CS 178

RESEARCH
IN THE
COMPUTER SCIENCE DEPARTMENT
AND SELECTED OTHER RESEARCH IN COMPUTING
AT
STANFORD UNIVERSITY

OCTOBER 17, 1970

RESEARCH

in the

COMPUTER SCIENCE DEPARTMENT

and SELECTED OTHER RESEARCH IN COMPUTING

at

STANFORD UNIVERSITY

presented by

Professor George E. Forsythe
Chairman, Department of Computer Science

and

Professor William F. Miller
Vice President for Research;
Associate Provost for Computing

October 17, 1970

RESEARCH IN THE COMPUTER SCIENCE DEPARTMENT
and SELECTED OTHER RESEARCH IN COMPUTING
at STANFORD UNIVERSITY

October 17, 1970

presented by

Professors George E. Forsythe and William F. Miller

The research program of the Computer Science Department can perhaps be best summarized in terms of its research projects. The chart on the following page lists the projects and the participation by faculty and students. The sections following the chart provide descriptions of the individual projects,

There are a number of projects in other schools or departments which are making significant contributions to computer science; and these add to the total computer environment. Descriptions of a few of these projects are also included with this report. This list of projects outside of Computer Science does not purport to be complete or even representative.

The Publications Committee of the Computer Science Department has undertaken a project to compile a comprehensive bibliography of research reports prepared by its faculty and graduated students. This is a constantly changing file stored as file &K608.BIBLIOGRAPHY ON SYS13 of the Computation Center's Campus Facility.

RESEARCH PROJECTS OF THE COMPUTER SCIENCE DEPARTMENT

Project	Numerical Analysis	Artificial Intelligence	Computer Simulation of Belief Systems	Programming Models and the Control of Computing Systems	Graphic Processing: Analysis, Languages, Data Structure	SLAC COMP. GROUP	Programming Languages	Theory of Computation Analysis of Algorithms	H-P 2116 Control Computer	Mathematical Programming Language	Operations Research (2)
Supporting Agency(s)	ONR NSF AEC	ARPA	NIH	AEC	NSF	AEC	IBM XEROX	ONR NSF	Hewlett-Packard	NSF	AEC ARO NSF ONR
Principal Investigator(s)	G.E. Forsythe J. G. Herrio G. H. Golub	J. McCarthy E. Feigenbaum A. Samuel	K.M. Colby	W.F. Miller	W.F. Miller	W.F. Miller	B. Floyd D. Knuth	R. Floyd D. Knuth	H. Stone	G. R. Dantzig	G.R. Dantzig
Other Investigator(s) (Faculty) and (res. Assoc.)	F. Dorr C. Moler	J. Feldman (3) R. Floyd Z. Manna L. Earnest E. Ashcroft A. Biermann T. Binford A. Duffield R. Engelmore A. Kay B. Buchanan M. Hueckel D. Luckham	J. Feldman (3) R. Schank H. Enea F. Hilf	H. Stone R. Fabry		J.R. Ehrman C. Riedl G.A. Robinson H. Sall		G. Forsythe J. Hopcroft		A. Manne (Op. Res.)	R.W. Cottle A.S. Manne (Op. Res.) R. Wilson (Business)
Number of Graduate Research Assistant(s) (1)											
Computer Science Dept.	7	16	3	3	2	7		2	1	2	1
Supported from Other Dept.	-	3				1			1	2	5
Fellows and outside Support Working on Project	3	8				3		3			

(1) Some graduate students are teaching assistants, and some are not yet associated with a research project.

(2) Professor Dantzig holds a joint appointment in Computer Science and Operations Research; these research projects are administratively in the latter department.

(3) On leave 1970-1971 academic year.

