems for Manipulation**, in E. Heer (Ed.), *Remotely Manned Systems*, Calif. Inst. of Technology, 1973.

12. Binford, Thomas, Jay M. Tenenbaum, **Computer Vision**, *Computer (IEEE)*, May 1973.

13. Bracci, Giampio, Marco Somalvico, **An Interactive Software System for Computer-aided Design: An Application to Circuit Project**, *Comm. ACM*, September 1970.

14. Buchanan, Bruce, Georgia Sutherland, **Heuristic Dendral: A Program for Generating Hypotheses in Organic Chemistry**, in Donald Michie (ed.), *Machine Intelligence 4*, American Elsevier, New York, 1969.

15. Buchanan, Bruce, Georgia Sutherland, Edward Feigenbaum, **Rediscovering some Problems of Artificial Intelligence in the Context of Organic Chemistry**, in Bernard Meltzer and Donald Michie (eds), *Machine Intelligence 5*, Edinburgh University Press, 1970.

16. Buchanan, Bruce, T. Headrick, **Some Speculation about Artificial Intelligence and Legal Reasoning**, *Stanford Law Review*, November 1970.

17. Buchanan, Bruce, A. M. Duffield, A. V. Robertson, **An Application of Artificial Intelligence to the Interpretation of Mass Spectra**, in *Mass Spectrometry Techniques and Appliances*, George W. Milne (ed), John Wiley & Sons, 1971.

18. Buchanan, Bruce, Edward Feigenbaum, Joshua Lederberg, **A Heuristic Programming Study of Theory Formation in Science**, *Proc. Second International Joint Conference on Arificial Intelligence (2IJCAI)*, British Computer Society, Sept. 1971.

19. Buchanan, Bruce, Joshua Lederberg, **The Heuristic DENDRAL Program for Explaining Empirical Data**, *Proc. IFIP Congress 1971*.

20. Buchanan, Bruce, E. A. Feigenbaum, and N. S. Sridharan, **Heuristic Theory Formation: Data Interpretation and Rule Formation**, in *Machine Intelligence 7*, Edinburgh University Press, 1972.

21. Buchanan, Bruce G., **Review of Hubert Dreyfus' 'What Computers Can't Do': A Critique of Artificial Reason**, *Computing Reviews*, January 1973.

22. 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.

23. Buchs, A., A. Delfino, A. Duffield, C. Djerassi, B. Buchanan, E. Feigenbaum, J. Lederberg, **Applications of Artificial Intelligence for Chemical Inference VI. Approach to a General Method of Interpreting Low Resolution Mass Spectra with a Computer**, Helvetica Chemica Acta, 53:6, 1970.

24. Buchs, A., A. Duffield, G. Schroll, Carl Djerassi, A. Delfino, Bruce Buchanan, Georgia Sutherland, Edward Feigenbaum, Joshua Lederberg, **Applications of Artificial Intelligence for Chemical Inference IV. Saturated Amines Diagnosed by their Low Resolution Mass Spectra and Nuclear Magnetic Resonance Spectra**, *J. Amer. Chem. Soc.*, 92:23, November 1970.

25. Cadiou, Jean M., Zohar Manna, **Recursive Definitions of Partial Functions and their Computations**, *ACM SIGPLAN Notices*, Vol. 7, No. 1, January 1972.

26. Campbell, John, **Algebraic Computation of Radiative Corrections for Electron-Proton Scattering**, *Nuclear Physics*, Vol. B1, pp. 238-300, 1967.

27. Campbell, Anthony Hearn and J. A., **Symbolic Analysis of Feynman Diagrams by Computer**, *Journal of Computational Physics* 5, 280-327, 1970.

28. Chowning, John M., **The Simulation of Moving Sound Sources**, *Proc. Audio Engineering Soc. Convention*, May 1970.

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

30. Churchman, C., Bruce Buchanan, **On the Design of Inductive Systems: Some Philosophical Problems**, *British Journal for the Philosophy of Science*, 20, 1969, pp. 311-323.

31. Colby, Kenneth, David Smith, **Dialogues between Humans and Artificial Belief Systems**, *Proc. International Conference on Artificial Intelligence*, Washington, D.C., 1969.

32. Colby, Kenneth, Larry Tesler, Horace Enea, **Experiments with a Search Algorithm for the Data Base of a Human Belief System**, *Proc. International Conference on Artificial Intelligence*, Washington, D.C., 1969.

33. Colby, Kenneth Mark, **The Rationale for Computer-Based Treatment of Language Difficulties in Nonspeaking Autistic Children**, *Journal of Autism and Childhood Schizophrenia*, Vol. 3, 254-260, 1970.

34. Colby, Kenneth, **Mind and Brain Again**, *Mathematical Biosciences*, Vol. 11, 47-52, 1970.

35. Colby, Kenneth, Sylvia Weber, Franklin Hilf, **Artificial Paranoia**, *J. Art. Int.*, Vol. 2, No. 1, 1971.

36. Colby, Kenneth, F. Hilf, S. Weber, H. C. Kraemer, **Turing-like Indistinguishability Tests for the Validation of a Computer Simulation of Paranoid Processes**, *Artificial Intelligence J.*, Vol. 3, No. 3, Fall 1972.

37. Colby, Kenneth M., **The Rationale for Computer-based Treatment of Language Difficulties in Nonspeaking Autistic Children**, *Journal of Autism and Childhood Schizophrenia*, Vol. 3, 254-260, 1973.

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

39. Duffield, Alan, A. Robertson, Carl Djerassi, Bruce Buchanan, G. Sutherland, Edward Feigenbaum, Joshua Lederberg, **Application of Artificial Intelligence for Chemical Interference II. Interpretation of Low Resolution Mass Spectra of Ketones**, *J. Amer. Chem. Soc.*, 91:11, May 1969.

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

41. Falk, Gilbert, **Scene Analysis Based on Imperfect Edge Data**, *Proc. 2IJCAI*, Brit. Comp. Soc., Sept. 1971.

42. Falk, Gilbert, **Interpretation of Imperfect Line Data as a Three-dimensional Scene**, *Artificial Intelligence J.*, Vol. 3, No. 2, 1972.

43. Feigenbaum, Edward, **Information Processing and Memory**, in *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 4, U.C. Press, Berkeley, 1967.

44. Feigenbaum, Edward, Joshua Lederberg, Bruce Buchanan, **Heuristic Dendral**, *Proc. International Conference on System Sciences*, University of Hawaii and IEEE, University of Hawaii Press, 1968.

45. Feigenbaum, Edward, **Artificial Intelligence: Themes in the Second Decade**, *Proc. IFIP Congress*, 1968.

46. Feigenbaum, Edward, Bruce Buchanan, Joshua Lederberg, **On Generality and Problem Solving: A Case Study using the DENDRAL Program**, *Machine Intelligence 6*, Edinburgh Univ. Press, 1971.

47. Feldman, Jerome, D. Gries, **Translator Writing Systems**, *Comm. ACM*, February 1968.

48. Feldman, Jerome, P. Rovner, **The Leap Language Data Structure**, *Proc. IFIP Congress*, 1968.

49. Feldman, Jerome, **Machine Intelligence**, review of Numbers I-III of the Machine Intelligence series, *Information and Control*, 14, 490-492, 1969.

50. Feldman, Jerome, **Towards Automatic Programming**, *Preprints of NATO Software Engineering Conference*, Rome, Italy, 1969.

51. Feldman, Jerome, Paul Rovner, **An Algol-based Associative Language**, *Comm. ACM*, August 1969.

52. Feldman, Jerome, Gary Feldman, G. Falk, Gunnar Grape, J. Pearlman, I. Sobel, and J. Tenenbaum, **The Stanford Hand-Eye Project**, *Proc. International Conf. on Artificial Intelligence*, Washington, D.C., 1969.

53. Feldman, Jerome, **Getting a Computer to See Simple Scenes**, *IEEE Student Journal*, Sept. 1970.

54. Feldman, Jerome, Alan Bierman, **A Survey of Grammatical Inference**, *Proc. International Congress on Pattern Recognition*, Honolulu, January 1971, also in S. Watanbe (ed.), *Frontiers of Pattern Recognition*, Academic Press, 1972.

55. Feldman, Jerome, Robert Sproull, **System Support for the Stanford Hand-eye System**, *Proc. 2IJCAI*, Brit. Comp. Soc., Sept. 1971.

56. Feldman, Jerome, et al, **The Use of Vision and Manipulation to Solve the 'Instant Insanity' Puzzle**, *Proc. 2IJCAI*, Brit. Comp. Soc., Sept. 1971.

57. Feldman, Jerome, **Some Decidability Results on Grammatical Inference and Complexity**, *Information and Control*, Vol. 20, No. 3, pp. 244-262, April 1972.

58. Feldman, Jerome, J. Low, D. Swinehart, R. Taylor, **Recent Developments in SAIL, an ALGOL-based language for Artificial Intelligence**, *Proc. Fall Joint Computer Conference*, 1972.

59. Floyd, Robert, **Toward Interactive Design of Correct Programs**, *Proc. IFIP Congress 1971*.

60. Garland, Stephan J., David Luckham, **Translating Recursive Schemes into Program Schemes**, *ACM SIGPLAN Notices*, Vol. 7, No. 1, January 1972.

61. Garland, Stephan J., David C. Luckham, **Program Schemes, Recursion Schemes, and Formal Languages**, *J. Computer and System Sciences*, Vol. 7, No. 2, April 1973.

62. Gips, James, **A New Reversible Figure**, *Perceptual & Motor Skills*, 34, 306, 1972.

63. Goodman, Joseph, **Digital Image Formation from Electronically Detected Holograms**, *Applied Physics Letters*, August 1967.

64. Goodman, Joseph, **Digital Image Formation from Electronically Detected Holograms**, in *Proc. SPIE Seminar on Digital Imaging Techniques*, Soc. Photo-Optical Instrumentation Engineering, Redondo Beach, California, 1967.

65. Gruen, Richard, William Weiher, **Rapid Program Generation**, *Proc. DECUS Symposium*, Fall 1968.

66. Hearn, Anthony, **Computation of Algebraic Properties of Elementary Particle Reactions Using a Digital Computer**, *Comm. ACM*, 9, pp. 573-577, August, 1966.

67. Hearn, Anthony, **REDUCE, A User-Oriented Interactive System for Algebraic Simplification**, *Proc. ACM Symposium on Interactive Systems for Experimental Applied Mathematics*, August 1967.

68. Hearn, Anthony, **The Problem of Substitution**, *Proc. IBM Summer Institute on Symbolic Mathematics by Computer*, July 1968.

69. Hearn, Anthony, **Applications of Symbol Manipulation in Theoretical Physics**, *Comm. ACM*, August 1971.

70. Hilf, Franklin, Kenneth Colby, David Smith, W. Wittner, William Hall, **Machine-Mediated Interviewing**, *J. Nervous & Mental Disease*, Vol. 152, No. 4, 1971.

71. Hilf, Franklin, **Non-Nonverbal Communication and Psychiatric Research**, *Archives of General Psychiatry*, Vol. 27, November 1972.

72. Hilf, Franklin, **Partially Automated Psychiatric Research Tool**, *J. Nervous and Mental Disease*, Vol. 155, No. 6, December 1972.

73. Hueckel, Manfred, **An Operator which Locates Edges in Digitized Pictures**, *JACM*, January 1971.

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

75. Ito, T., **Note on a Class of Statistical Recognition Functions**, *IEEE Trans. Computers*, January 1969.

76. Kahn, Michael, Bernard Roth, **The Near-minimum-time Control of Open-loop Articulated Kinematic Chains**, *Trans. ASME*, Sept. 1971.

77. Kaplan, Donald, **Some Completeness Results in the Mathematical Theory of Computation**, *ACM Journal*, January 1968.

78. Kaplan, Donald, **Regular Expressions and the Completeness of Programs**, *J. Comp. & System Sci.*, Vol 3, No. 4, 1969.

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

80. Kieburtz, Richard B., David Luckham, **Compatibility and Complexity of Refinements of the Resolution Principal**, *SIAM J. Comput.*, 1972.

81. Kieburtz, Richard, David Luckham, **Compatability and Complexity of Refinements of the Resolution Principle**, *SIAM J. on Computing*, 1-4, 1973.

82. Kling, Robert, **A Paradigm for Reasoning by Analogy**, *Proc. 2IJCAI*, Brit. Comp. Soc., Sept. 1971.

83. Knuth, Donald E., *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*, Addison-Wesley, Menlo Park, Calif., 1969.

84. Knuth, Donald E., **An Empirical Study of FORTRAN Programs**, *Software -- Practice and Experience*, Vol. 1, 105-133, 1971.

85. Knuth, Donald E., **Ancient Babylonian Algorithms**, *Comm. ACM*, July 1972.

86. Knuth, Donald E., *The Art of Computer Programming, Vol. 3, Sorting and Searching*, Addison-Wesley, Menlo Park, Calif., 1973.

87. Lederberg, Joshua, **Hamilton Circuits of Convex Trivalent Polyhedra**, *American Mathematical Monthly* 74, 522, May 1967.

88. Lederberg, Joshua, Edward Feigenbaum, **Mechanization of Inductive Inference in Organic Chemistry**, in B. Kleinmuntz (ed.), *Formal Representation of Human Judgment*, John Wiley, New York, 1968.

89. Lederberg, Joshua, **Topology of Organic Molecules**, National Academy of Science, *The Mathematical Sciences: a Collection of Essays*, MIT Press, Cambridge, 1969.

90. Lederberg, Joshua, Georgia Sutherland, Bruce Buchanan, Edward Feigenbaum, A. Robertson, A. Duffield, Carl Djerassi, **Applications of Artificial Intelligence for Chemical Inference I. The Number of Possible Organic Compounds: Acyclic Structures Containing C, H, O, and N**, *J. Amer. Chem. Soc.*, 91:11, May 1969.

91. Lederberg, Joshua, Georgia Sutherland, Bruce Buchanan, Edward Feigenbaum, **A Heuristic Program for Solving a Scientific Inference Problem: Summary of Motivation and Implementation**, in M. Mesarovic (ed.), *Theoretical Approaches to Non-numerical Problem Solving*, Springer-Verlag, New York, 1970.

92. London, Ralph, **Correctness of a Compiler for a LISP Subset**, *ACM SIGPLAN Notices*, Vol. 7, No. 1, January 1972.

93. Luckham, David, **Refinement Theorems in Resolution Theory**, *Proc. 1968 IRIA Symposium in Automatic Deduction*, Versailles, France, Springer-Verlag, 1970.

94. Luckham, David, D. Park and M. Paterson, **On Formalised Computer Programs**, *J. Comp. & System Sci.*, Vol. 4, No. 3, June 1970.

95. Luckham, David, Nils Nilsson, **Extracting Information from Resolution Proof Trees**, *Artificial Intelligence Journal*, Vol. 2, No. 1, pp. 27-54, June 1971.

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

97. Manna, Zohar, **Properties of Programs and the First Order Predicate Calculus**, *J. ACM*, Vol. 16, No. 2, April 1969.

98. Manna, Zohar, **The Correctness of Programs**, *J. System and Computer Sciences*, Vol. 3, No. 2, May 1969.

99. Manna, Zohar, John McCarthy, **Properties of Programs and Partial Function Logic** in Bernard Meltzer and Donald Michie (eds.), *Machine Intelligence 5*, Edinburgh University Press, 1970.

100. Manna, Zohar, **The Correctness of Non-Deterministic Programs**, *Artificial Intelligence Journal*, Vol. 1, No. 1, 1970.

101. Manna, Zohar, **Termination of Algorithms Represented as Interpreted Graphs**, *AFIPS Conference Proc. (SJCC)*, Vol. 36, 1970.

102. Manna, Zohar, **Second-order Mathematical Theory of Computation**, *Proc. ACM Symposium on Theory of Computing*, May 1970.

103. Manna, Zohar, Amir Pnueli, **Formalization of Properties of Functional Programs**, *J. ACM*, Vol. 17, No. 3, July 1970.

104. Manna, Zohar, R. Waldinger, **Toward Automatic Program Synthesis**, *Comm. ACM*, March 1971.

105. Manna, Zohar, **Mathematical Theory of Partial Correctness**, *J. Comp. & Sys. Sci.*, June 1971.

106. Manna, Zohar, S. Ness, J. Vuillemin, **Inductive Methods for Proving Properties of Programs**, *ACM SIGPLAN Notices*, Vol. 7, No. 4, January 1972.

107. Manna, Zohar, J. Vuillemin, **Fixpoint Approach to the Theory of Computation**, *Comm. ACM*, July 1972.

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

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

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

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

112. McCarthy, John, **Towards a Mathematical Theory of Computation**, in *Proc. IFIP Congress 62*, North-Holland, Amsterdam, 1963.

113. McCarthy, John, **A Basis for a Mathematical Theory of Computation**, in P. Biaffort and D. Hershberg (eds.), *Computer Programming and Formal Systems*, North-Holland, Amsterdam, 1963.

114. McCarthy, John, S. Boilen, E. Fredkin, J.C.R. Licklider, **A Time-Sharing Debugging System for a Small Computer**, *Proc. AFIPS Conf.* (SJCC), Vol. 23, 1963.

115. McCarthy, John, F. Corbato, M. Daggett, **The Linking Segment Subprogram Language and Linking Loader Programming Languages**, *Comm. ACM*, July 1963.

116. McCarthy, John, **Problems in the Theory of Computation**, *Proc. IFIP Congress 1965*.

117. McCarthy, John, **Time-Sharing Computer Systems**, in W. Orr (ed.), *Conversational Computers*, Wiley, 1966.

118. McCarthy, John, **A Formal Description of a Subset of Algol**, in T. Steele (ed.), *Formal Language Description Languages for Computer Programming*, North-Holland, Amsterdam, 1966.

119. McCarthy, John, **Information**, *Scientific American*, September 1966.

120. McCarthy, John, **Computer Control of a Hand and Eye**, in *Proc. Third All-Union Conference on Automatic Control (Technical Cybernetics)*, Nauka, Moscow, 1967 (Russian).

121. McCarthy, John, D. Brian, G. Feldman, and J. Allen, **THOR -- A Display Based Time-Sharing System**, *Proc. AFIPS Conf.* (FJCC), Vol. 30, Thompson, Washington, D.C., 1967.

122. McCarthy, John, James Painter, **Correctness of a Compiler for Arithmetic Expressions**, Amer. Math. Soc., *Proc. Symposia in Applied Math., Math. Aspects of Computer Science*, New York, 1967.

123. McCarthy, John, **Programs with Common Sense**, in Marvin Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, 1968.

124. McCarthy, John, Lester Earnest, D. Raj. Reddy, Pierre Vicens, **A Computer with Hands, Eyes, and Ears**, *Proc. AFIPS Conf.* (FJCC), 1968.

125. McCarthy, John, Patrick Hayes, **Some Philosophical Problems from the Standpoint of Artificial Intelligence**, in Donald Michie (ed.), *Machine Intelligence 4*, American Elsevier, New York, 1969.