TABLE OF CONTENTS

SECTION		PAGE
1.	Numerical Analysis.	1
2.	Artificial Intelligence - Complete Project.	9
	Heuristic Dendral	85
	Computer Simulation of Belief Systems	95
3.	Programming Models and the Control of Computing Systems	101
4.	Graphics Processing: Analysis, Languages, Data Structure . . .	106
5.	SLAC.	107
6.	Programming Languages	114
7.	Analysis of Algorithms.	116
8.	H-P 2116 Control Computer	118
9.	Mathematical Programming Language and Related Operations Research Activity.	120

Selected Other Research in Computing

1.	Digital Systems Laboratory	128
2.	Computing and Business Education.	169
3.	Information Retrieval and Library Automation: SPIRES/BALLOTS.	172

NUMERICAL ANALYSIS RESEARCH

The major computer science research projects under way in numerical analysis are supported by the Office of Naval Research, the National Science Foundation, and the Atomic Energy Commission. Professor George Forsythe is the Principal Investigator of the ONR project, which has been supported at Stanford for 12 years. Professor Forsythe and Professor John G. Herriot are Principal Investigators of the project supported by NSF. Professor Gene Golub is Principal Investigator of the AEC project, but during Golub's sabbatical Forsythe is acting for him. Some of Professor George Dantzig's work is closely related, but is separately described under "Operations Research" below.

These existing projects provide modest funds to bring visiting faculty to Stanford for periods of up to one year.

Statements of the purpose of the research and a listing of the research areas of the three projects follow.

Office of Naval Research

Purpose of project: The project has the purpose of conducting research to increase the effectiveness with which automatic digital computers are used to solve contemporary scientific and technological problems of a mathematical nature. This includes the invention, criticism, and particularly the mathematical analysis of algorithms for numerical computation. In the future it will deal in growing amounts also with algorithms for symbolic computation that enters mathematical problems. It will also

include a study of on-line man-machine interaction in the solution of mathematical problems.

Research areas: Research projects are selected from the following areas:

1. Analysis and computation methods for large, sparse matrices
2. Minimizing functions of many variables
3. Conversational mode of solving problems
4. Maintenance of scientific program libraries
5. Improving, certifying and recording algorithms
6. Round-off analysis
7. Solution of partial differential equations

National Science Foundation

Purpose of project: The project has purposes that differ very little from those of the Office of Naval Research project. The problem areas now being investigated include the solution of linear and nonlinear systems of algebraic equations and partial differential equations, and linear and non-linear least-squares problems.

Research areas: Research projects are mainly selected from the following areas:

1. Analysis of large sparse matrices
2. Algorithms for least squares problems, including linear least squares problems with inequality constraints, or with a quadratic constraint, perturbation theory, exponential fitting, interactive data fitting, and algorithms for large matrices
3. Solution of partial differential equations

4. Minimizing functions of many variables
5. Conversational mode of solving problems
6. Maintenance of scientific program libraries
7. Publication and evaluation of algorithms

Atomic Energy Commission

Purpose of project: Again, the general purposes of the AEC-sponsored project are approximately the same as those of the projects sponsored by the Office of Naval Research-and the National Science Foundation. There is, however, less emphasis on differential equations and more attention to problems of mathematical programming.

Research areas: Research projects are being selected from the following areas:

1. Nonlinear least-squares problems
2. Statistical calculations
3. Construction of invariant subspaces
4. Improving the solution of linear systems
5. Perturbation theory in linear least-squares problems
6. Fitting of curves by sums of exponentials
7. Least-squares algorithms for large matrices

Computation Center

A few of the faculty members in numerical analysis receive part of their support from the Stanford Computation Center. This is in return for consulting and leadership in the Center's scientific programming library.

For years the Center has had good compilers and languages that were dialects of Algol 60, and our library had been built around Algol. With the departure of the IBM 7090 in September 1967 and the Burroughs B5500 in December 1967, the Center now has only System/360 computers. Under OS/360 we have Fortran, including the Watfor computer, a very slow PL/1 compiler, and an extended Algol dialect called Algol W. Fortran, PL/1 and Algol W now provide for direct access input/output to disks, the storage of precompiled programs, and linkage with machine-language subroutines.

The situation is roughly this: many-of the world's best small algorithms are published in Algol 60; the most versatile compiler language for System/360 is Fortran; and IBM states that PL/1 is the language of the future. The problem, then, is what language should a scientific programming library now deal with?