126. Milner, Robin, **An Algebraic Definition of Simulation between Programs**, *Proc. 2IJCAI*, Brit. Comp. Soc., Sept. 1971.

127. Milner, Robin, **Implementation and Application of Scott's Logic for Computable Functions**, *ACM SIGPLAN NOTICES*, Vol. 7, No. 1, January 1972.

128. Milner, Robin, Richard Weyhrauch, **Proving Compiler Correctness in a Mechanized Logic**, *Machine Intelligence 7*, Edinburgh University Press, 1972.

129. Montanari, Ugo, **Continuous Skeletons from Digitized Images**, *JACM*, October 1969.

130. Montanari, Ugo, **A Note on Minimal Length Polygonal Approximation to a Digitized Contour**, *Comm. ACM*, January 1970.

131. Montanari, Ugo, **On Limit Properties in Digitization Schemes**, *JACM*, April 1970.

132. Montanari, Ugo, **Separable Graphs, Planar Graphs and Web Grammars**, *Information and Control*, May 1970.

133. Montanari, Ugo, **Heuristically Guided Search and Chromosome Matching**, *J. Artificial Intelligence*, Vol. 1, No. 4, December 1970.

134. Montanari, Ugo, **On the Optimal Detection of Curves in Noisy Pictures**, *Comm. ACM*, May 1971.

135. Moorer, James A., **Dual Processing for the PDP-6/10**, *Decuscope*, Vol. 8, No. 3, 1969.

136. Moorer, James A., **Music and Computer Composition**, *Comm. ACM*, January 1972.

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

138. Nilsson, Nils, *Problem-solving Methods in Artificial Intellegence*, McGraw-Hill, New York, 1971.

139. Paul, Richard, G. Falk, Jerome Feldman, **The Computer Representation of Simply Described Scenes**, *Proc. 2nd Illinois Graphics Conference*, Univ. Illinois, April 1969.

140. Paul, Richard, Gilbert Falk, Jerome Feldman, **The Computer Description of Simply Described Scenes**, in *Pertinent Concepts in Computer Graphics*, J. Neivergelt and M. Faiman (eds.), U. Illinois Press, 1969.

141. Paul, Richard, **Trajectory Control of a Computer Arm**, *Proc. 2IJCAI*, Brit. Comp. Soc., Sept. 1971.

142. Pingle, Karl, J. Singer, and W. Wichman, **Computer Control of a Mechanical Arm through Visual Input**, *Proc. IFIP Congress* 1968, 1968.

143. Pingle, Karl, **Visual Perception by a Computer**, *Automatic Interpretation and Classification of Images*, Academic Press, New York, 1970.

144. Pingle, Karl, J. Tenenbaum, **An Accomodating Edge Follower**, *Proc. 2IJCAI*, Brit. Comp. Soc., Sept. 1971.

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

146. Reddy, D. Raj., **Segmentation of Speech Sounds**, *J. Acoust. Soc. Amer.*, August 1966.

147. Reddy, D. Raj., **Phoneme Grouping for Speech Recognition**, *J. Acoust. Soc. Amer.*, May 1967.

148. Reddy, D. Raj., **Pitch Period Determination of Speech Sounds**, *Comm. ACM*, June 1967.

149. Reddy, D. Raj., **Computer Recognition of Connected Speech**, *J. Acoust. Soc. Amer.*, August 1967.

150. Reddy, D. Raj., **Computer Transcription of Phonemic Symbols**, *J. Acoust. Soc. Amer.*, August 1968.

151. Reddy, D. Raj., **Consonantal Clustering and Connected Speech Recognition**, *Proc. Sixth International Congress on Acoustics*, Vol. 2, pp. C-57 to C-60, Tokyo, 1968.

152. Reddy, D. Raj., Ann Robinson, **Phoneme-to-Grapheme Translation of English**, *IEEE Trans. Audio and Electroacoustics*, June 1968.

153. Reddy, D. Raj., Pierre Vicens, **Procedure for Segmentation of Connected Speech**, *J. Audio Eng. Soc.*, October 1968.

154. Roth, Bernard, **Design, Kinematics, and Control of Computer-controlled Manipulators**, *Proc. 6th All Union Conference on New Problems in Theory of Machines & Mechanics*, Leningrad, Jan. 1971.

155. Sagan, Carl, J. Lederberg, E. Levinthal, L. Quam, R. Tucker, et al, **Variable Features on Mars: Preliminary Mariner 9 Television Results**, *Icarus 17*, pages 346-372, 1972.

156. Samuel, Arthur, **Studies in Machine Learning Using the Game of Checkers, II-Recent Progress**, *IBM Journal*, November 1967.

157. Schank, Roger, Larry Tesler, **A Conceptual Parser for Natural Language**, *Proc. International Joint Conference on Artificial Intelligence*, Washington, D.C., 1969.

158. Schank, Roger, **Finding the Conceptual Content and Intention in an Utterance in Natural Language Conversation**, *Proc. 2IJCAI*, Brit. Comp. Soc., 1971.

159. Schank, Roger, **Conceptual Dependency: a Theory of Natural Language Understanding**, *Cognitive Psychology*, Vol 3, No. 4, 1972.

160. Schank, Roger C., Neil M. Goldman, **Theoretical Considerations in Text Processing**, *Conf. Proc. Computer Text*

Processing and Scientific Research (1972), O.N.R., Pasadena, Calif., March 1973.

161. 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.

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

163. Schroll, G., A. Duffield, Carl Djerassi, Bruce Buchanan, G. Sutherland, Edward Feigenbaum, Joshua Lederberg, **Applications of Artificial Intelligence for Chemical Inference III. Aliphatic Ethers Diagnosed by Their Low Resolution Mass Spectra and NMR Data**, *J. Amer. Chem. Soc.*, 91:26, December 1969.

164. Sheikh, Y., A. Buchs, A Delfino, Bruce Buchanan, G. Sutherland, Joshua Lederberg, **Applications of Artificial Intelligence for Chemical Inference V. An Approach to the Computer Generation of Cyclic Structures. Differentiation Between All the Possible Isometric Ketones of Composition C6H10O**, *Organic Mass Spectrometry*, Vol. 4 pp.493-501, 1970.

165. Silvestri, Anthony, Joseph Goodman, **Digital Reconstruction of Holographic Images**, 1968 NEREM Record, IEEE, Vol. 10, pp. 118-119. 1968.

166. Slagle, James, Carl Farrell, **Experiments in Automatic Learning for a Multiputpose Heuristic Program**, *Comm. ACM*, February 1971.

167. Smith, D. H., B. G. Buchanan, R. S. Engelmore, A. M. Duffield, A. Yeo, E. A. Feigenbaum, J. Lederberg, 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 Estrogenic Steroids**, *Journal of the American Chemical Society*, 94, 5962-5971, 1972.

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

169. Smith, Leland, **SCORE -- A Musician's Approach to Computer Music**, *J. Audio Eng. Soc.*, Jan./Feb. 1972.

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

171. Sobel, Irwin, **On Calibrating Computer Controlled Cameras for Perceiving 3-D Scenes**, *Proc. Third Int. Joint Conf. on Artificial Intelligence*, Stanford U., 1973.

172. 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.

173. Sullivan, S. Brodsky and J., **W-Boson Contribution to the Anomalous Magnetic Moment of the Muon**, *Phys Rev* 156, 1644, 1967.

174. Sutherland, Georgia, C. W. Evans, G. F.Wallace, *Simulation Using Digital Computers*, Prentice-Hall, Engelwood Cliffs, N.J., 1967.

175. Tenenbaum, Jay, et al, **A Laboratory for Hand-eye Research**, *Proc. IFIP Congress*, 1971.

176. Tesler, Larry, Horace Enea, Kenneth Colby, **A Directed Graph Representation for Computer Simulation of Belief Systems**, *Math. Bio.* 2, 1968.

177. 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.

178. Waterman, Donald, **Generalization Learning Techniques for Automating the Learning of Heuristics**, *J. Artificial Intelligence*, Vol. 1, No. 1/2.

179. Weyhrauch, Richard, Robin Milner, **Program Semantics and Correctness in a Mechanized Logic**, *Proc. USA-Japan Computer Conference*, Tokyo, 1972.

180. Wilkes, Yorick, **Semantic Considerations in Text Processing**, *Conf. Proc. Computer Text Processing and Scientific Research (1972)*, O.N.R., Pasadena, Calif., March 1973.

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

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

183. 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.

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

185. 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.

Appendix E

A. I. MEMO ABSTRACTS

In the listing below, there are up to three numbers given for each Artificial Intelligence Memo: 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. If there is no "*" to the left of the AIM number, then it is in stock at Stanford at this writing and may be requested from:

Documentation Services
Artificial Intelligence Laboratory
Stanford University
Stanford, California 94305

Alternatively, if there is an NTIS number given, then the report may be ordered using that number from:

National Technical Information Service
P. O. 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. O. 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. See Appendix A for directions on how to access such files.

AIM-1
John McCarthy,
Predicate Calculus with 'Undefined' as a Truth-value,
5 pages, March 1963.

The use of predicate calculus in the mathematical theory of computation and the problems involved in interpreting their values.

AIM-2
John McCarthy,
Situations, Actions, and Causal Laws,
11 pages, July 1963.

A formal theory is given concerning situations, causality and the possibility and effects of actions is given. The theory is intended to be used by the Advice Taker, a computer program that is to decide what to do by reasoning. Some simple examples are given of descriptions of situations and deductions that certain goals can be achieved.

AIM-3
Fred Safier,
'The Mikado' an an Advice Taker Problem,
4 pages, July 1963.

The situation of the Second Act of 'The Mikado' is analyzed from the point of view of Advice Taker formalism. This indicates defects still present in the language.

*AIM-4
Horace Enea,
Clock Function for LISP 1.5,
2 pages, August 1963.

This paper describes a clock function for LISP 1.5.

AIM-5
Horace Enea, Dean Wooldridge,
Algebraic Simplication,
2 pages, August 1963.

Herein described are proposed and effected changes and additions to Steve Russell's Mark IV Simplify.

*AIM-6
Dean Wooldridge,
Non-printing Compiler,
2 pages, August 1963.

A short program which redefines parts of the LISP 1.5 compiler and suppresses compiler print out (at user's option) is described.

AIM-7
John McCarthy,
Programs With Common Sense,
7 pages, September 1963.

Interesting work is being done in programming computers to solve problems which require a high degree of intelligence in humans. However, certain elementary verbal reaesoning processes so simple they can be carried out by any non-feeble-minded human have yet to be simulated by machine programs.

This paper will discuss programs to manipulate in a suitable formal language (most likely a part of the predicate calculus) common instrumental statements. The basic program will draw immediate conclusions from a list of premises. These conclusions will be either declarative or imperative sentences. When an imperative sentence is deduced the program takes a corresponding action. These actions may include printing sentences, moving sentences on lists, and reinitiating the basic deduction process on these lists.

Facilities will be provided for communication with humans in the system via manual intervention and display devices connected to the computer.

*AIM-8
John McCarthy,
Storage Conventions in LISP 2,
? pages, September 1963.

Storage conventions and a basic set of functions for LISP 2 are proposed. Since the memo was written, a way of supplementing the features of this system with the unique storage of list structure using a hash rule for computing the address in a separate free storage area for lists has been found.

*AIM-9
C. M. Williams,
Computing Estimates for the Number of Bisections of an NxN Checkerboard for N Even,
9 pages, December 1963.

This memo gives empirical justification for the assumption that the number of bisections of an NxN (N even) checkerboard is approximately given by the binomial coefficient (A, A/2) where 2A is the length of the average bisecting cut.

AIM-10
Stephan R. Russell,
Improvements in LISP Debugging,
3 pages, December 1963.

Experience with writing large LISP progrrams and helping students learning LISP suggests that spectacular improvements can be made in this area. These improvements are partly an elimination of sloppy coding in LISP 1.5, but mostly an elaboration of DEFINE, the push down list backtrace, and the current tracing facility. Experience suggests that these improvements would reduce the number of computer runs to debug a program a third to a half.

AIM-11
Dean Wooldridge, Jr.,
An Algebraic Simplify Program in LISP,
57 pages, December 1963.

A program which performs 'obvious' (non-controversial) simplifying transformations on algebraic expressions (written in LISP prefix notation) is described. Cancellation of inverses and consolidation of sums and products are the basic accomplishments of the program; however, if the user desires to do so, he may request the program to perform special tasks, such as collect common factors from the products in sums or expand products. Polynomials are handled by routines which take advantage of the special form by polynomials; in particular, division (not cancellation) is always done in terms of polynomials. The program (run on the IBM 7090) is slightly faster than a human; however, the computer does not need to check its work by repeating the simplification.

Although the program is usable -- no bugs are known to exist -- it is by no means a finished project. A rewriting of the simplify system is anticipated; this will eliminate much of the existing redundancy and other inefficiency, as well as implement an identity-recognizing scheme.

AIM-12
Gary Feldman,
Documentation of the MacMahon Squares Problem,
4 pages, January 1964.

An exposition of the MacMahon Squares problem together with some 'theoretical' results on the nature of its solutions and a short discussion of an ALGOL program which finds all solutions are contained herein.

AIM-13
Dean E. Wooldridge,
The New LISP System (LISP 1.55),
4 pages, February 1964.

The new LISP system is described. Although differing only slightly it is thought to be an improvement on the old system.

AIM-14
John McCarthy,
Computer Control of a Machine for Exploring Mars,
6 pages, January 1964.

Landing a 5000 pound package on Mars that would spend a year looking for life and making other measurements has been proposed. We believe that this machine should be a stored program computer with sense and motor organs and that the machine should be mobile. We discuss the following points:

1. Advantages of a computer controlled system.
2. What the computer should be like.
3. What we can feasible do given the present state of work on artificial intelligence.
4. A plan for carrying out research in computer controlled experiments that will make the Mars machine as effective as possible.

AIM-15
Mark Finkelstein, Fred Safier,
Axiomatization and Implementation,
6 pages, June 1964.

An example of a typical Advice-Taker axiomatization of a situation is given, and the situation is programmed in LISP as an indication of how the Advice-Taker could be expected to react. The situation chosen is the play of a hand of bridge.

AIM-16
John McCarthy,
A Tough nut for Proof Procedures,
3 pages, July 1964.

It is well known to be impossible to tile with dominoes a checkerboard with two opposite corners deleted. This fact is readily stated in the first order predicate calculus, but the usual proof which involves a parity and counting argument does not readily translate into predicate calculus. We conjecture that this problem will be very difficult for programmed proof procedures.

AIM-17
John McCarthy,
Formal Description of the Game of Pang-Ke,
2 pages, July 1964.

The game of Pang-Ke is formulated in a first-order-logic in order to provide grist for the Advice-Taker Mill. The memo does not explain all the terms used.

AIM-18
Jan Hext,
An Expression Input Routine for LISP,
5 pages, July 1964.

The expression input routine is a LISP function, Mathread [ ] with associated definitions, which reads in expressions such as (A+3-F(X,Y,Z)). Its result is an equivalent S-expression. The syntax of allowable expressions is given, but (unlike ALGOL's) it does not define the precedence of the operators; nor does the program carry out any explicit syntax analysis. Instead the program parses the expression according to a set of numerical precedence values, and reports if it finds any symbol out of context.

*AIM-19
Jan Hext,
Programming Languages and Translation,
14 pages, August 1964.

A notation is suggested for defining the syntax of a language in abstract form, specifying only its semantic constituents. A simple language is presented in this form and its semantic definition given in terms of these constituents. Methods are then developed for translating this language, first into LISP code and from there to machine code, and for proving that the translation is correct.

AIM-20
D. Raj. Reddy,
Source Language Optimization of For-loops,
37 pages, August 1964.

Program execution time can be reduced, by a considerable amount, by optimizing the 'For-loops' of Algol programs. By judicious use of index registers and by evaluating all the sub-expressions whose values are not altered within the 'For-loop', such optimization can be achieved.

In this project we develop an algorithm to optimize Algol programs in list-structure form and generate a new source language program, which contains the 'desired contents in the index registers' as a part of the For-clause of the For-statement and additional statements for evaluating the same expressions outside the 'For-loop' This optimization is performed only for the innermost 'For-loops'.

The program is written entirely in LISP. Arrays may have any number of subscripts. Further array declarations may have variable dimensions. (Dynamic allocation of storage.) The program does not try to optimize arithmetic expressions. (This has already been extensively investigated.)

AIM-21
R. W. Mitchell,
LISP 2 Specifications Proposal,
12 pages, August 1964.

Specifications for a LISP 2 system are proposed. The source language is basically Algol 60 extended to include list processing, input/output and language extension facilities. The system would be implemented with a source language translator and optimizer, the output of which could be processed by either an interpreter or a compiler. The implementation is specified for a single address computer with particular reference to an IBM 7090 where necessary .

Expected efficiency of the system for list processing is significantly greater than the LISP 1.5 compiler. For execution of numeric algorithms the systems should be be comparable to many current "algebraic" compilers. Some familiarity with LISP, 1.5 Algol and the IBM 7090 is assumed.

AIM-22
Richard Russell,
Kalah -- the Game and the Program,
13 pages, September 1964.

A description of Kalah and the Kalah program, including sub-routine descriptions and operating instructions.

AIM-23
Richard Russell,
Improvements to the Kalah Program,
12 pages, September 1964.

Recent improvements to the Kalah program are listed, and a proposal for speeding up the program by a factor of three is discussed.

AIM-24
John McCarthy,
A Formal Description of a Subset of ALGOL,
43 pages, September 1964.

We describe Microalgol, a trivial subset of Algol, by means of an interpreter. The notions of abstract syntax and of 'state of the computation' permit a compact description of both syntax and semantics. We advocate an extension of this technique as a general way of describing programming language.

AIM-25
Richard Mansfield,
A Formal System of Computation,
7 pages, September 1964.

We discuss a tentative axiomatization for a formal system of computation and within this system we prove certain propositions about the convergence of recursive definitions proposed by J. McCarthy.

AIM-26
D. Raj. Reddy,
Experiments on Automatic Speech Recognition by a Digital Computer,
19 pages, October 1964.

Speech sounds have in the past been investigated with the aid of spectographs, vocoders and other analog devices. With the availability of digital computers with improved i-o devices such as Cathode Ray tubes and analog digital converters, it has recently become practicable to employ this powerful tool in the analysis of speech sounds.

Some papers have appeared in the recent literature reporting the use of computers in the determination of the fundamental frequency and for vowel recognition. This paper discusses the details and results of a preliminary investigation conducted at Stanford. It includes various aspects of speech sounds such as waveforms of vowels and consonants; determination of a fundamental of the wave; Fourier (spectral) analysis of the sound waves format determination, simple vowel recognition algorithm and synthesis of sounds. All were obtained by the use of a digital computer.

AIM-27
John McCarthy,
A Proof-checker for Predicate Calculus,
7 pages, March 1965.

A program that checks proofs in J. A. Robinson's formulation of predicate calculus has been programmed in LISP 1.5. The program is available in CTSS at Project MAC and is also available as a card deck. The program is used for class exercises at Stanford.

AIM-28
John McCarthy,
Problems in the Theory of Computation,
7 pages, March 1965.

The purpose of this paper is to identify and discuss a number of theoretical problems whose solutions seem feasible and likely to advance the practical art of computation. The problems that will be discussed include the following:

1. Semantics of programming languages. What do the strings of symbols representing computer programs, statements, declarations, labels, etc., denote? How can the semantics of programming languages be described formally?

2. Data spaces. What are the spaces of data on which computer programs act and how are they built up up from simpler spaces?

3. How can time dependent and simultaneous processes be described?

4. Speed of computation. What can be said about how much computation is required to carry out certain processes?

5. Storage of information. How can information be stored so that items identical or similar to a given item can be retrieved?

6. Syntax directed computation. What is the appropriate domain for computations described by productions or other data format recognizers?

7. What are the appropriate formalisms for writing proofs that computer programs are equivalent?

8. In the view of Godel's theorem that tells us that any formal theory of computation must be incomplete, what is a reasonable formal system that will enable us to prove that programs terminate in practical cases?

AIM-29
Charles M. Williams,
Isolation of Important Features of a Multitoned Picture,
9 pages, January 1965.

A roughly successful attempt is made to reduce a multi-toned picture to a two-toned (line drawing) representation capable of being recognized by a human being.

AIM-30
Edward A. Feigenbaum, Richard W. Watson,
An Initial Problem Statement for a Machine Induction Research Project,
8 pages, April 1965.

A brief description is given of a research project presently getting under way. This project will study induction by machine, using organic chemistry as a task area. Topics for graduate student research related to the problem are listed.

AIM-31
John McCarthy,
Plans for the Stanford Artificial Intelligence Project,
17 pages, April 1965.

The following is an excerpt from a proposal to ARPA and gives some of the project plans for the near future.

AIM-32
Harry Ratchford,
The 138 Analog Digital Converter,
9 pages, May 1965.

A discussion of the programming and hardware characteristics of the analog to digital converter on the PDP-1 is given; several sample programs are also presented.

AIM-33
Barbara Huberman,
The Advice Taker and GPS,
8 pages, June 1965.

Using the formalism of the Newell-Shaw-Simon General Problem Solver to solve problems expressed in McCarthy's Advice Taker formalism is discussed. Some revisions of the formalism of *can* and *cause* described in AI Memo 2 are proposed.

AIM-34
Peter Carah,
A Television Camera Interface for the PDP-1,
8 pages, June 1965.

This paper is a discussion of several methods for the connection of a television camera to the PDP-1 computer. Three of these methds are discussed in detail and have in common that only a 36 bit portion of any horizontal scanning line may be read and this information is read directly into the working registers of the computer. The fourth involves a data channel to read information directly into the core memory of the computer, and is mentioned only in passing. The major concepts and some of the details of these methods are due to Marvin Minsky.

*AIM-35
Fred Safier,
Simple Simon,
17 pages, June 1965.

SIMPLE SIMON is a program which solves the problem of finding an object satisfying a predicate from a list of facts. It operates by backward chaining. The rules of procedure and heuristics are discussed and the structure of the program is outlined.

AIM-36
James Painter,
Utilization of a TV Camera on the PDP-1,
6 pages, September 1965.

A description of the programming required to utilize the TV camera connected to the PDP-1 and of the initial collection of programs.

*AIM-37
Knut Korsvold,
An on Line Algebraic Simplification Program,
36 pages, November 1965.

We describe an on-line program for algebraic simplification. The program is written in LISP 1.5 for the Q-32 computer at System Development Corporation in Santa Monica, California. The program has in its entirety been written and debugged from a teletype station at Stanford University.

AIM-38
Donald A. Waterman,
A Filter for a Machine Induction System,
19 pages, January 1966.

This report contains current ideas about the Machine Induction Research Project, and attempts to more clearly define some of the problems involved. In particular, the on-line data acquisition problem, the filter, and the inductive inference problem associated with the filter are discussed in detail.

AIM-39
Karl Pingle,
A Program to Find Objects in a Picture,
22 pages, January 1966.

A program is described which traces around objects in a picture, using the picture scanner attached to the PDP-1 computer, and fits curves to the edges.

AIM-40 CS-38 AD662880
John McCarthy, James Painter,
Correctness of a Compiler for Arithmetic Expressions,
13 pages, April 1966.

This is a preprint of a paper given at the Symposium of Mathematical Aspects of Computer Science of the American Mathematical Society held April 7 and 8, 1966. It contains a proof of the correctness of a compiler for arithmetic expressions.

*AIM-41
Phil Abrams, Dianna Rode,
A Proposal for a Proof-checker for Certain Axiomatic Systems,
10 pages, May 1966.

A proposed design for a proof-checker to operate on many axiomatic domains is presented. Included are descriptions of the organization and operation of the program to be written for the PDP-6.

AIM-42
Karl Pingle,
A Proposal for a Visual Input Routine,
11 pages, June 1966.

Some comments are made on the characteristics believed desirable in the next eye for the Stanford Artificial Intelligence Project and a proposal is given for a program to input scenes using the eye.

*AIM-43 CS-49 SS640-836
D. Raj Reddy,
An Approach to Computer Speech Recognition by Direct Analysis of the Speech Wave,
Thesis: Ph.D. in Computer Science,
144 pages, September 1966.

A system for obtaining a phonemic transcription from a connected speech sample entered into the computer by a microphone and an analog-to-digital converter is described. A feature-extraction program divides the speech utterance into segments approximately corresponding to phonemes, determine pitch periods of those segments where pitch analysis is appropriate, and computes a list of parameters for each segment. A classification program assigns a phoneme-group label (vowel-like segment, fricative-like segment, etc.) to each segment, determines whether a segment should be classified as a phoneme or whether it represents a phoneme boundary between two phonemes, and then assigns a phoneme label to each segment that is not rejected as being a phoneme boundary. About 30 utterances of one to two seconds duration were analyzed using the above programs on an interconnnected IBM 7090-PDP-1 system. Correct identification of many vowel and consonantal phonemes was achieved for a single speaker. The time for analysis of each utterance was about 40 times real time. The results were encouraging and point to a new direction in speech research.

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

This is a semantic proof of the correctness of a compiler. The abstract syntax and semantic definition are given for the language Mickey, an extension of Micro-algol. The abstract syntax and semantics are given for a hypothetical one-register single-address computer with 14 operations. A compiler, using recursive descent, is defined. Formal definitions are also given for state vector, a and c functions, and correctness of a compiler. Using these definitions, the compiler is proven correct.

AIM-45
Donald Kaplan,
Some Completeness Results in the Mathematical Theory of Computation,
22 pages, October 1966.

A formal theory is described which incorporates the 'assignment' function a(i, k, psi) and the 'contents' function c(i, psi). The axioms of the theory are shown to comprise a complete and consistent set.

*AIM-46 CS-50 PB176761
Staffan Persson,
Some Sequence Extrapolating Programs: a Study of Representation and Modeling in Inquiring Systems,
Thesis: Ph.D. in Computer Science,
176 pages, September 1966.

The purpose of this thesis is to investigate the feasibility of designing mechanized inquiring-systems for finding suitable representations of problems, i.e., to perform the 'creative' task of finding analogies. Because at present a general solution to this problem does not seem to be within reach, the feasibility of mechanizing a particular representational inquirer is chosen as a reasonable first step towards an increased understanding of the general problem. It is indicated that by actually designing, programming and running a representational inquirer as a program for a digital computer, a severe test of its consistency and potential for future extensions can be performed.

*AIM-47
Bruce Buchanan,
Logics of Scientific Discovery,
Thesis: Ph.D. in Philosophy U.C. Berkeley,
210 pages, December 1966.

The concept of a logic of discovery is discussed from a philosophical point of view. Early chapters discuss the concept of discovery itself, some arguments have been advanced against the logics of discovery, notably by N. R. Hanson, and S. E. Toulmin. While a logic of discovery is generally understood to be an algorithm for formulating hypotheses, other concepts have been suggested. Chapters V and VI explore two of these: (A) a set of criteria by which a hypothesis could be judged reasonable, and (B) a set of rational (but not necessarily effective) methods for formulating hypotheses.

AIM-48
Donald M. Kaplan,
Correctness of a Compiler for Algol-like Programs,
46 pages, July 1967.

A compiling algorithm is given which maps a class of Algol-like programs into a class of machine language programs. The semantics, i. e., the effect of execution, of each class is specified, and recursion induction used to prove that program semantics is preserved under the mapping defined by the compiling algorithm.

AIM-49
Georgia Sutherland,
DENDRAL -- a Computer Program for Generating and Filtering Chemical Structures,
34 pages, February 1967.

A computer program has been written which can generate all the structural isomers of a chemical composition. The generated structures are inspected for forbidden substructures in order to eliminate structures which are chemically impossible from the output. In addition, the program contains heuristics for determining the most plausible structures, for utilizing supplementary data, and for interrogating the on-line user as to desired options and procedures. The program incorporates a memory so that past experiences are utilized in later work.

AIM-50
Anthony C. Hearn,
Reduce Users' Manual,
53 pages, February 1967.

REDUCE is a program designed for general algebraic computations of interest to physicists and engineers. Its capabilities include:

1) expansion and ordering of rational functions of polynomials,
2) symbolic differentiation,
3) substitutions in a wide variety of forms,
4) reduction of quotients of polynomials by cancellation of common factors,
5) calculation of symbolic determinants,
6) calculations of interest to high energy physicists including spin 1/2 and spin 1 algebra.

The program is written completely in the language LISP 1.5 and may therefore be run with little modification on any computer possessing a LISP 1.5 compiler or interpreter.

AIM-51
Lester D. Earnest,
Choosing an eye for a Computer,
154 pages, April 1967.

In order for a computer to operate efficiently in an unstructured environment, it must have one or more manipulators (e. g., arms and hands) and a spatial sensor analogous to the human eye. Alternative sensor systems are compared here in their performance on certain simple tasks. Techniques for determining color, texture, and depth of surface elements are examined. Sensing elements considered include the photomultiplier, image dissector, image orthicon, vidicon, and SEC camera tube. Performance measures strongly favor a new (and undemonstrated) configuration that may be termed a laser jumping spot system.

AIM-52
Arthur L. Samuel,
Some Studies in Machine Learning Using the Game of Checkers II - Recent Progress,
48 pages, June 1967.

A new signature table technique is described together with an improved book learning procedure which is thought to be much superior to the linear polynomial method described earlier. Full use is made of the so called *alpha-beta* pruning and several forms of forward pruning to restrict the spread of the move tree and to permit the program to look ahead to a much greater depth than it otherwise could do. While still unable to outplay checker masters, the programs's playing ability has been greatly improved. Some of these newer techniques should be applicable to problems of economic importance.

AIM-53
William Weiher,
The PDP-6 Proof Checker,
47 pages, June 1967.

A descrription is given for the use of a proof checker for propositional calculus. An example of its use as well as the M and S expressions for the proof checker are also included.

AIM-54
Joshua Lederberg, Edward A. Feigenbaum,
Mechanization of Inductive Inference in Organic Chemistry,
29 pages, August 1967.

A computer program for formulating hypotheses in the area of organic chemistry is described from two standpoints: artificial intelligence and organic chemistry. The Dendral Algorithm for uniquely representing and ordering chemical structures defines the hypothesis-space; but heuristic search through the space is necessary because of its size. Both the algorithm and the heuristics

are described explicitly but without reference to the LISP code in which these mechanisms are programmed. Within the program some use has been made of man-machine interaction, pattern recognition, learning, and tree-pruning heuristics as well as chemical heuristics which allow the program to focus its attention on a subproblem to rank the hypotheses in order of plausibility. The current performance of the program is illustrated with selected examples of actual output showing both its algorithmic and heuristic aspects. In addition some of the more important planned modifications are discussed.

AIM-55
Jerome Feldman,
First Thoughts of Grammatical Inference,
18 pages, August 1967.

A number of issues relating to the problem of inferring a grammar are disscussed. A strategy for grammatical inference is presented and its weaknesses and possible improvements are discussed. This is a working paper and should not be reproduced, quoted or believed without the author's permission. .

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

This paper reports an experimental investigation of the application of visual feedback to a simple computer-controller block-stacking task. The system uses a vidicon camera to examine a table containing two cubical blocks, generating a data structure which is analyzed to determine the position of one block. An electric arm picks up the block and removes it from the scene, then after the program locates the second block, places the first on top of the second. Finally, the alignment of the stack is improved by analysis of the relative position error as seen by the camera. Positions are determined throughout by perspective transformation of edges detected from a single viewpoint, using a support hypothesis to supply sufficient information on depth. The Appendices document a portion of the hardware used in the project.

AIM-57
Anthony C. Hearn,
REDUCE, a User-oriented Interactive System for Algebraic Simplification,
69 pages, October 1967.

This paper describes in outline the structure and use of REDUCE, a program designed for large-scale algebraic computations of interest to applied mathematicians, physicists, and engineers. The capabilities of the system include:

1) expansion, ordering and reduction of rational functions of polynomials,
2) symbol differentiation,
3) substitutions for variables and expressions appearing in other expressions,
4) simplification of symbolic determinants and matrix expressions,
5) tensor and non-commutative algebraic calculations of interest to high energy physicists.

In addition to the operations of addition, subtraction, multiplication, division, numerical exponentiation numerical exponentiation and differentiation, it is possible for the user to add new operators and define rules for their simplification. Derivations of these operators may also may also be defined.

The program is written complete in the language of LISP 1.5 and is organized so as to minimize the effort required in transferring from one LISP system to another.

Some particular problems which have arisen in using REDUCE in a time-sharing environment are also discussed.

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

The objectives of the research reported here are to develop an automated decision process for real time allocation of defense missiles to attacking ballistic missiles in general war and to demonstrate the effectiveness of applying heuristic learning to seek optimality in the process. The approach is to model and simulate a missile defense environment and generate a decision procedure featuring a self-modifying, heuristic decision function which improves its performance with experience. The goal of the decision process that chooses between the feasible allocations is to minimize the total effect of the attack, measured in cumulative loss of target value. The goal is pursued indirectly by considering the more general problem of maintaining a strong defense posture, the ability of the defense system to protect the targets from both current and future loss.

Using simulation and analysis, a set of calculable *features* are determined which effectively reflect the marginal deterioration of defense posture for each allocation in a time interval. A *decision function*, a linear polynomial of the features, is evaluated for each feasible allocation and the allocation having the smallest value is selected. A heuristic learning process is incorporated in the model to evaluate the performancee of the decision process and adjust the decision function coefficients to encourage correct comparison of alternative allocations. Simulated attacks presenting typical defense situations were cycled against the decision procedure with the result that the decision function coefficients converged under the learning process and the decision process become increasingly effective.

*AIM-59
Donald M. Kaplan,
A Formal Theory Concerning the Equivalence of Algorithms,
20 pages, May 1968.

Axioms and rules of inference are given for the derivation of equivalence for algorithms. The theory is shown to be complete for certain subclasses of algorithms, and several applications of the theory are illustrated. This paper was originally presented at the Mathematical Theory of Computation Conference, IBM Yorktown Heights, November 27-30, 1967.

AIM-60 CS-101 AD672923
Donald M. Kaplan,
The Formal Theoretic Analysis of Strong Equivalence for Elemental Programs,
Thesis: Ph.D. in Computer Science,
263 pages, June 1968.

The syntax and semantics is given for elemental programs, and the strong equivalence of these simple ALGOL-like flowcharts is shown to be undecidable. A formal theory is introduced for deriving statements of strong equivalence, and the completeness of this theory is obtained for various sub-cases. Several applications of the theory are discussed. Using a regular expression representation for elemental programs and an unothodox semantics for these expressions, several strong equivalence detecting procedures are developed. This work was completed in essentially its present form March, 1968.

*AIM-61
Takayasu Ito,
Notes on Theory of Computation and Pattern Recognition,
144 pages, May 1968.

This is a collection of some of the author's raw working notes during the period December 1965 - October 1967 besides the introduction. They have been privately or internally distributed for some time. Portions of this work have been accepted for publication; others are being developed for submission to journals. Some aspects and ideas have been referred to and used, sometimes without explicit references, and others are developed by other researchers and the author. Hence we have decided to publish this material as a Computer Science Technical Report, although the author is planning to submit all of these works to some journals, adding several new results (not mentioned in this report), improving notations, definitions and style of presentation in some parts and reformulating completely in other parts.

AIM-62
Bruce Buchanan, Georgia Sutherland,
Heuristic Dendral: a Program for Generating Explanatory Hypotheses in Organic Chemistry,
76 pages, July 1968.

A computer program has been written which can formulate hypotheses from a given set of scientific data. The data consist of the mass spectrum and the empirical formula of an organic chemical compound. The hypotheses which were produced describe molecular structures which are plausible explanations of the data. The hypotheses are generated systematically within the program's theory of chemical stability and within limiting constraints which are inferred from the data by heuristic rules. The program excludes hypotheses inconsistent with the data and lists its candidate explanatory hypotheses in order of decreasing plausibility. The computer program is heuristic in that it searches for plausible hypotheses in a small subset of the total hypothesis space according to heuristic rules learned from chemists.

AIM-63
Donald M. Kaplan,
Regular Expressions and the Equivalence of Programs,
42 pages, July 1968.

The strong equivalence of ALGOL-like programs is, in general, an undecidable property. Several mechanical procedures are discussed which nevertheless are useful in the detection of strong equivalence. These methods depend on a regular expression representation of programs. An unorthodox semantics for these expressions is introduced which appreciably adds to the ability to detect strong equivalence. Several other methods of extending this ability are also discussed.

*AIM-64
Zohar Manna,
Formalization of Properties of Programs,
18 pages, July 1968.

Given a program, an algorithm will be described for constructing an expression, such that the program is valid (i.e., terminates and yields the right answer) if and only if the expression is inconsistent. Similar result for the equivalence problem of programs is given. These results suggest a new approach for proving the validity and equivalence of programs.

AIM-65 CS-106 AD673971
Barbara J. Huberman,
A Program to Play Chess end Games,
Thesis: Ph.D. in Computer Science,
168 pages, August 1968.

A program to play chess end games is described. The model used in the program is

very close to the model assumed in chess books. Embedded in the model are two predicates, BETTER and WORSE, which contain the heuristics of play, different for each end game. The definitions of BETTER and WORSE were obtained by programmer translation from the chess books.

The program model is shown to be a good one for chess and games by the success achieved for three end games. Also the model enables us to prove that the program can reach checkmate from any starting position. Insights about translation from book problem solving methods into computer program heuristics are discussed; they are obtained by comparing the chess book methods with definitions of BETTER and WORSE, and by considering the difficulty encountered by the programmer when doing the translation.