Despite the over-all uncertainty, we are continuing to develop and collect useful algorithms. G. H. Golub and his students have been particularly active in this, and are generally using Algol W as the development language, while publishing in Algol 60 and translating to Fortran. Some students under G. E. Forsythe are collecting Fortran programs from Argonne National Laboratories and other sources, putting them on disk files of the 360/67 accessible from the Wylbur console system, making indexes, and creating much publicity about their existence. A number of translations to Fortran have also been made.

Our students are offering to consult for scientific computer users at Stanford who have special problems. It requires a good deal of effort to convince people to take the trouble to seek advice, and then one cannot always help them!

I. RECENT PUBLICATIONS

(a) Technical Reports

Technical Report No.	Title	Author	Date
CS 119	MATHEMATICAL PROGRAMMING LANGUAGE	G. Dantzig	5-15-68
cs 121	ACCURATE BOUNDS FOR THE EIGENVALUES OF THE LAPLACIAN AND APPLICATIONS TO RHOMBICAL DOMAINS	C.B. Moler	2-19-69
cs 122	HEURISTIC ANALYSIS OF NUMERICAL VARIANTS OF THE GRAM-SCHMIDT ORTHO-NORMALIZATION PROGRESS	W.C. Mitchell D.L. McCraith	2-24-69
CS 123	EMPIRICAL EVIDENCE FOR A PROPOSED DISTRIBUTION OF SMALL PRIME GAPS	R. P. Brent	2-28-69
CS 124	MATRIX DECOMPOSITIONS AND STATISTICAL CALCULATIONS	G.H. Golub	3-10-69
CS 128	THE METHOD OF ODD/EVEN REDUCTION AND FACTORIZATION WITH APPLICATION TO POISSONS EQUATION	G.H. Golub	April 69
cs 131	THE USE OF MAN-MACHINE INTERACTION IN DATA-FITTING PROBLEMS	L.B. Smith	March 69
cs 133	HANDBOOK SERIES LINEAR ALGEBRA SINGULAR VALUE DECOMPOSITION AND LEAST SQUARES SOLUTIONS	G.H. Golub C. Reinsch	May 69
cs 134 -	LINEAR LEAST SQUARES AND QUADRATIC PROGRAMMING	G.H. Golub M.A. Saunders	May 69
cs 137	FIXED POINTS OF ANALYTIC FUNCTIONS	P. Henrici	July 69
cs 140	DESIGN - THEN AND NOW	G.E. Forsythe	September 69
CS 141	BOUNDS FOR THE ERROR OF LINEAR SYSTEMS OF EQUATIONS USING THE THEORY OF MOMENTS	G. Dahlquist S.C. Eisenstat G.H. Golub	October 69
CS 142	STATIONARY VALUES OF THE RATIO OF QUADRATIC FORMS SUBJECT TO LINEAR CONSTRAINTS	G.H. Golub R. Underwood	November 69

(cont.)

Technical Report No.	Title	Author	Date
CS 143	THREE-STAGE VARIABLE-SHIFT ITERATIONS FOR THE SOLUTION OF POLYNOMIAL EQUATIONS WITH A POSTERIOR1 ERROR BOUNDS FOR THE ZEROS	M.A. Jenkins	August 69
CS 144	THE MAXIMUM AND MINIMUM OF A POSITIVE DEFINITE QUADRATIC POLYNOMIAL ON A SPHERE ARE CONVEX FUNCTIONS OF THE RADIUS	G.E. Forsythe	July 1969
CS 145	METHODS OF SEARCH FOR SOLVING POLYNOMIAL EQUATIONS	P. Henrici	December 69
CS 146	ROUNDOFF ERROR ANALYSIS OF THE FAST FOURIER TRANSFORM	G.U. Ramos	February 70
CS 147	PITFALLS IN COMPUTATION, OR WHY A MATH BOOK ISN'T ENOUGH	G.E. Forsythe	January 70
CS 150	ELEMENTARY PROOF OF THE WIELANDT-HOFFMAN THEOREM AND OF ITS GENERALIZATION	J.H. Wilkinson	
CS 151	ON THE PROPERTIES OF THE DERIVATIVES OF THE SOLUTION OF LAPLACE'S EQUATION AND THE ERRORS OF THE METHOD OF FINITE DIFFERENCES FOR BOUNDARY VALUES IN C_2 AND $C_{1,1}$	E.A. Volkov (G.E. Forsythe, translator)	January 70
CS 152	RAPID COMPUTATION OF INTERPOLATION FORMULAE AND MECHANICAL QUADRATURE RULES	S. Gustafson	February 70
CS 153	ERROR PROPAGATION BY USE OF INTERPOLUTION FORMULAE AND QUADRATURE RULES WHICH ARE COMPUTED NUMERICALLY	S. Gustafson	February 70
CS 155	THE METHOD OF ODD/EVEN REDUCTION AND FACTORIZATION WITH APPLICATION TO POISSON'S EQUATION, PART II	B.L. Buzbee G.H. Golub C.W. Nielson	March 70
CS 157	ALGORITHMS FOR MATRIX MULTIPLICATION	R.P. Brent	March 70