*AIM-66
Jerome A. Feldman, Paul D. Rovner,
An Algol-based Associative Language,
31 pages, August 1968.

A high-level programming language for large complex relational structures has been designed and implemented. The underlying relational data structure has been implemented using a hash-coding technique. The discussion includes a comparison with other work and examples of applications of the language. A version of this paper will appear in the *Communications of the ACM*.

*AIM-67 AD680487
Edward A. Feigenbaum,
Artificial Intelligence: Themes in the Second Decade,
39 pages, August 1968.

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

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

AIM-68
Zohar Manna, Amir Pnueli,
The Validity Problem of the 91-function,
20 pages, August 1968.

Several methods for proving the weak and strong validity of algorithms are presented.

For proving the weak validity (i.e., correctness) we use satisfiability methods, while proving the strong validity (i.e., termination and correctness) we use unsatisfiability methods.

Two types of algorithms are discussed: recursively defined functions and programs.

Among the methods we include known methods due to Floyd, Manna, and McCarthy. All the methods will be introduced quite informally by means of an example (the 91-function).

*AIM-69 AD677588
John McCarthy, Edward Feigenbaum,
Arthur Samuel,
Project Technical Report,
90 pages, September 1968.

Recent work of Stanford Artificial Intelligence Project is summarized in several areas:

Scientific Hypothesis Formation
Symbolic Computation
Hand-Eye Systems

Computer Recognition of Speech
Board Games
Other Projects

AIM-70 AD680072
Anthony C. Hearn,
The Problem of Substitution,
14 pages, December 1968.

One of the most significant features of programs designed for non-numeric calculation is that the size of expressions manipulated, and hence the amount of storage necessary, changes continually during the execution of the program. It is, therefore, usually not possible for the user to know ahead of time whether the calculation will in fact fail because of lack of available computer memory. The key to keeping both the size of intermediate expressions and output under control often lies in the manner in which substitutions for variables and expressions declared by the programmer are implemented by the system. In this paper various methods which have been developed to perform these substitutions in the author's own system REDUCE are discussed. A brief description of the REDUCE system is also given.

AIM-71 AD677520
Pierre Vicens,
Preprocessing for Speech Analysis,
33 pages, October 1968.

This paper describes a procedure, and its hardware implementation, for the extraction of significant parameters of speech. The process involves division of the speech spectrum into convenient frequency bands, and calculation of amplitude and zero-crossing parameters in each of these bands every 10 ms. In the software implementation, a smooth function divides the speech spectrum into two frequency bands (above and below 1000 Hz). In the hardware implementation, the spectrum is divided into three bands using bandpass filters (150-900 Hz, 900-2200 Hz, 2200-5000 Hz). Details of the design and implementation of the hardware device are given.

*AIM-72 CS-116 AD680036
Donald L. Pieper,
The Kinematics of Manipulators under Computer Control,
Thesis: Ph.D. in Mechanical Engineering,
157 pages, October 1968.

The kinematics of manipulators are studied. A model is presented which allows for the systematic description of new and existing manipulators.

Six degree-of-freedom manipulators are studied. Several solutions to the problem of finding the manipulator configuration leading to a specified position and orientation are presented. Numerical as well as explicit solutions are given. The problem of positioning a multi-link digital arm is also discussed.

Given the solution to the position problem, as a set of heuristics is developed for moving a six degree-of-freedom manipulator from an initial position to a final position through a space containing obstacles. This results in a computer program shown to be able to direct a manipulator around obstacles

*AIM-73 AD678878
John McCarthy, Patrick Hayes,
Some Philosophical Problems From the Standpoint of Artificial Intelligence,
51 pages, November 1968.

A computer program capable of acting intelligently in the world must have a general representation of the world in terms of which its inputs are interpreted. Designing such a program requires commitments about what knowledge is and how it is obtained. Thus some of the major traditional problems of philosophy arise in artificial intelligence.

More specifically, we want a computer program that decides what to do by inferring in a formal language that a certain strategy will achieve its assigned goal. This requires formalizing concepts of causality, ability, and knowledge. Such formalisms are also considered in philosophical logic.

The first part of the paper begins with a philosophical point of view that seems to arise naturally once we take seriously the idea of actually making an intelligent machine. We go on to the notions of metaphysically and epistemologically adequate representations of the world and then to an explanation of *can*, *causes*, and *knows*, in terms of a representation of the world by a system of interacting automata. A proposed resolution of the problem of freewill in a deterministic universe and of counterfactual conditional sentences is presented.

The second part is mainly concerned with formalisms within which it can be proved that a strategy will achieve a goal. Concepts of situation, fluent, future operator, action, strategy, result of a strategy and knowledge are formalized. A method is given of constructing a sentence of first order logic which will be true in all models of certain axioms if and only is a certain strategy will achieve a certain goal.

The formalism of this paper represents an advance over (McCarthy 1963) and (Green 1968) in that it permits proof of the correctness of strategies that contain loops and strategies that involve the acquisition of knowledge, and it is also somewhat more concise.

The third part discusses open problems in extending the formalism of Part II.

The fourth part is a review of work in philosophical logic in relation to problems of Artificial Intelligence and discussion of previous efforts to program *general intelligence* from the point of view of this paper. This paper is based on a talk given to the 4th Machine Intelligence Workshop held at Edinburgh, August 12-21, 1968, and is a preprint of a paper to be published in *Machine Intelligence 4* (Edinburgh University Press, 1969).

*AIM-74 CS-118 AD681027
Donald Waterman,
Machine Learning of Heuristics,
Thesis: Ph.D. in Computer Science,
? pages, December 1968.

The research reported here is concerned with devising machine-learning techniques which can be applied to the problem of automating the learning heuristics.

*AIM-75
Roger C. Schank,
A Notion of Linguistic Concept: a Prelude to Mechanical Translation,
21 pages, December 1968.

The conceptual dependency framework has been used as an automatic parser for natural language. Since the parser gives as output a conceptual network capable of expressing meaning in language-free terms, it is possible to regard this as an interlingua. If an interlingua is actually available how might this interlingua be used in translation? The primary problem that one encounters is the definition of just what these concepts in the network are. A concept is defined as an abstraction in terms of percepts and the frequency of connection of other concepts. This definition is used to facilitate the understanding of some of the problems in paraphrasing and translation. The motivation for this abstract definition of linguistic concept is discussed in the context of its proposed use.

DESCRIPTORS: computational linguistics, concepts research, computer understanding.

AIM-76
Roger C. Schank,
A Conceptual Parser for Natural Language,
22 pages, December 1968.

This paper describes an operable automatic parser for natural language. The parser is not concerned with producing the syntactic structure of an input sentence. Instead, it is a conceptual parser, concerned with determining the underlying meaning of the input. The output of the parser is a network of concepts explicating the conceptual relationships in a piece of discourse. The structure of this network is language-free; thus, sentences in different languages or paraphrases within the same language will parse into the same network. The theory behind this representation is outlined in this paper and the parsing algorithm is explained in some detail.

DESCRIPTORS: computational linguistics, concepts, linguistic research, computer understanding.

AIM-77
Joseph D. Becker,
The Modeling of Simple Analogic and Inductive Processes in a Semantic Memory System,
21 pages, January 1969.

In this paper we present a general data structure for a semantic memory, which is distinguished in that a notion of consequence (temporal, causal, logical, or behavioral, depending on interpretation) is a primitive of the data representation. The same item of a data may at one time serve as a logical implication, and at another time as a 'pattern/action' rule for behavior.

We give a definition of 'analogy' between items of semantic information. Using the notions of consequence and analogy, we construct an inductive process in which general laws are formulated and verified on the basis of observations of individual cases. We illustrate in detail the attainment of the rule 'Firemen wear red suspenders' by this process.

Finally, we discuss the relationship between analogy and induction, and their use in modeling aspects of 'perception' and 'understanding'.

AIM-78
D. Raj Reddy,
On the use of Environmental, Syntactic and Probalistic Constraints in Vision and Speech,
23 pages, January 1969.

In this paper we consider both vision and speech in the hope that a unified treatment, illustrating the similarities, would lead to a better appreciation of the problems, and possibly programs which use the same superstructure. We postulate a general perceptual system and illustrate how various existing systems either avoid or ignore some of the difficult problems that must be considered by a general perceptual system. The purpose of this paper is to point out some of the unsolved problems, and to suggest some heuristics that reflect environmental, syntactic, and probabilistic constraints useful in visual and speech perception by machine. To make effective use of these heuristics, a program must provide for

1. An external representation of heuristics for ease of man-machine communication
2. An internal representation of heuristics for effective use by machine
3. A mechanism for the selection of appropriate heuristics for use in a given situation.

Machine perception of vision and speech, thus, provides a problem domain for testing the adequacy of the models of representation (McCarthy and Hayes), planning heuristic selection (Minsky, Newell and Simon), and

generalization learning (Samuel); a domain in which (perceptual) tasks are performed by people easily and without effort.

AIM-79 AD685611
D. Raj. Reddy, Richard B. Neely,
Contextual Analysis of Phonemes of English,
71 pages, January 1969.

It is now well known that the acoustic characteristics of a Phoneme depend on both the preceding and following phonemes. This paper provides some needed contextual and probabilistic data about trigram phonemic sequences of spoken English. Since there are approximately 4013 such sequences, one must discover and study only the more commonly occurring sequences. To this purpose, three types of tables are presented, viz.,

a. Commonly occurring trigram sequences of the form /abc/ for every phoneme /b/.
b. Commonly occurring sequences /abc/ for every pair of phonemes /a/ and /c/.
c. Commonly occurring word boundary sequences of the form /-ab/ and /ab-/ where /-/ represents the silence phoneme.

Entries of the above tables contain examples of usage and probabilities of occurrence for each such sequence.

*AIM-80 AD685612
Georgia Sutherland,
Heuristic Dendral: a Family of LISP Programs,
46 pages, March 1969.

The Heuristic Dendral program for generating explanatory hypotheses in organic chemistry is described as an application of the programming language LISP. The description emphasizes the non-chemical aspects of the program, particularly the 'topologist' which generates all tree graphs of a collection of nodes.

*AIM-81 AD685613
David Luckham,
Refinement Theorems in Resolution Theory,
31 pages, March 1969.

The paper discusses some basic refinements of the Resolution Principle which are intended to improve the speed and efficiency of theorem-proving programs based on this rule of inference. It is proved thxt two of the refinements preserve the logical completeness of the proof procedure when used separately, but not when used in conjunction. The results of some preliminary experiments with the refinements are given.

Presented at the IRIA Symposium on Automatic Deduction, Versailles, France, December 16-21, 1968.

*AIM-82 AD685614
Zohar Manna, Amir Pneuli,
Formalization of Properties of Recursively Defined Functions,
26 pages, March 1969.

This paper is concerned with the relationship between the convergence, correctness and equivalence of recursively defined functions and the satisfiability (or unsatisfiability) of certain first-order formulas.

*AIM-83 CS-130
Roger C. Schank,
A Conceptual Representation for Computer-oriented Semantics,
Thesis: Ph.D. in Linguistics U. of Texas,
201 pages, March 1969.

Machines that may be said to function intelligently must be able to understand questions posed in natural language. Since natural language may be assumed to have an underlying conceptual structure, it is desirable to have the machine structure its own experience, both linguistic and nonlinguistic, in a manner concomitant with

the human method for doing so. Some previous attempts at organizing the machine's data base conceptually are discussed. A conceptually-oriented dependency grammar is posited as an interlingua that may be used as an abstract representation of the underlying conceptual structure. The conceptual dependencies are utilized as the highest level in a stratified system that incorporates language-specific realization rules to map from concepts and their relations, into sentences. In order to generate coherent sentences, a conceptual semantics is posited that limits possible conceptual dependencies to statements about the system's knowledge of the real world. This is done by the creation of semantic files that serve to spell out the defining characteristics of a given concept and enumerate the possibilities for relations with other concepts within the range of conceptual experience. The semantic files are created, in part, from a hierarchical organization of semantic categories. The semantic category is part of the definition of a concept and the information at the nodes dominating the semantic category in the hierarchical tree may be used to fill in the semantic file. It is possible to reverse the realization rules to operate on sentences and produce a conceptual parse. All potential parses are checked with the conceptual semantics in order to eliminate semantic and syntactic ambiguities. The system has been programmed; coherent sentences have been generated and the parser is operable. The entire system is posited as a viable linguistic theory.

*AIM-84 AD691791
David Canfield Smith,
MLISP Users' Manual,
57 pages, January 1969.

MLISP is a LISP pre-processor designed to facilitate the writing, use, and understanding of LISP progams. This is accomplished through parentheses reduction, comments, introduction of a more visual flow of control with block structure and mnemonic key words, and language redundancy. In addition, some 'meta-constructs' are introduced to increase the power of the language.

*AIM-85 CS-127 AD687720
Pierre Vicens,
Aspects of Speech Recognition by Computer,
Thesis: Ph.D. in Computer Science,
210 pages, April 1969.

This thesis describes techniques and methodology which are useful in achieving close to real-time recognition of speech by computer. To analyze connected speech utterances, any speech recognition system must perform the following processes: preprocessing, segmentation, segment classification, recognition of words, recognition of sentences. We present implemented solutions to each of these problems which achieved accurate recognition in all the trial cases.

*AIM-86 AD691788
Patrick J. Hayes,
A Machine-oriented Formulation of the Extended Functional Calculus,
44 pages, June 1969.

The Extended Functional Calculus (EFC), a three-valued predicate calculus intended as a language in which to reason about the results of computations, is described in some detail. A formal semantics is given. A machine-oriented (axiomless) inference system for EFC is then described and its completeness relative to the semantics is proved by the method of Semantic Trees. Finally some remarks are made on efficiency.

*AIM-87 AD691789
John McCarthy, A.I. Project Staff,
Project Technical Report,
98 pages, June 1969.

Plans and accomplishments of the Stanford Artificial Intelligence Project are reviewed in several areas including: theory (epistemology and mathematical theory of computation), visual perception and control (Hand-eye and Cart), speech recognition by computer, heuristics in machine learning and automatic deduction, models of cognitive processes (Heuristic DENDRAL), Language Research, and Higher Mental Functions. This is an excerpt of a proposal to ARPA.

*AIM-88 AD691790
Roger C. Schank,
Linguistics from a Conceptual Viewpoint (Aspects of Aspects of a Theory of Syntax),
22 pages, April 1969.

Some of the assertions made by Chomsky in Aspects of Syntax are considered. In particular, the notion of a 'competence' model in linguistics is criticized. Formal postulates for a conceptually-based linguistic theory are presented.

AIM-89 CS-125 AD692390
Jerome A. Feldman, J. Gips, J. J. Horning, and S. Reder,
Grammatical Complexity and Inference,
100 pages, June 1969.

The problem of inferring a grammar for a set of symbol strings is considered and a number of new decidability results obtained. Several notions of grammatical complexity and their properties are studied. The question of learning the least complex grammar for a set of strings is investigated leading to a variety of positive and negative results. This work is part of a continuing effort to study the problems of representation and generalization through the grammatical inference question.

*AIM-90 AD691799
Anthony C. Hearn,
Standard LISP,
33 pages, May 1969.

A uniform subset of LISP 1.5 capable of assembly under a wide range of existing compilers and interpreters is described.

AIM-91
J. A. Campbell and Anthony C. Hearn,
Symbolic Analysis of Feynman Diagrams by Computer,
73 pages, August 1969.

We describe a system of programs in the language LISP 1.5 which handles all stages of calculation from the specification of an elementary-particle process in terms of a Hamiltonian of interaction or Feynman diagrams to the derivation of an absolute square of the matrix element for the process. Examples of significant parts of the programs are presented in the text, while a detailed listing of this material is contained in two Appendices which are available on request from the authors.

*AIM-92
Victor D. Scheinman,
Design of a Computer Controlled Manipulator,
Thesis: Eng. in Mechanical Engineering,
53 pages, June 1969.

This thesis covers the preliminary system studies, the design process, and the design details associated with the design of a new computer controlled manipulator for the Stanford Artificial Intelligence Project. A systems study of various manipulator configurations, force sensing methods, and suitable components and hardware was first performed. Based on this study, a general design concept was formulated. This concept was then developed into a detailed manipulator design, having six degrees of freedom, all electric motor powered. The

manipulator has exceptionally high position accuracy, comparatively fast feedback servo performance, and approximately human arm reach and motion properties. Supporting some of the design details and selections are several examples of the design calculation procedure employed.

AIM-93.1 AD693106
Jerome Feldman,
Some Decidability Results on Grammatical Inference and Complexity,
31 pages, August 1969, revised May 1970.

The problem of grammatical inference is considered and a number of positive answers to decidability questions obtained. Conditions are prescribed under which it is possible for a machine to infer a grammar (or the best grammar) for even the general rewriting systems.

This revision was motivated by the discovery that our original definition of approachability was too weak and could be satisfied by trivial inference devices. Definition 1.2 and the surrounding material discuss this situation.

The theorems in Section 2 have been slightly reordered and new proofs given. The explicit use of a bounding function gives rise to an important new result, Corollary 2.4. Section 3 is changed primarily in the more detailed discussion of mixed strategy machines.

AIM-94 AD692391
Kenneth Mark Colby, Lawrence Tesler, Horace Enea,
Experiments With a Search Algorithm on the Data Base of a Human Belief Structure,
28 pages, August 1969.

Problems of collecting data regarding human beliefs are considered. Representation of this data in a computer model designed to judge credibility involved paraphrasings from natural language into the symbolic expressions of the programming language MLISP. Experiments in processing this data with a particular search algorithm are described, discussed and criticized.

*AIM-95 AD694971
Zohar Manna,
The Correctness of Non-deterministic Programs,
44 pages, August 1969.

In this paper we formalize properties of non-deterministic programs by means of the satisfiability and validity of formulas in first-order logic. Our main purpose is to emphasize the wide variety of possible applications of the results.

*AIM-96 CS-138 AD696394
Claude Cordell Green,
The Application of Theorem Proving to Question-answering Systems,
Thesis: Ph.D. in Electrical Engineering,
166 pages, August 1969.

This paper shows how a question-answering system can use first-order logic as its language and an automatic theorem prover based upon the resolution inference principle as its deductive mechanism. The resolution proof procedure is extended to a constructive proof procedure. An answer construction algorithm is given whereby the system is able not only to produce yes or no answers but also to find or construct an object satisfying a specified condition. A working computer program, QA3, based on these ideas, is described. The performance of the program, illustrated by extended examples, compares favorably with several other question-answering programs.

Methods are presented for solving state transformation problems. In addition to question-answering, the program can do automatic programming (simple program writing, program verifying, and debugging), control and problem solving for a simple

robot, pattern recognition (scene description), and puzzles.

AIM-97 AD694972
Kenneth Mark Colby, David Canfield Smith,
Dialogues Between Humans and an Artificial Belief System,
28 pages, August 1969.

An artificial belief system capable of conducting on-line dialogues with humans has bee. conntructed. It accepts information, answers questions and establishes a credibility for the information it acquires and for its human informants. Beginning with beliefs of high credibility from a highly believed source, the system is being subjected to the experience of dialogues with other humans.

AIM-98 CS-139 AD695401
James Jay Horning,
A Study of Grammatical Inference,
Thesis: Ph.D. in Computer Science,
166 pages, August 1969.

The present study has been motivated by the twin goals of devising useful inference procedures and of demonstrating a sound formal basis for such procedures. The former has led to the rejection of formally simple solutions involving restrictions which are unreasonable in practice; the latter, to the rejection of heuristic "bags of tricks" whose performance is in general imponderable. Part I states the general grammatical inference problem for formal languages, reviews previous work, establishes definitions and notation, and states my position for a particular class of grammatical inference problems based on an assumed probabilistic structure. The fundamental results are contained in Chapter V; the remaining chapters discuss extensions and removal of restrictions. Part III covers a variety of related topics, none of which are treated in any depth.

AIM-99
Bruce G. Buchanan, G. L. Sutherland, E. A. Feigenbaum,
Toward an Understanding of Information Processes of Scientific Inference in the Context of Organic Chemistry
66 pages, September 1969.

The program called Heuristic DENDRAL solves scientific induction problems of the following type: given the mass spectrum of an organic molecule, what is the most plausible hypothesis of organic structure that will serve to explain the given empirical data. Its problem solving power derives in large measure from the vast amount of chemical knowledge employed in controlling search and making evaluations.

A brief description of the task environment and the program is given in Part I. Recent improvements in the design of the program and the quality of its performance in the chemical task environment are noted.

The acquisition of task-specific knowledge from chemist-'experts', the representation of this knowledge in a form best suited to facilitate the problem solving, and the most effective deployment of this body of knowledge in restricting search and making selections have been major foci of our research. Part II discusses the techniques used and problems encountered in eliciting mass spectral theory from a cooperative chemist. A sample 'scenario' of a session with a chemist is exhibited. Part III discusses more general issues of the representation of the chemical knowledge and the design of processes tha. utilize it effectively. The initial, rather straight-forward, implementations were found to have serious defects. These are discussed. Part IV is concerned with our presently-conceived solutions to some of these problems, particularly the rigidity of processes and knowledge-structures.

The paper concludes with a bibliography of publications related to the DENDRAL effort.

AIM-100
Zohar Manna, John McCarthy,
Properties of Programs and Partial Function Logic,
21 pages, October 1969.

We consider recursive definitions which consist of Algol-like conditional expressions. By specifying a computation rule for evaluating such recursive definition, it determines a partial function. However, for different computation rules, the same recursive definition may determine different partial functions. We distinguish between two types of computation rules: sequential and parallel.

The purpose of this paper is to formalize properties (such as termination, correctness and equivlance) of these partial functions by means of the satisfiability or validity of certain formulas in partial function logic.

This paper was presented in the 5th Machine Intelligence Workshop held at Edinburgh (September 15-20, 1969), and will be published in *Machine Intelligence 5* (Edinburgh University Press, 1970).

*AIM-101
Richard Paul, G. Falk, J. A. Feldman,
The Computer Representation of Simply Described Scenes,
16 pages, October 1969.

This paper describes the computer representation of scenes consisting of a number of simple three-dimensional objects. One method of representing such scenes is a space oriented representation where information about a region of space is accessed by its coordinates. Another approach is to access the information by object, where, by giving the object name, its description and position are returned. As the description of an object is lengthy, it is desirable to group similar objects. Groups of similar objects can be represented in terms of a common part and a number of individual parts. If it is necessary to simulate moving an object then only the individual information need be saved.

*AIM-102
Donald A. Waterman,
Generalization Learning for Automating the Learning of Heuristics,
74 pages, July 1969.

This paper investigates the problem of implementing machine learning of heuristics. First, a method of representing heuristics as production rules is developed which facilitates dynamic manipulation of the heuristics by the program embodying them. Second, procedures are developed which permit a problem-solving program employing heuristics in production rule form to learn to improve its performance by evaluating and modifying existing heuristics and hypothesizing new ones, either during an explicit training process or during normal program operation. Third, the feasibility of these ideas in a complex problem-solving situation is demonstrated by using them in a program to make the bet decision in draw poker. Finally, problems which merit further investigation are discussed, including the problem of defining the task environment and the problem of adapting the system to board games.

AIM-103
John Allen, David Luckham
An Interactive Theorem-proving Program,
27 pages, October 1969.

We present an outline of the principle features of an on-line interative theorem-proving program, and a brief account of the results of some experiments with it. This program has been used to obtain proofs of new mathematical results recently announced

without proof in the Notices of the American Mathematical Society.

AIM-104
Joshua Lederberg, Georgia Sutherland, B. G. Buchanan, E. A Feigenbaum,
A Heuristic Program for Solving a Scientific Inference Problem: Summary of Motivation and Implementation,
15 pages, November 1969.

The primary motivation of the Heuristic DENDRAL project is to study and model processes of inductive inference in science, in particular, the formation of hypotheses which best explain given sets of empirical data. The task chosen for detailed study is organic molecular structure determination using mass spectral data and other associated spectra. This paper first summarizes the motivation and general outline of the approach. Next, a sketch is given of how the program works and how good its performance is at this stage. The paper concludes with a comprehensive list of publications of the project.

*AIM-105
Manfred Heuckel,
An Operator Which Locates Edges in Digitized Pictures,
37 pages, October 1969.

This paper reports the development of an edge finding operator (subroutine) which accepts the digitized light intensities within a small disc-shaped subarea of a picture and yields a description of any edge (brightness step) which may pass over the disc. In addition, the operator reports a measure of the edge's reliability. A theoretical effort disclosed the unique best operator which satisfies a certain set of criteria for a local edge recognizer. The main concerns of the criteria are speed and reliability in the presence of noise.

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

The time-optimal control of a system of rigid bodies connected in series by single-degree-of-freedom joints is studied.

The dynamical equations of the system are highly nonlinear and a closed-form representation of the minimum-time feedback control is not possible. However, a suboptimal feedback control which provides a close approximation to the optimal control is developed.

The suboptimal control is expressed in terms of switching curves for each of the system controls. These curves are obtained from the linearized equations of motion for the system. Approximations are made for the effects of gravity loads and angular velocity terms in the nonlinear equations of motion.

Digital simulation is used to obtain a comparison of response times of the optimal and suboptimal controls. The speed of response of the suboptimal control is found to compare quite favorably with the response speed of the optimal control.

The analysis is applied to the control of three joints of a mechanical manipulator. Modifications of the suboptimal control for use in a sampled-data system are shown to result in good performance of a hydraulic manipulator under computer control.

*AIM-107
Gilbert Falk,
Some Implications of Planarity for Machine Perception,
27 pages, December 1969.

The problem of determining the shape and

orientation of an object based on one or more two-dimensional images is considered. For a restricted class of projections it is shown that monocular information is often "nearly" sufficient for complete specification of the object viewed.

AIM-108
Michael D. Kelly,
Edge Detection in Pictures by Computer Using Planning,
28 pages, January 1970.

This paper describes a program for extracting an accurate outline of a man's head from a digital picture. The program accepts as input digital, grey scale pictures containing people standing in front of various backgrounds. The output of the program is an ordered list of the points which form the outline of the head. The edges of background objects and the interior details of the head have been suppressed.

The program is successful because of an improved method for edge detection which uses heuristic planning, a technique drawn from artificial intelligence research in problem solving. In brief, edge detection using planning consists of three steps. A new digital picture is prepared from the original; the new picture is smaller and has less detail. Edges of objects are located in the reduced picture. The edges found in the reduced picture are used as a plan for finding edges in the original picture.

*AIM-109
Roger C. Schank, Lawrence Tesler, Sylvia Weber,
Spinoza II: Conceptual Case-based Natural Language Analysis,
107 pages, January 1970

This paper presents the theoretical changes that have developed in Conceptual Dependency Theory and their ramifications in computer analysis of natural language. The major items of concern are: the elimination of reliance on 'grammar rules' for parsing with the emphasis given to conceptual rule based parsing, the development of a conceptual case system to account for the power of conceptualizations; the categorization of ACT's based on permissible conceptual cases and other criteria. These items are developed and discussed in the context of a more powerful conceptual parser and a theory of language understanding.

*AIM-110
Edward Ashcroft, Zohar Manna,
Formalization of Properties of Parallel Programs,
58 pages, February 1970.

In this paper we describe a class of parallel programs and give a formalization of certain properties of such programs in predicate calculus.

Although our programs are syntactically simple, they do exhibit interaction between asynchronous parallel processes, which is the essential feature we wish to consider. The formalization can easily be extended to more complicated programs.

Also presented is a method of simplifying parallel programs, i.e., constructing simpler equivalent programs, based on the "independence" of statements in them. With these simplications our formalization gives a practical method for proving properties of such programs.

*AIM-111
Zohar Manna,
Second-order Mathematical Theory of Computation,
25 pages, March 1970.

In this work we show that it is possible to formalize all properties regularly observed in (deterministic and non-deterministic) algorithms in second-order predicate calculus.

Moreover, we show that for any given algorithm it suffices to know how to formalize its "partial correctness" by a second-order formula in order to formalize all other properties by second-order formulas.

This result is of special interest since "partial correctness" has already been formalized in second-order predicate calculus for many classes of algorithms.

This paper will be presented at the ACM Symposium on Theory of Computing (May 1970).

AIM-112
Franklin D. Hilf, Kenneth M. Colby, David C. Smith, William K. Wittner,
Machine-mediated Interviewing,
27 pages, March 1970.

A technique of psychiatric interviewing is described in which patient and interviewer communicate by means of remotely located teletypes. Advantages of non-nonverbal communication in the study of the psychiatric interview and in the development of a computer program designed to conduct psychiatric interviews are discussed. Transcripts from representative interviews are reproduced.

*AIM-113
Kenneth Mark Colby, Franklin D. Hilf, William A. Hall,
A Mute Patient's Experience With Machine-mediated Interviewing,
19 pages, March 1970.

A hospitalized mute patient participated in seven machine-mediated interviews, excerpts of which are presented. After the fifth interview he began to use spoken language for communication. This novel technique is suggested for patients who are unable to participate in the usual vis-a-vis interview.

*AIM-114
Alan W. Biermann, Jerome A. Feldman,
On the Synthesis of Finite-state Acceptors,
31 pages, April 1970.

Two algorithms are presented for solving the following problem: Given a finite-set S of strings of symbols, find a finite-state machine which will accept the strings of S and possibly some additional strings which "resemble" those of S. The approach used is to directly construct the states and transitions of the acceptor machine from the string information. The algorithms include a parameter which enable one to increase the exactness of the resulting machine's behavior as much as desired by increasing the number of states in the machine. The properties of the algorithms are presented and illustrated with a number of examples.

The paper gives a method for identifying a finite-state language from a randomly chosen finite subset of the language if the subset is large enough and if a bound is known on the number of states required to recognize the language. Finally, we discuss some of the uses of the algorithms and their relationship to the problem of grammatical inference.

AIM-115
Ugo Montanari,
On the Optimal Detection of Curves in Noisy Pictures,
35 pages, March 1970.

A technique for recognizing systems of lines is presented, in which the heuristic of the problem is not embedded in the recognition algorithm but is expressed in a figure of merit. A multistage decision process is then able to recognize in the input picture the optimal system of lines according to the given figure of merit. Due to the global approach, greater flexibility and adequacy in the particular problem is achieved. The relation between the structure of the figure of merit and the complexity of the optimization

process is then discussed. The method described is suitable for parallel processing because the operations relative to each state can be computed in parallel, and the number of stages is equal to the length N of the curves (or to log2(N) if an approximate method is used).

*AIM-116
Kenneth Mark Colby,
Mind and Brain, Again,
10 pages, March 1970.

Classical mind-brain questions appear deviant through the lens of an analogy comparing mental processes with computational processes. Problems of reducibility and personal consciousness are also considered in the light of this analogy.

*AIM-117
John McCarthy, et al,
Project Technical Report,
75 pages, April 1970.

Current research is reviewed in artificial intelligence and related areas, including representation theory, mathematical theory of computation, models of cognitive processes, speech recognition, and computer vision.

AIM-118
Ugo Montanari,
Heuristically Guided Search and Chromosome Matching,
29 pages, April 1970.

Heuristically guided search is a technique which takes systematically into account information from the problem domain for directing the search. The problem is to find the shortest path in a weighted graph from a start vertex Va to a goal vertex Vz: for every intermediate vertex, an estimate is available of the distance to Vz. If this estimate satisfies a consistency assumption, an algorithm by Hart, Nilsson and Raphael is guaranteed to find the optimum, looking at the *a priori* minimum number of vertices. In this paper, a version of the above algorithm is presented, which is guaranteed to succeed with the minimum amount of storage. An application of this technique to the chromosome matching problem is then shown. Matching is the last stage of automatic chromosome analysis procedures, and can also solve ambiguities in the classification stage. Some peculiarities of this kind of data suggest the use of an heuristically guided search algorithm instead of the standard Edmonds' algorithm. The method that we obtain in this way is proved to exploit the clustering of chromosome data: a linear-quadratic dependence from the number of chromosomes is obtained for perfectly clustered data. Finally, some experimental results are given.

*AIM-119
Joseph Becker,
An Information-processing Model of Intermediate-Level Cognition,
123 pages, May 1970.

There is a large class of cognitive operations in which an organism adapts its previous experience in order to respond properly to a new situation -- for example: the perceptual recognition of objects and events, the prediction of the immediate future (e.g. in tracking a moving object), and the employment of sensory-motor "skills". Taken all together, these highly efficient processes form a cognitive subsystem which is intermediate between the low-level sensory-motor operations and the more deliberate processes of high-level 'thought'.

The present report describes a formal information-processing model of this 'Intermediate-Level' cognitive system. The model includes memory structures for the storage of experience, and processes for responding to new events on the basis of previous experience. In addition, the proposed system contains a large number of

mechanisms for making the response-selection process highly efficient, in spite of the vast amount of stored information that the system must cope with. These devices include procedures for heuristically evaluating alternative subprocesses, for guiding the search through memory, and for reorganizing the information in memory into more efficient representations.

*AIM-120
Kenneth Mark Colby, David Canfield Smith,
Computer as Catalyst in the Treatment of Nonspeaking Autistic Children,
32 pages, April 1970.

Continued experience with a computer-aided treatment method for nonspeaking autistic children has demonstrated improvement effects in thirteen out of a series of seventeen cases. Justification for this conclusion is discussed in detail. Adoption of this method by other research groups is needed for the future development of computer-aided treatment.

*AIM-121
Irwin Sobel,
Camera Models and Machine Perception,
Thesis: Ph.D. in Electrical Engineering,
89 pages, May 1970.

We have developed a parametric model for a computer-controlled moveable camera on a pan-tilt head. The model expresses the transform relating object space to image space as a function of the control variables of the camera. We constructed a calibration system for measuring the model parameters which has a demonstrated accuracy more than adequate for our present needs. We have also identified the major source of error in model measurement to be undesired image motion and have developed means of measuring and compensating for some of it and eliminating other parts of it. The system can measure systematic image distortions if they become the major accuracy limitation. We have shown how to generalize the model to handle small systematic errors due to aspects of pan-tilt head geometry not presently accounted for.

We have demonstrated the model's application in stereo vision and have shown how it can be applied as a predictive device in locating objects of interest and centering them in an image.

*AIM-122
Roger C. Schank,
'Semantics' in Conceptual Analysis,
56 pages, May 1970.

This paper examines the question of what a semantic theory should account for. Some aspects of the work of Katz, Fillmore, Lakoff and Chomsky are discussed. *Semantics* is concluded to be the representation problem with respect to conceptual analysis. The beginnings of a solution to this problem are presented in the light of developments in conceptual dependency theory.

AIM-123
Bruce G. Buchanan, Thomas E. Headrick,
Some Speculation About Artificial Intelligence and Legal Reasoning,
54 pages, May 1970.

Legal reasoning is viewed here as a complex problem-solving task to which the techniques of artificial intelligence programming may be applied. Some existing programs are discussed which successfully attack various aspects of the problem, in this and other task domains. It remains an open question, to be answered by intensive research, whether computers can be programmed to do creative legal reasoning. Regardless of the answer, it is argued that much will be gained by the research.

AIM-124
M. M. Astrahan,
Speech Analysis by Clustering, or the Hyperphoneme Method,
22 pages, June 1970.

In this work, measured speech waveform data was used as a basis for partitioning an utterance into segments and for classifying those segments. Mathematical classifications were used instead of the traditional phonemes or linguistic categories. This involved clustering methods applied to hyperspace points representing periodic samples of speech waveforms. The cluster centers, or hyperphonemes (HPs), were used to classify the sample points by the nearest-neighbor technique. Speech segments were formed by grouping adjacent points with the same classification. A dictionary of 54 different words from a single speaker was processed by this method. 216 utterances, representing four more repetitions by the same speaker each of the original 54 words, were similarly analyzed into strings of hyperphonemes and matched against the dictionary by heuristically developed formulas. 87% were correctly recognized, although almost no attempt was made to modify and improve the initial methods and parameters.

*AIM-125
Kenneth Mark Colby, Sylvia Weber,
Franklin Hilf,
Artificial Paranoia,
35 pages, July 1970.

A case of artificial paranoia has been synthesized in the form of a computer model. Using the test operations of a teletyped psychiatric interview, clinicians judge the input-output behavior of the model to be paranoid. Formal validation of the model will require experiments involving indistinguishability tests.

AIM-126 CS-169 AD711329
Donald E. Knuth,
Examples of Formal Semantics,
34 pages, July 1970.

A technique of formal definition, based on relations between 'attributes' associated with nonterminal symbols in a context-free grammar, is illustrated by several applications to simple yet typical problems. First we define the basic properties of lambda expressions, involving substitution and renaming of bound variables. Then a simple programming language is defined using several different points of view. The emphasis is on 'declarative' rather than 'imperative' forms of definition.

AIM-127 CS-174 AD711395
Zohar Manna, Richard J. Waldinger,
Towards Automatic Program Synthesis,
54 pages, July 1970.

An elementary outline of the theorem-proving approach to automatic program synthesis is given, without dwelling on technical details. The method is illustrated by the automatic construction of both recursive and iterative programs operating on natural numbers, lists, and trees.

In order to construct a program satisfying certain specifications, a theorem induced by those specifications is proved, and the desired program is extracted from the proof. The same technique is applied to transform recursively defined functions into iterative programs, frequently with a major gain in efficiency.

It is emphasized that in order to construct a program with loops or with recursion, the principle of mathematical induction must be applied. The relation between the version of the induction rule used and the form of the program constructed is explored in some detail.

AIM-128 CS-166 AD713841
Erik J. Sandewall,
Representing Natural-language Information in Predicate Calculus,
27 pages, July 1970.

A set of general conventions are proposed for representing natural language information in many-sorted first order predicate calculus. The purpose is to provide a testing-ground for existing theorem-proving programs.

*AIM-129 CS-167 AD712460
Shigeru Igarashi,
Semantics of ALGOL-like Statements,
95 pages, June 1970.

The semantics of elementary Algol-like statements is discussed, mainly based on an axiomatic method.

Firstly, a class of Algol-like statements is introduced by generalized inductive definition, and the interpretation of the statements belonging to it is defined in the form of a function over this class, using the induction principle induced by the above definition. Then a category of program is introduced in order to clarify the concept of equivalence of statements, which becomes a special case of isomorphism in that category.

A revised formal system representing the concept of equivalence of Algol-like statements is presented, followed by elementary metatheorems.

Finally, a process of decomposition of Algol-like statements, which can be regarded as a conceptual compiler, or a constructive description of semantics based on primitive actions, is defined and its correctness is proved formally, by the help of the induced induction principle.

AIM-130 CS-168 AD713252
Michael D. Kelly,
Visual Identification of People by Computer,
Thesis: Ph.D. in Computer Science,
238 pages, July 1970.

This thesis describes a computer program which performs a complex picture processing task. The task is to choose, from a collection of pictures of people taken by a TV camera, those pictures that depict the same person. The primary purpose of this research has been directed toward the development of new techniques for picture processing.

In brief, the program works by finding the location of features such as eyes, nose, or shoulders in the pictures. Individuals are classified by measurements between such features. The interesting and difficult part of the work reported in this thesis is the detection of those features in digital pictures. The nearest neighbor method is used for identification of individuals once a set of measurements has been obtained.

The success of the program is due to and illustrates the heuristic use of context and structure. A new, widely useful, technique called planning has been applied to picture processing. Planning is a term which is drawn from artificial intelligence research in problem solving.

The principal positive result of this research is the use of goal-directed techniques to successfully locate features in cluttered digital pictures. This success has been verified by displaying the results of the feature finding algorithms and comparing these locations with the locations obtained by hand from digital printouts of the pictures. Successful performance in the task of identification of people provides further verification for the feature finding algorithms.

AIM-131 CS-176 AD715128
Edward A. Feigenbaum, Bruce G. Buchanan, Joshua Lederberg,
On Generality and Problem Solving: a Case Study Using the Dendral Program,
48 pages, August 1970.

Heuristic DENDRAL is a computer program written to solve problems of inductive inference in organic chemistry. This paper will use the design of Heuristic DENDRAL and its performance on different problems for a discussion of the following topics:

1. the design for generality;
2. the performance problems attendent upon too much generality
3. the coupling of expertise to the general problem solving processes,
4. the symbiotic relationship between generality and expertnness of problem solving systems.

We conclude the paper with a view of the design for a general problem solver that is a variant of the "big switch" theory of generality.

*AIM-132 CS-180 AD715665
Gilbert Falk,
Computer Interpretation of Imperfect Line Data as a Three-dimensional Scene,
Thesis: Ph.D. in Electrical Engineering,
187 pages, August 1970.

The major portion of this paper describes a heuristic scene description program. This program accepts as input a scene represented as a line drawing. Based on a set of known object models the program attempts to determine the identity and location of each object viewed. The most significant feature of the program is its ability to deal with imperfect input data.

We also present some preliminary results concerning constraints in projections of planar-faced solids. We show that for a restricted class of projects, 4 points located in 3-space in addition to complete monocular information are sufficient to specify all the visible point locations precisely.

*AIM-133 CS-181
Anthony C. Hearn,
Reduce 2,
Diskfile: REDUCE ACH[AIM,DOC],
85 pages, October 1970.

This manual provides the user with a description of the algebraic programming system REDUCE 2. The capabilities of this system include:

1) Expansion and ordering of rational functions of polynomials,
2) symbolic differentiation of rational functions of polynomials and general functions,
3) substitutions and pattern matching in a wide variety of forms,
4) calculation of the greatest common divisor of two polynomials,
5) automatic and user controlled simplification of expressions,
6) calculations with symbolic matrices,
7) a complete language for symbolic calculations, in which the REDUCE program itself is written,
8) calculations of interest to high energy physicists including spin 1/2 and spin 1 algebra,
9) tensor operations.

*AIM-134 CS-182 AD748565
Jay Martin Tenenbaum,
Accommodation in Computer Vision,
Thesis: Ph.D. in Electrical Engineering,
452 pages, September 1970.

We describe an evolving computer vision system in which the parameters of the camera are controlled by the computer. It is distinguished from conventional picture processing systems by the fact that sensor accommodation is automatic and treated as an integral part of the recognition process.

A machine, like a person, comes in contact with far more visual information than it can process. Furthermore, no physical sensor can simultaneously provide information about the full range of the environment. Consequently, both man and machine must accommodate their sensors to emphasize selected characteristics of the environment.

Accommodation improves the reliability and efficiency of machine perception by matching the information provided by the sensor with that required by specific perceptual functions. The advantages of accommodation are demonstrated in the context of five key functions in computer vision: acquisition, contour following, verifying the presence of an expected edge, range-finding, and color recognition.

We have modeled the interaction of camera parameters with scene characteristics to determine the composition of an image. Using a priori knowledge of the environment, the camera is tuned to satisfy the information requirements of a particular task.

Task performance depends implicitly on the appropriateness of available information. If a function fails to perform as expected, and if this failure is attributable to a specific image deficiency, then the relevant accommodation parameters can be refined.

This schema for automating sensor accommodation can be applied in a variety of perceptual domains.

AIM-135 CS-179 AD716566
David Canfield Smith,
MLISP,
Diskfile: MLISP.DAV[AIM,DOC]
99 pages, October 1970.

MLISP is a high level list-processing and symbol-manipulation language based on the programming language LISP. MLISP programs are translated into LISP programs and then executed or compiled. MLISP exists for two purposes: (1) to facilitate the writing and understanding of LISP programs; (2) to remedy certain important deficiencies in the list-processing ability of LISP.

*AIM-136 CS-183 AD717600
George M. White,
Machine Learning Through Signature Trees. Applications to Human Speech,
40 pages, October 1970.

Signature tree 'machine learning', pattern recognition heuristics are investigated for the specific problem of computer recognition of human speech. When the data base of given utterances is insufficient to establish trends with confidence, a large number of feature extractors must be employed and 'recognition' of an unknown pattern made by comparing its feature values with those of known patterns. When the data base is replete, a 'signature' tree can be constructed and recognition can be achieved by the evaluation of a select few features. Learning results from selecting an optimal minimal set of features to achieve recognition. Properties of signature trees and the heuristics for this type of learning are of primary interest in this exposition.

*AIM-137
Donald E. Knuth,
An Empirical Study of Fortran in Use,
44 pages, November 1970.

A sample of programs, written in Fortran by a wide variety of people for a wide variety of applications, was chosen 'at random' in an attempt to discover quantitatively 'what programmers really do'. Statistical results of this survey are presented here, together with some of their apparent implications for future work in compiler design. The principle conclusion which may be drawn is the importance of a program 'profile', namely

a table of frequency counts which record how often each statement is performed in a typical run: there are strong indications that profile-keeping should become a standard practice in all computer systems, for casual users as well as system programmers. Some new approaches to compiler optimization are also suggested. This paper is the report of a three month study undertaken by the author and about a dozen students and representatives of the software industry during the summer of 1970.

AIM-138 CS-188 PB197161
Edward Ashcroft, Zohar Manna,
The Translation of 'GO-TO' Programs to 'WHILE' Programs,
28 pages, January 1971.

In this paper we show that every flowchart program can be written without go-to statements by using while statements. The main idea is to introduce new variables to preserve the values of certain variables at particular points in the program; or alternatively, to introduce special boolean variables to keep information about the course of the computation.

The while programs produced yield the same final results as the original flowchart program but need not perform computations in exactly the same way. However, the new programs preserve the *topology* of the original flowchart program, and are of the same order of efficiency.

We also show that this cannot be done in general without adding variables.

AIM-139 CS-189 AD717601
Zohar Manna,
Mathematical Theory of Partial Correctness,
24 pages, January 1971.

In this work we show that it is possible to express most properties regularly observed in algorithms in terms of *partial correctness* (i.e., the property that the final results of the algorithm, if any, satisfy some given input-output relation).

This result is of special interest since *partial correctness* has already been formulated in predicate calculus and in partial function logic for many classes of algorithms.

*AIM-140 CS-193
Roger C. Schank,
Intention, Memory, and Computer Understanding,
59 pages, January 1971.

Procedures are described for discovering the intention of a speaker by relating the Conceptual Dependence representation of the speaker's utterance to the computer's world model such that simple implications can be made. These procedures function at levels higher than that of structure of the memory. Computer understanding of natural language is shown to consist of the following parts: assigning a conceptual representation to an input; relating that representation to the memory such as to extract the intention of the speaker; and selecting the correct response type triggered by such an utterance according to the situation.

*AIM-141 CS-203 AD730506
Bruce G. Buchanan, Joshua Lederberg,
The Heuristic DENDRAL Program for Explaining Empirical Data,
20 pages, February 1971.

The Heuristic DENDRAL program uses an information processing model of scientific reasoning to explain experimental data in organic chemistry. This report summarizes the organization and results of the program for computer scientists. The program is divided into three main parts: planning, structure generation, and evaluation.

The planning phase infers constraints on the

search space from the empirical data input to the system. The structure generation phase searches a tree whose termini are models of chemical models using pruning heuristics of various kinds. The evaluation phase tests the candidate structures against the original data. Results of the program's analyses of some tests are discussed.

AIM-142 CS-205 AD731383
Robin Milner,
An Algebraic Definition of Simulation Between Programs,
21 pages, February 1971.

A simulation relation between programs is defined which is quasi-ordering. Mutual simulation is then an equivalence relation, and by dividing out by it we abstract from a program such details as how the sequencing is controlled and how data is represented. The equivalence classes are approximations to the algorithms which are realized, or expressed, by their member programs.

A technique is given and illustrated for proving simulation and equivalence of programs; there is an analogy with Floyd's technique for proving correctness of programs. Finally, necessary and sufficient conditions for simulation are given.

AIM-143 CS-209 AD724867
John McCarthy, et al,
Project Technical Report,
80 pages, March 1971.

An overview is presented of current research at Stanford in artificial intelligence and heuristic programming. This report is largely the text of a proposal to the Advanced Research Projects Agency for fiscal years 1972-3.

AIM-144 CS 219
Lynn H. Quam,
Computer Comparison of Pictures,
Thesis: Ph.D. in Computer Science,
120 pages, May 1971.

This dissertation reports the development of digital computer techniques for detecting changes in scenes by normalizing and comparing pictures which were taken from different camera positions and under different conditions of illumination. The pictures are first geometrically normalized to a common point of view. Then they are photometrically normalized to eliminate the differences due to different illumination, camera characteristics, and reflectance properties of the scene due to different sun and view angles. These pictures are then geometrically registered by maximizing the cross correlation between areas in them. The final normalized and registered pictures are then differenced point by point.

The geometric normalization techniques require relatively accurate geometric models for the camera and the scene, and static spatial features must be present in the pictures to allow precise geometric alignment using the technique of cross correlation maximization.

Photometric normalization also requires a relatively accurate model for the photometric response of the camera, a reflectance model for the scene (reflectance as a function of the illumination view, and phase angles) and some assumptions about the kinds of reflectance changes which are to be detected.

These techniques have been incorporated in a system for comparing Mariner 1971 pictures of Mars to detect variable surface phenomena as well as color and polarization differences. The system has been tested using Mariner 6 and 7 pictures of Mars.

Although the techniques described in this

dissertation were developed for Mars pictures, their use is not limited to this application. Various parts of this software package, which was developed for interactive use on the time-sharing system of the Stanford Artificial Intelligence Project, are currently being applied to other scenery.

*AIM-145 CS-221 AD731729
Bruce G. Buchanan, Edward A. Feigenbaum, Joshua Lederberg,
A Heuristic Programming Study of Theory Formation in Science,
41 pages, June 1971.

The Meta-DENDRAL program is a a vehicle for studying problems of theory formation in science. The general strategy of Meta-DENDRAL is to reason from data to plausible generalizations and then to organize the generalizations into a unified theory. Three main subproblems are discussed: (1) explain the experimental data for each individual chemical structure, (2) generalize the results from each structure to all structures, and (3) organize the generalizations into a unified theory. The program is built upon the concepts and programmed routines already available in the Heuristic DENDRAL performance program, but goes beyond the performance program in attempting to formulate the theory which the performance program will use.

AIM-146 CS-224 PB212183
Andrei P. Ershov,
Parallel Programming,
14 pages, July 1971.

This report is based on lectures given at Stanford University by Dr. Ershov in November, 1970.

*AIM-147 CS-216 AD732457
Robert E. Kling,
Reasoning by Analogy with Applications to Heuristic Problem Solving: a Case Study,
Thesis: Ph.D. in Computer Science,
191 pages, August 1971.

An information-processing approach to reasoning by analogy is developed that promises to increase the efficiency of heuristic deductive problem-solving systems. When a deductive problem-solving system accesses a large set of axioms more than sufficient for a particular problem, it will often create many irrelevant deductions that saturate the memory of the problem solver.

Here, an analogy with some previously solved problem and a new unsolved problem is used to restrict the data base to a small set of appropriate axioms. This procedure (ZORBA) is studied in detail for a resolution theorem proving system. A set of algorithms (ZORBA-I) which automatically generates an analogy between a new unproved theorem, T↓A, and a previously proved theorem, T, is described in detail. ZORBA-I is implemented in LISP on a PDP-10.

ZORBA-I is examined in terms of its empirical performance on parts of analogous theorems drawn from abstract algebra. Analytical studies are included which show that ZORBA-I can be useful to aid automatic theorem proving in many pragmatic cases while it may be a detriment in certain specially contrived cases.

AIM-148 CS-217 AD731730
Edward Ashcroft, Zohar Manna, Amir Pneuli,
Decidable Properties of Monadic Functional Schemas,
10 pages, July 1971.

We define a class of (monadic) functional schemas which properly include 'Ianov' flowchart schemas. We show that the

termination, divergence and freedom problems for functional schemas are decidable. Although it is possible to translate a large class of non-free functional schemas into equivalent free functional schemas, we show that this cannot be done in general. We show also that the equivalence problem for free functional schemas is decidable. Most of the results are obtained from well-known results in Formal Languages and Automata Theory.

AIM-149 CS-231 AD732644
Rodney Albert Schmidt, Jr.,
A Study of the Real-time Control of a Computer-driven Vehicle,
Thesis: Ph.D. in Electrical Engineering,
180 pages, August 1971.

Vehicle control by the computer analysis of visual images is investigated. The areas of guidance, navigation, and incident avoidance are considered. A television camera is used as the prime source of visual image data.

In the guidance system developed for an experimental vehicle, visual data is used to gain information about the vehicle system dynamics, as well as to guide the vehicle. This information is used in real time to improve performance of the non-linear, time-varying vehicle system.

A scheme for navigation by pilotage through the recognition of two dimensional scenes is developed. A method is proposed to link this to a computer-modelled map in order to make journeys.

Various difficulties in avoiding anomolous incidents in the automatic control of automobiles are discussed, together with suggestions for the application of this study to remote exploration vehicles or industrial automation.

*AIM-150
Robert W. Floyd,
Toward Interactive Design of Correct Programs,
12 pages, September 1971.

We propose an interactive system proving the correctness of a program, or locating errors, as the program is designed.

*AIM-151 CS-240 AD738568
Ralph L. London,
Correctness of Two Compilers for a LISP Subset,
41 pages, October 1971.

Using mainly structural induction, proofs of correctness of each of two running Lisp compilers for the PDP-10 computer are given. Included are the rationale for presenting these proofs, a discussion of the proofs, and the changes needed to the second compiler to complete its proof.

*AIM-152 CS-241 AD732642
Alan W. Biermann,
On the Inference of Turing Machines from Sample Computations,
31 pages, October 1971.

An algorithm is presented which when given a complete description of a set of Turing machine computations finds a Turing machine which is capable of doing those computations. This algorithm can serve as the basis for designing a trainable device which can be trained to simulate any Turing machine by being led through a series of sample computations done by that machine. A number of examples illustrate the use of the technique and the possibility of the application to other types of problems.

*AIM-153 CS-242 AD738569
Patrick J. Hayes,
The Frame Problem and Related Problems in Artificial Intelligence,
18 pages, November 1971.

The frame problem arises in considering the logical structure of a robot's beliefs. It has been known for some years, but only recently has much progress been made. The problem is described and discussed. Various suggested methods for its solution are outlined, and described in a uniform notation. Finally, brief consideration is given to the problem of adjusting a belief system in the face of evidence which contradicts beliefs. It is shown that a variation on the situation notation of (McCarthy and Hayes, 1969) permits an elegant approach, and relates this problem to the frame problem.

*AIM-154 CS-243 AD738570
Zohar Manna, Stephen Ness, Jean Vuillemin,
Inductive Methods for Proving Properties of Programs,
24 pages, November 1971.

We have two main purposes in this paper. First, we clarify and extend known results about computation of recursive programs, emphasizing the difference between the theoretical and practical approaches. Secondly, we present and examine various known methods for proving properties of recursive programs. We discuss in detail two powerful inductive methods, computational induction and structural induction, illustrating their applications by various examples. We also briefly discuss some other related methods.

Our aim in this work is to introduce inductive methods to as wide a class of readers as possible and to demonstrate their power as practical techniques. We ask the forgiveness of our more theoretical-minded colleagues for our occasional choice of clarity over precision.

AIM-155 CS-245
Jonathan Leonard Ryder,
Heuristic Analysis of Large Trees as Generated in the Game of Go,
Thesis: Ph.D. in Computer Science,
300 pages, December 1971.

The Japanese game of Go is of interest both as a problem in mathematical repesentation and as a game which generates a move tree with an extraordinarily high branching factor (100 to 300 branches per ply). The complexity of Go (and the difficulty of Go for human players) is thought to be considerably greater than that of chess. The constraints of being able to play a complete game and of being able to produce a move with a moderate amount of processing time were placed on the solution.

The basic approach used was to find methods for isolating and exploring several sorts of relevant subsections of the global game tree. This process depended heavily on the ability to define and manipulate the entities of Go as recursive functions rather than as patterns of stones. A general machine-accessible theory of Go was developed to provide context for program evaluations.

A program for playing Go is now available on the Stanford PDP-10 computer. It will play a complete game, taking about 10 to 30 seconds for an average move. The quality of play is better than that of a beginner in many respects, but incompletenesses in the current machine-representable theory of Go prevent the present program from becoming a strong player.

AIM-156 CS-246 AD740141
Kenneth Mark Colby, Franklin D. Hilf,
Sylvia Weber, Helena C. Kraemer,
A Resemblance Test for the Validation of a Computer Simulation of Paranoid Processes,
29 pages, November 1971.

A computer simulation of paranoid processes in the form of a dialogue algorithm was subjected to a validation study using an experimental resemblance test in which judges rate degrees of paranoia present in initial psychiatric interviews of both

paranoid patients and of versions of the paranoid model. The statistical results indicate a satisfactory degree of resemblance between the two groups of interviews. It is concluded that the model provides a successful simulation of naturally occuring paranoid processes.

AIM-157 CS-247
Yorick Wilks,
One Small Head -- Some Remarks on the use of 'Model' in Linguistics,
17 pages, December 1971.

I argue that the present situation in formal linguistics, where much new work is presented as being a "model of the brain", or of "human language behavior", is an undesirable one. My reason for this judgement is not the conservative (Braithwaitian) one that the entities in question are not really models but theories. It is rather that they are called models because they cannot be theories of the brain at the present stage of brain research, and hence that the use of "model" in this context is not so much aspirational as resigned about our total ignorance of how the brain stores and processes linguistic information. The reason such explanatory entities cannot be theories is that this ignorance precludes any "semantic ascent" up the theory; i.e., interpreting the items of the theory in terms of observables. And the brain items, whatever they may be, are not, as Chomsky has sometimes claimed, in the same position as the "occult entities" of Physics like Gravitation; for the brain items are not theoretically unreachable, merely unreached.

I then examine two possible alternate views of what linguistic theories should be proffered as theories of: theories of sets of sentences, and theories of a particular class of algorithms. I argue for a form of the latter view, and that its acceptance would also have the effect of making Computational Linguistics a central part of Linguistics, rather than the poor relation it is now.

I examine a distinction among 'linguistic models' proposed recently by Mey, who was also arguing for the self-sufficiency of Computational Linguistics, though as a 'theory of performance'. I argue that his distinction is a bad one, partly for the reasons developed above and partly because he attempts to tie it to Chomsky's inscrutable competance-performance distinction. I conclude that the independence and self-sufficiency of Computational Linguistics are better supported by the arguments of this paper.

AIM-158 CS-250 AD740127
Ashok Chandra, Zohar Manna,
Program Schemas With Equality,
13 pages, December 1971.

We discuss the class of program schemas augmented with equality tests, that is, tests of equality between terms.

In the first part of the paper we illustrate the 'power' of equality tests. It turns out that the class of program schemas with equality is more powerful than the 'maximal' classes of schemas suggested by other investigators.

In the second part of the paper, we discuss the decision problems of program schemas with equality. It is shown, for example, that while the decision problems normally considered for schemas (such as halting, divergence, equivalence, isomorphism and freedom) are decidable for Ianov schemas. They all become undecidable if general equality tests are added. We suggest, however, limited equality tests which can be added to certain subclasses of program schemas while preserving their solvable properties.

AIM-159 CS-253
Jerome A. Feldman, Paul C. Shields,
Total Complexity and Inference of Best Programs,
40 pages, April 1972.

Axioms for a total complexity measure for abstract programs are presented. Essentially, they require that total complexity be an unbounded increasing function of the Blum time and size measures. Algorithms for finding the best program on a finite domain are presented, and their limiting behavior for infinite domains described. For total complexity, there are important senses in which a machine can find the best program for a large class of functions.

AIM-160 CS-255 AD740140
Jerome A. Feldman,
Automatic Programming,
20 pages, February 1972.

The revival of interest in Automatic Programming is considered. The research is divided into direct efforts and theoretical developments and the successes and prospects of each are described.

AIM-161 CS-264 AD741189
Yorick Wilks,
Artificial Intelligence approach to Machine Translation,
44 pages, February 1972.

The paper describes a system of semantic analysis and generation, programmed in LISP 1.5 and designed to pass from paragraph length input in English to French via an interlingual representation. A wide class of English input forms will be covered, but the vocabulary will initially be restricted to one of a few hundred words. With this subset working, and during the current year (1971-72), it is also hoped to map the interlingual representation onto some predicate calculus notation so as to make possible the answering of very simple questions about the translated matter. The specification of the translation system itself is complete, and its main points are:

i) It translates phrase by phrase--with facilities for reordering phrases and establishing essential semantic connectivities between them--by mapping complex semantic stuctures of "message" onto each phrase. These constitute the interlingual representation to be translated. This matching is done without the explicit use of a conventional syntax analysis, by taking as the appropriate matched structure the 'most dense' of the alternative structures derived. This method has been found highly successful in earlier versions of this analysis system.

ii) The French output strings are generated without the explicit use of a generative grammar. That is done by means of *stereotypes*: strings of French words, and functions evaluating to French words, which are attached to English word senses in the dictionary and built into the interlingual representation by the analysis routines. The generation program thus receives an interlingual representation that already contains both French output and implicit procedures for assembling the output, since the stereotypes are in effect recursive procedures specifying the content and production of the output word strings. Thus the generation program at no time consults a word dictionary or inventory of grammar rules.

It is claimed that the system of notation and translation described is a convenient one for expressing and handling the items of semantic information that are *essential* to any effective MT system. I discuss in some detail the semantic information needed to ensure the correct choice of output prepositions in French; a vital matter inadequately treated by virtually all previous formalisms and projects.

*AIM-162 CS-265 AD744634
Roger C. Schank, N. Goldman, C. J. Rieger, C. K. Riesbeck,
Primitive Concepts Underlying Verbs of Thought,
102 pages, April 1972.

In order to create conceptual structures that will uniquely and unambiguously represent the meaning of an utterance, it is necessary to establish 'primitive' underlying actions and states into which verbs can be mapped. This paper presents analyses of the most common mental verbs in terms of such primitive actions and states. In order to represent the way people speak about their mental processes, it was necessary to add to the usual ideas of memory structure the notion of Immediate Memory. It is then argued that there are only three primitive mental ACTs.

AIM-163 CS-266
Jean M. Cadiou,
Recursive Definitions of Partial Functions and Their Computations,
Thesis: Ph.D. in Computer Science,
160 pages, April 1972.

A formal syntactic and semantic model is presented for 'recursive definitions' which are generalizations of those found in LISP, for example. Such recursive definitions can have two classes of fixpoints, the strong fixpoints and the weak fixpoints, and also possess a class of computed partial functions.

Relations between these classes are presented: fixpoints are shown to be extensions of computed functions. More precisely, strong fixpoints are shown to be extensions of computed functions when the computations may involve 'call by name' substitutions; weak fixpoints are shown to be extensions of computed functions when the computation only involve 'call by value' substitutions. The Church-Rosser property for recursive definitions with fixpoints also follows from these results.

Then conditions are given on the recursive definitions to ensure that they possess least fixpoints (of both classes), and computation rules are given for computing these two fixpoints: the 'full' computation rule, which leads to the least weak fixpoint. A general class of computation rules, called 'safe innermost', also lead to the latter fixpoint. The "leftmost innermost" rule is a special case of those, for the LISP recursive definitions.

AIM-164 CS-272 AD742748
Zohar Manna, Jean Vuillemin,
Fixpoint Approach to the Theory of Computation,
29 pages, April 1972.

Following the fixpoint theory of Scott, we propose to define the semantics of computer programs in terms of the least fixpoints of recursive programs. This allows one not only to justify all existing verification techniques, but also to extend them to handle various properties of computer programs, including correctness, termination and equivalence, in a uniform manner.

*AIM-165 CS-280 AD742751
D. A. Bochvar,
Two Papers on Partial Predicate Calculus,
50 pages, April 1972.

These papers, published in 1938 and 1943, contain the first treatment of a logic of partial predicates. Bochvar's treatment is of current interest for two reasons. First, partial predicate and function logic are important for mathematical theory of computation because functions defined by programs or by recursion cannot be guaranteed to be total. Second, natural language may be better approximated by a logic in which some sentences may be undefined than by a conventional logic. Bochvar use of his system to avoid Russell's paradox is of interest here, and in partial predicate logic it may be possible to get more of an axiomatization of truth and knowledge than in a conventional logic.

The papers translated are **On a three-valued logical calculus and its application to the analysis of contradictions,** *Recueil*

Mathematique, N. S. 4 (1938), pp. 287-308, and **On the consistency of a three-valued logical calculus**, ibid. 12 (1943), pp. 353-369.

We also print a review and a correction by Alonzo Church that appeared in the Journal of Symbolic Logic. The review was in Vol. 4.2 (June 1939), p. 99, and the additional comment was in Vol. 5.3 (September 1940), p. 119.

AIM-166 CS-281 AD-743598
Lynn H. Quam, S. Liebes P. B. Tucker, M. J. Hannah, B. G. Eross,
Computer Interactive Picture Processing,
40 pages, April 1972.

This report describes work done in image processing using an interactive computer system. Techniques for image differencing are described and examples using images returned from Mars by the Mariner Nine spacecraft are shown. Also described are techniques for stereo image processing. Stereo processing for both conventional camera systems and the Viking 1975 Lander camera system is reviewed.

AIM-167 CS-282 AD747254
Ashok K. Chandra,
Efficient Compilation of Linear Recursive Programs,
43 pages, June 1972.

We consider the class of linear recursive programs. A linear recursive program is a set of procedures where each procedure can make at most one recursive call. The conventional stack implementation of recursion requires time and space both proportional to n, the depth of recursion. It is shown that in order to implement linear recursion so as to execute in time n one doesn't need space proportional to n : n^{ϵ} for sufficiently small ϵ will do. It is also known that with constant space one can implement linear recursion in time n^2. We show that one can do much better: $n^{1+\epsilon}$ for arbitrarily small ϵ. We also describe an algorithm that lies between thee two: it takes time n log(n) and space log(n).

It is shown that several problems are closely related to the linear recursion problem, for example, the problem of reversing an input tape given a finite automaton with several one-way heads. By casting all these problems into canonical form, efficient solutions are obtained simultaneously for all.

AIM-168 CS-287 AD746146
Shigeru Igarashi,
Admissibility of Fixed-point Induction in First-order Logic of Typed Theories,
Diskfile: FIXPNT.IGR[AIM,DOC]
40 pages, May 1972.

First-order logic is extended so as to deal with typed theories, especially that of continuous functions with fixed-point induction formalized by D. Scott. The translation of his formal system, or the λ calculus-oriented system derived and implemented by R. Milner, into this logic amounts to adding predicate calculus features to them.

In such a logic the fixed-point induction axioms are no longer valid, in general, so that we characterize formulas for which Scott-type induction is applicable, in terms of syntax which can be checked by machines automatically.

AIM-169 CS-288
Robin Milner,
Logic for Computable Functions: Description of a Machine Implementation,
Diskfile: LCFMAN.RGM[AIM,DOC],
36 pages, May 1972.

This paper is primarily a user's manual for LCF, a proof-checking program for a logic of computable functions proposed by Dana Scott in 1969, but unpublished by him. We use the name LCF also for the logic itself,

which is presented at the start of the paper. The proof-checking program is designed to allow the user interactively to generate formal proofs about computable functions and functionals over a variety of domains, including those of interest to the computer scientist -- for example, integers, lists and computer programs and their semantics. The user's task is alleviated by two features: a subgoaling facility and a powerful simplification mechanism. Applications include proofs of program correctness and in particular of compiler correctness; these applications are not discussed herein, but are illustrated in the papers referenced in the introduction.

AIM-170 CS-289 AD748607
Yorick Wilks,
Lakoff on Linguistics and Natural Logic,
Diskfile: LAKOFF.YAW[AIM,DOC]
19 pages, June 1972.

The paper examines and criticizes Lakoff's notions of a natural logic and of a generative semantics described in terms of logic, I argue that the relationship of these notions to logic as normally understood is unclear, but I suggest, in the course of the paper, a number of possible interpretations of his thesis of generative semantics. I argue further that on these interpretations a mere notational variant of Chomskyan theory. I argue, too, that Lakoff's work may provide a service in that it constitutes a *reductio ad absurdum* of the derivational paradigm of modern linguistics; and shows, inadvertently, that only a system with the ability to reconsider its own inferences can do the job that Lakoff sets up for linguistic enquirey -- that is to say only an 'artificial intelligence' system.

AIM-171 CS-290 AD746147
Roger Schank,
Adverbs and Belief,
30 pages, June 1972.

The treatment of a certain class of adverbs in conceptual representation is given. Certain adverbs are shown to be representative of complex belief structures. These adverbs serve as pointers that explain where the sentence that they modify belongs in a belief structure.

*AIM-172 CS-299 AD752801
Sylvia Weber Russell,
Semantic Categories of Nominals for Conceptual Dependency Analysis of Natural Language,
64 pages, July 1972.

A system for the semantic categorization of conceptual objects (nominals) is provided. The system is intended to aid computer understanding of natural language. Specific implementations for 'noun-pairs' and prepositional phrases are offered.

*AIM-173 CS-305 AD755139
Gerald Jacob Agin,
Representation and Description of Curved Objects,
Thesis: Ph.D. in Computer Science,
134 pages, October 1972.

Three dimensional images, similar to depth maps, are obtained with a triangulation system using a television camera, and a deflectable laser beam diverged into a plane by a cylindrical lens.

Complex objects are represented as structures joining parts called generalized cylinders. These primitives are formalized in a volume representation by an arbitrary cross section varying along a space curve axis. Several types of joint structures are discussed.

Experimental results are shown for the description (building of internal computer models) of a handful of complex objects, beginning with laser range data from actual objects. Our programs have generated complete descriptions of rings, cones, and snake-like objects, all of which may be

described by a single primitive. Complex objects, such as dolls, have been segmented into parts, most of which are well described by programs which implement generalized cylinder descriptions.

*AIM-174 CS-303 PB212827
Francis Lockwood Morris,
Correctness of Translations of Programming Languages -- an Algebraic Approach,
Thesis: Ph.D. in Computer Science,
124 pages, August 1972.

Programming languages and their sets of meanings can be modelled by general operator algebras; semantic functions and compiling functions by homomorphisms of operator algebras. A restricted class of individual programs, machines, and computations can be modelled in a uniform manner by binary relational algebras. A restricted class of individual manner by binary relational algebras. These two applications of algebra to computing are compatible: the semantic function provided by interpreting ('running') one binary relational algebra on another is a homomorphism on an operator algebra whose elements are binary relational algebras.

Using these mathematical tools, proofs can be provided systematically of the correctness of compilers for fragmentary programming languages each embodying a single language 'feature'. Exemplary proofs are given for statement sequences, arithmetic expressions, Boolean expressions, assignment statements, and while statement. Moreover, proofs of this sort can be combined to provide (synthetic) proofs for in principle, many different complete programming languages. One example of such a synthesis is given.

AIM-175 CS-307
Hozumi Tanaka,
Hadamard Transform for Speech Wave Analysis,
Diskfile: HADAM.HT[AIM,DOC],
34 pages, August 1972.

Two methods of speech wave analysis using the Hadamard transform are discussed. The first method is a direct application of the Hadamard transform for speech waves. The reason this method yields poor results is discussed. The second method is the application of the Hadamard transform to a log-magnitude frequency spectrum. After the application of the Fourier transform the Hadamard transform is applied to detect a pitch period or to get a smoothed spectrum. This method shows some positive aspects of the Hadamard transform for the analysis of a speech wave with regard to the reduction of processing time required for smoothing, but at the cost of precision. A formant tracking program for voiced speech is implemented by using this method and an edge following technique used in scene analysis.

*AIM-176 CS-308 AD754109
Jerome A. Feldman, J. R. Low, D. C. Swinehart, R. H. Taylor,
Recent Developments in SAIL -- an ALGOL based Language for Artificial Intelligence,
22 pages, November 1972.

New features added to SAIL, an ALGOL based languaged for the PDP-10, are discussed. The features include: procedure variables; multiple processes; coroutines; a limited form of backtracking; an event mechanism for inter-process communication; and matching procedures, a new way of searching the LEAP associative data base.

AIM-177 CS-311
Richard Paul,
Modelling, Trajectory Calculation and Servoing of a Computer Controlled Arm,
Thesis: Ph.D. in Computer Science,
89 pages, November 1972.

The problem of computer contral of an arm is divided into four parts: modelling, trajectory calculation, servoing and control.

In modelling we use a symbolic data structure to represent objects in the environment. The program considers how the hand may be positioned to grasp these objects and plans how to turn and position them in order to make various moves. An arm model is used to calculate the configuration-dependent dynamic properties of the arm before it is moved.

The arm is moved along time-coordinated space trajectories in which velocity and acceleration are controlled. Trajectories are calculated for motions along defined space curves, as in turning a crank; in such trajectories various joints must be free due to external motion constraints.

The arm is servoed by a small computer. No analog servo is used. The servo is compensated for gravity loading and for configuration-dependent dynamic properties of the arm.

In order to control the arm, a planning program interprets symbolic arm control instructions and generates a plan consisting of arm motions and hand actions.

The move planning program has worked successfully in the manipulation of plane faced objects. Complex motions, such as locating a bold and screwing a nut onto it, have also been performed.

*AIM-178 CS-312 AD754108
Aharon Gill,
Visual Feedback and Related Problems in Computer Controlled Hand eye Coordination,
Thesis: Ph.D. in Electrical Engineering,
130 pages, October 1972.

A set of programs for precise manipulation of simple planar bounded objects, by means of visual feedback, was developed for use in the Stanford hand-eye system. The system includes a six degrees of freedom computer controlled manipulator (arm and hand) and a fully instrumented computer television camera.

The image of the hand and manipulated objects is acquired by the computer through the camera. The stored image is analyzed using a corner and line finding program developed for this purpose. The analysis is simplified by using all the information available about the objects and the hand, and previously measured coordination errors. Simple touch and force sensing by the arm help the determination of three dimensional positions from one view.

The utility of the information used to simplify the scene analysis depends on the accuracy of the geometrical models of the camera and arm. A set of calibration updating techniques and programs was developed to maintain the accuracy of the camera model relative to the arm model.

The precision obtained is better than .1 inch. It is limited by the resolution of the imaging system and of the arm position measuring system.

*AIM-179 CS-320
Bruce G. Baumgart,
Winged Edge Polyhedron Representation,
46 pages, October 1972.

A winged edge polyhedron representation is

stated and a set of primitives that preserve Euler's F-E+V=2 equation are explained. Present use of this representation in Artificial Intelligence for computer graphics and world modeling is illustrated and its intended future application to computer vision is described.

*AIM-180 CS-321 AD759712
Ruzena Bajcsy,
Computer Identification of Textured Visual Scenes,
Thesis: Ph.D. in Computer Science,
156 pages, October 1972.

This work deals with computer analysis of textured outdoor scenes involving grass, trees, water and clouds. Descriptions of texture are formalized from natural language descriptions; local descriptors are obtained from the directional and non-directional components of the Fourier transform power spectrum. Analytic expressions are obtained for orientation, contrast, size, spacing, and in periodic cases, the locations of texture elements. These local descriptors are defined over windows of various sizes; the choice of sizes is made by a simple higher-level program.

The process of region growing is represented by a sheaf-theoretical model which formalizes the operation of pasting local structure (over a window) into global structure (over a region). Programs were implemented which form regions of similar color and similar texture with respect to the local descriptors.

An interpretation is made of texture gradient as distance gradient in space. A simple world model is described. An interpretation of texture regions and texture gradient is made with a simulated correspondence with the world model. We find that a problem-solving approach, involving hypothesis-verification, more satisfactory than an earlier pattern recognition effort (Bajcsy 1970) and more crucial to work with complex scenes than in scenes of polyhedra. Geometric clues from relative sizes, texture gradients, and interposition are important in interpretation.

*AIM-181 CS-325
Bruce G. Buchanan,
Review of Hubert Dreyfus' 'What Computers Can't Do': a Critique of Artificial Reason,
14 pages, November 1972.

The recent book "What Computers Can't Do" by Hubert Dreyfus is an attack on artificial intelligence research. This review takes the position that the philosophical content of the book is interesting, but that the attack on artificial intelligence is not well reasoned.

*AIM-182 CS-326 AD754107
Kenneth Mark Colby and Franklin Dennis Hilf,
Can Expert Judges, using Transcripts of Teletyped Psychiatric Interviews, Distinguish Human Paranoid Patients from a Computer Simulation of Paranoid Processes?,
10 pages, December, 1972.

Expert judges (psychiatrists and computer scientists) could not correctly distinguish a simulation model of paranoid processes from actual paranoid patients.

*AIM-183 CS-344 AD759716
Roger C. Schank,
The Fourteen Primitive Actions and their Inferences,
70 pages, March 1973.

In order to represent the conceptual information underlying a natural language sentence, a conceptual structure has been established that uses the basic actor-action-object framework. It was the intent that these structures have only one representation for one meaning, regardless of the semantic form of the sentence being represented.

Actions were reduced to their basic parts so as to effect this. It was found that only fourteen basic actions were needed as building blocks by which all verbs can be represented. Each of these actions has a set of actions or states which can be inferred when they are present.

*AIM-184 CS-330 AD758651
Malcolm Newey,
Axioms and Theorems for Integers, Lists and Finite Sets in LCF,
53 pages, January 1973.

LCF (Logic for Computable Functions) is being promoted as a formal language suitable for the discussion of various problems in the Mathematical Theory of Computation (MTC). To this end, several examples of MTC problems have been formalised and proofs have been exhibited using the LCF proof-checker. However, in these examples, there has been a certain amount of ad-hoc-ery in the proofs; namely many mathematical theorems have been assumed without proof and no axiomatisation of the mathematical domains involved was given. This paper describes a suitable mathematical environment for future LCF experiments and its axiomatic basis. The environment developed deemed appropriate for such experiments, consists of a large body of theorems from the areas of integer arithmetic, list manipulation and finite set theory.

*AIM-185 CS-333 AD757367
Ashok K. Chandra, Zohar Manna,
On the Power of Programming Features,
29 pages, January 1973.

We consider the power of several programming features such as counters, pushdown stacks, queues, arrays, recursion and equality. In this study program schemas are used as the model for computation. The relations between the powers of these features is completely described by a comparison diagram.

*AIM-186 CS-332 AD758645
Robin Milner,
Models of LCF,
17 pages, January 1973.

LCF is a deductive system for computable functions proposed by D. Scott in 1969 in an unpublished memorandum. The purpose of the present paper is to demonstrate the soundness of the system with respect to certain models, which are partially ordered domains of continuous functions. This demonstration was supplied by Scott in his memorandum; the present paper is merely intended to make this work more accessible.

*AIM-187 CS-331 AD757364
George E. Collins,
The Computing Time of the Euclidean Algorithm,
17 pages, January 1973.

The maximum, minimum and average computing times of the classical Euclidean algorithm for the greatest common divisor of two integers are derived, to within codominance, as functions of the lengths of the two inputs and the output.

*AIM-188 CS-336 AD758646
Ashok K. Chandra,
On the Properties and Applications of Program Schemas,
Thesis: Ph.D. in Computer Science,
231 pages, March 1973.

The interesting questions one can ask about program schemas include questions about the 'power' of classes of schemas and their decision problems viz. halting divergence, equivalence, etc. We first consider the powers of schemas with various features: recursion, equality tests, and several data structures such as pushdown stacks, lists, queues and arrays. We then consider the decision problems for schemas with equality and with commutative and invertible functions. Finally a generalized class of

schemas is described in an attempt to unify the various classes of uninterpreted and semi-interpreted schemas and schemas with special data structures.

*AIM-189 CS-337 PB218682
James Gips, George Stiny,
Aesthetics Systems,
22 pages, January 1973.

The formal structure of aesthetics systems is defined. Aesthetics systems provide for the essential tasks of interpretation and evaluation in aesthetic analysis. Kolmogorov's formulation of information theory is applicable. An aesthetics system for a class of non-representational, geometric paintings and its application to three actual paintings is described in the Appendix.

AIM-190 CS-340 AD759714
Malcolm Newey,
Notes on a Problem Involving Permutations as Sequences,
20 pages, March 1973.

The problem (attributed to R. M. Karp by Knuth) is to describe the sequences of minimum length which contain, as subsequences, all the permutations of an alphabet of n symbols. This paper catalogs some of the easy observations on the problem and proves that the minimum lengths for n=5, n=6 and n=7 are 19, 28, and 39 respectively. Also presented is a construction which yields (for n>2) many appropriate sequences of length n^2-2n+4 so giving an upper bound on length of minimum strings which matches exactly all known values.

AIM-191 CS-341 AD764272
Shmuel M. Katz, Zohar Manna,
A Heuristic Approach to Program Verification,
40 pages, March 1973.

We present various heuristic techniques for use in proving the correctness of computer programs. The techniques are designed to obtain automatically the "inductive assertions" attached to the loops of the program which previously required human "understanding" of the program's approaches: one in which we obtain the inductive assertion by analyzing predicates which are known to be true at the entrances and exits of the loop (top-down approach), and another in which we generate the inductive assertion directly from the statements of the loop (bottom-up approach).

AIM-192 CS-345
George E. Collins, Ellis Horowitz,
The Minimum Root Separation of a Polynomial,
13 pages, April 1973.

The minimum root separation of a complex polynomial A is defined as the minimum of the distances between distinct roots of A. For polynomials with Gaussian integer coefficients and no multiple roots, three lower bounds are derived for the root separation. In each case the bound is a function of the degree, n, of A and the sum, d, of the absolute values of the coefficients of A. The notion of a semi-norm for a commutative ring is defined, and it is shown how any semi-norm can be extended to polynomial rings and matrix rings, obtaining a very general analogue of Hadamard's determinant theorem.

AIM-193 CS-346 AD759717
Kenneth Mark Colby,
The Rationale for Computer Based Treatment of Language Difficulties in Nonspeaking Autistic Children,
Diskfile: AUTISM.KMC[AIM,DOC],
13 pages, March 1973.

The principles underlying a computer-based treatment method for language acquisition in nonspeaking autistic children are described. The main principle involves encouragement of exploratory learning with minimum adult interference.

AIM-194 CS-347 PB221170/4
Kenneth Mark Colby, Franklin Dennis Hilf,
Multidimensional Analysis in Evaluating a Simulation of Paranoid Thought,
10 pages, May 1973.

The limitations of Turing's Test as an evaluation procedure are reviewed. More valuable are tests which ask expert judges to make ratings along multiple dimensions essential to the model. In this way the model's weaknesses become clarified and the model builder learns where the model must be improved.

AIM-195 CS-356 PB222164
David Canfield Smith, Horace J. Enea,
MLISP2,
Diskfile: MLISP2.DAV[AIM,DOC],
91 pages, May 1973.

MLISP2 is a high-level programming language based on LISP. Features:

1. The notation of MLISP.
2. Extensibility -- the ability to extend the language and to define new languages.
3. Pattern matching -- the ability to match input against context free or sensitive patterns.
4. Backtracking -- the ability to set decision points, manipulate contexts and backtrack.

AIM-196 CS-357 AD762471
Neil M. Goldman, Christopher K. Riesbeck,
A Conceptually Based Sentence Paraphraser,
Diskfile: MARGIE.NMG[AIM,DOC],
88 pages, May 1973.

This report describes a system of programs which perform natural language processing based on an underlying language free (conceptual) representation of meaning. This system is used to produce sentence paraphrases which demonstrate a form of understanding with respect to a given context. Particular emphasis has been placed on the major subtasks of language analysis (mapping natural language into conceptual structures) and language generation (mapping conceptual structures into natural language), and on the interaction between these processes and a conceptual memory model.

*AIM-197 CS-358 AD762470
Roger C. Schank, Charles J. Rieger III,
Inference and the Computer Understanding of Natural Language,
63 pages, May 1973.

The notion of computer understanding of natural language is examined relative to inference mechanisms designed to function in a language-free deep conceptual base (Conceptual Dependency). The conceptual analysis of a natural language seentence into this conceptual base, and the nature of the memory which stores and operates upon these conceptual structures are described from both theoretical and practical standpoints. The various types of inferences which can be made during and after the conceptual analysis of a sentence are defined, and a functioning program which performs these inference tasks is described. Actual computer output is included.

AIM-198 CS-364 AD763611
Ravindra B. Thosar,
Estimation of Probability Density using Signature Tables for Application to Pattern Recognition,
37 pages, May 1973.

Signature table training method consists of cumulative evaluation of a function (such as a probability density) at pre-assigned co-ordinate values of input parameters to the table. The training is conditional: based on a binary valued 'learning' input to a table which is compared to the label attached to each training sample. Interpretation of an unknown sample vector is then equivalent of a table lookup, i.e. extraction of the function

value stored at the proper co-ordinates. Such a technique is very useful when a large number of samples must be interpreted as in the case of samples must be interpreted as in the case of speech recognition and the time required for the trainng as well as for the recognition is at a premium. However, this method is limited by prhibitive storage requirements, even for a moderate number of parameters, when their relative independence cannot be assumed. This report investigates the conditions under which the higher dimensional probability density function can be decomposed so that the density estimate is obtained by a hierarchy of signature tables with consequent reduction in the storage requirement. Practical utility of the theoretical results obtained in the report is demonstrated by a vowel recognition experiment.

AIM-199 CS-398 AD771300
Bruce G. Baumgart,
Image Contouring and Comparing,
52 pages, (in preparation).

A contour image representation is stated and an algorithm for converting a set of digital television images into this representation is explained. The algorithm consists of five steps: digital image thresholding, binary image contouring, polygon nesting, polygon smoothing, and polygon comparing. An implementation of the algorithm is the main routine of a program called CRE; auxiliary routines provide cart and turn table control, TV camera input, image display, and xerox printer output. A serendip application of CRE to type font construction is explained. Details about the intended application of CRE to the perception of physical objects will appear in sequels to this paper.

AIM-200 CS-365
Shigeru Igarashi, David C. Luckham, Ralph L. London,
Automatic Program Verification I: Logical Basis and its Implementation,
50 pages, May 1973.

Defining the semantics of programming languages by axioms and rules of inference yields a deduction system within which proofs may be given that programs satisfy specifications. The deduction system herein is shown to be consistent and also deductive complete with respect to Hoare's sustem. A subgoaler for the deductive system is described whose input is a significant subset of Pascal programs plus inductive assertions. The output is a set of verification conditions or lemmas to be proved. Several non-trivial arithmetic and sorting programs have been shown to satisfy specifications by using an interactive theorem prover to automatically generate prrofs of the veification conditions Additional components for a more powerful verficiation system are under construction.

AIM-201 CS-366 AD763673
Gunnar Rutger Grape,
Model Based (Intermediate Level) Computer Vision,
Thesis: Ph.D. in Computer Science,
256 pages, May 1973.

A system for computer vision is presented, which is based on two-dimensional prototypes, and which uses a hierarchy of features for mapping purposes. More specifically, we are dealing with scenes composed of planar faced, convex objects. Extensions to the general planar faced case are discussed. The visual input is provided by a TV-camera, and the problem is to interpret that input by computer, as a projection of a three-dimensional scene. The digitized picture is first scanned for significant intensity gradients (called edges), which are likely to appear at region- and object junctions. The two-dimensional scene-representation given by the totality of such intensity discontinuities (that word used somewhat inexactly) is referred to in the sequel as the 'edge-drawing', and constitutes the input to the vision system presented here.

The system proposed and demonstrated in

this paper utilizes perspectively consistent two-dimensional models (prototypes) of views of three-dimensional objects, and interpretations of scene-representations are based on the establish- ment of mapping relationships from conglomerates of scene-elements (line-constellations) to prototype templates. The prorotypes are learned by the program through analysis of - and generalization on - ideal instances. The system works better than any sequential (or other) system presented so far. It should be well suited to the context of a complete vision system using depth, occlusion, support relations, etc. The general case of irregularly shaped, planar faced objects, including concave ones, would necessitate such an extended context.

AIM-202 CS-368 AD764396
Roger C. Schank, Yorick Wilks,
The Goals of Linguistic Theory Revisited,
44 pages, May 1973.

We examine the original goals of generative linguistic theory. We suggest that these goals were well defined but misguided with respect to their avoidance the problem of modelling performance. We developments such as Generative Semantics, it is no longer clear that the goals are clearly defined. We argue that it is vital for linguistics to concern itself with the procedures that humans use in language. We then introduce a number of basic human competencies, in the field of language understanding, understanding in context and the use of inferential information, and argue that the modelling of these aspects of language understanding requires procedures of a sort that cannot be easily accomodated within the dominant paradigm. In particular, we argue that the procedures that will be required in these cases ought to be linguistic, and that the simple-minded importation of techniques, and that the simple-minded importation of techniques from logic may create a linguistics in which there cannot be procedures of the required sort.

AIM-203 CS-369 AD764274
Roger C. Schank,
The Development of Conceptual Structures in Children,
31 pages, May 1973.

Previous papers by the author have hypothesized that is is possible to represent the meaning of natural language sentences using a framework which has only fourteen primitive ACTs. This paper addresses the problem of when and how these ACTs might be learned by children. The speech of a child of age 2 is examined for possible knowledge of the primitive ACTs as well as the conceptual relations underlying language. It is shown that there is evidence that the conceptual structures underlying language are probably complete by age 2. Next a child is studied from birth to age 1. The emergence of the primitive ACTs and the conceptual relations is traced. The hypothesis is made that the structures that underlie and are necessary for language are present by age 1.

AIM-204 CS-373 AD765353
Kurt VanLehn,
SAIL Users Manual,
Diskfile: SAIL.KVL[AIM,DOC],
122 pages, July 1973.

SAIL is a high-level programming language for the PDP-10 computer. It includes an extended ALGOL 60 compiler and a companion set of execution-time routines. In addition to ALGOL, the language features: (1) flexible linking to hand-coded machine language algorithms, (2) complete access to the PDP-10 I/O facilities, (3) a complete system of compile-time arithmetic and logic as well as a flexible macro system (4) user modifiable error handling, (5) backtracking, and (6) interrupt facilities. Furthermore, a subset of the SAIL language, called LEAP, provides facilities for (1) sets and lists, (2) an associative data structure, (3) independent processes, and (4) procedure variables. The LEAP subset of SAIL is an extension of the

LEAP language, which was designed by J. Feldman and P. Rovner, and implemented on Lincoln Laboratory's TX-2 (see [Feldman & Rovner]). The extensions to LEAP are partially described in 'Recent Developments in SAIL' (see [Feldman]).

This manual describes the SAIL language and the execution-time routines for the typical SAIL user: a non-novice programmer with some knowledge of ALGOL. It lies somewhere between being a tutorial and a reference manual.

AIM-205 CS-370 AD764288
N. S. Sridharan, et al.
A Heuristic Program to Discover Syntheses for Complex Organic Molecules,
30 pages, June 1973.

Organic Chemical Synthesis is found to be a suitable program for developing machine intelligence. A previous paper described the objective and global characteristics of the project. The present article aims to describe the program organization as a heuristic search, the design of the Problem Solving Tree and the search procedures in considerable detail. Examples of syntheses discovered and the problem solving tree developed are given. The programs are written mostly in PL1(F) applicable to an IBM 360/67 and the timings (batch mode) indicate that we have fast and efficient practical systems.

AIM-206 CS-377 AD764652
Yorick Wilks,
Preference Semantics,
20 pages, July 1973.

Preference semantics [PS] is a set of formal procedures for representing the meaning structure of natural language, with a view to embodying that structure within a system that can be said to understand, rather than within what I would call the 'derivational paradigm', of transformational grammar [TG] and generative semantics [GS], which seeks to determine the well-formedness, or otherwise, of sentences.

I outline a system of preference semantics that does this: for each phrase or clause of a complex sentence, the system builds up a network of lexical trees with the aid of structured items called templates and, at the next level, it structures those networks with higher level items called paraplates and common-sense inference rules. At each stage the system directs itself towards the correct network by always opting for the most 'semantically dense' one it can construct. I suggest that this opting for the 'greatest semantic density' can be seen as an interpretation of Joos' 'Semantic Axiom Number 1'.

I argue that the analysis of quite simple examples requires the use of inductive rules of inference which cannot, theoretically cannot, be accomodated within the derivational paradigm. I contrast this derivational paradigm of language processing with the artificial intelligence [AI] paradigm.

AIM-207 CS-378 AD767333
James Anderson Moorer,
The 'Optimum-comb' Method of Pitch Period Analysis in Speech,
25 pages, June 1967.

A new method of tracking the fundamental frequency of voiced speech is descirbed. The method is shown to be of similar accuracy as the Cepstrum technique. Since the method involves only addition, no multiplication, it is shown to be faster than the SIFT algorithm.

AIM-208 CS-379 AD767334
James Anderson Moorer,
The Heterodyne Method of Analysis of Transient Waveforms,
25 pages, June 1973.

A method of analysis of transient waveforms is discussed. Its properties and limitations are presented in the context of musical tones. The method is shown to be useful when the risetimes of the partials of the tone are not too short. An extention to inharmonic partials and polyphonic musical sound is discussed.

AIM-209 CS-380 AD767695
Yoram Yakimovsky,
Scene Analysis using a Semantic Base for Region Growing,
Thesis: Ph.D. in Computer Science,
120 pages, July 1973.

The problem of breaking an image into meaningful regions is considered. A probabilistic semantic basis is effectively integrated with the segmentation process, providing various decision criteria. Learning facilities are provided for generating interactively the probabilistic bases. A programming system which is based on these ideas and its successful application to two problem domains are described.

AIM-210 CS-382 AD767335
Zohar Manna, Amir Pnueli,
Axiomatic Approach to Total Correctness of Programs,
25 pages, July 1973.

We present here an axiomatic approach which enables one to prove by formal methods that his program is 'totally correct' (i.e., it terminates and is logically correct -- does what it is supposed to do). The approach is similar to Hoare's approach for proving that a program is 'partially correct' (i.e., that whenever it terminates it produces correct results). Our extension to Hoare's method lies in the possibility of proving correctness and termination at once, and in the enlarged scope of properties that can be proved by it.