(b) Printed Publications

Richard H. Bartels and Gene H. Golub, "The simplex method of linear programming using LU decomposition,' Comm. Assoc. Comput. Mach. Vol. 12 (1969), pp. 266-268.

George E. Forsythe, "Solving a quadratic equation on a computer," pp. 138-152 of COSRIMS and George Boehm (editors), THE MATHEMATICAL SCIENCES, MIT Press, 1969.

George E. Forsythe, "Remarks on the paper by Dekker," pp. 49-51 of Bruno Dejon and Peter Henrici (editors), CONSTRUCTIVE ASPECTS OF THE FUNDAMENTAL THEOREM OF ALGEBRA, Wiley-Interscience, 1969.

George E. Forsythe, "What to do until the computer scientist comes," Amer. Math. Monthly, Vol. 75 (1968), pp. 454-462.

George E. Forsythe, "What is a satisfactory quadratic equation solver," pp. 51-61 of Bruno Dejon and Peter Henrici, op. cit.

George E. Forsythe, "Today's computational methods of linear algebra," pp. 106-132 of Anonymous, STUDIES IN NUMERICAL ANALYSIS I, Society for Industrial and Applied Mathematics. Philadelphia, 1968. (Reprint of earlier article.)

-G.E. Forsythe, "Design - then and now," The Digest Record of the ACM-SIAM-IEEE 1969 Joint Conference on Mathematical and Computer Aids to Design, Assoc. Comput. Machinery, 1969, pp. 2-10.

G.E. Forsythe and C.B. Moler, COMPUTER SOLUTION OF LINEAR ALGEBRAIC SYSTEMS, Authorized Japanese translation, 1969.

G.E. Forsythe and C.B. Moler, COMPUTER SOLUTION OF LINEAR ALGEBRAIC SYSTEMS, Unauthorized Russian translation, 1969.

George E. Forsythe and Wolfgang R. Wasow, FINITE-DIFFERENCE METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS, Authorized Japanese translation in two volumes, 1968.

George E. Forsythe and Wolfgang R. Wasow, op. cit., Unauthorized Russian translation, 1963. [Not previously reported.]

J. Alan George, "Review of Procédures ALGOL en analyse numkrique," Math. Comput., Vol. 23 (1969), pp. 675-676.

Gene H. Golub and John H. Welsch, "Calculation of Gauss quadrature rules," Math. Comput., Vol. 23 (1969), pp. 221-230.

Peter Henrici, "Methods of search for solving polynomial equations," J. Assoc. Comput. Machinery, Vol. 17 (1970), pp. 273-283.

M.A. Jenkins and J.F. Traub, "A three-stage variable-shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration," Numer. Math. 14 (1970), pp. 252-263.

(cont.)

P.L. Richman, "Compressible fluid flow and the approximation of iterated integrals of a singular function," Math. Comput., Vol. 23 (1968), pp. 355-372.

Lyle B. Smith, "Interval arithmetic determinant evaluation and its use in testing for a Chebyshev system," Comm. Assoc. Comput. Mach., Vol. 12 (1969), pp. 89-93.

J.M. Varah, "The calculation of the eigenvalues of a general complex matrix by inverse iteration," Math. Comput., Vol. 22 (1968), pp. 785-791.

J.M. Varah, "Rigorous machine bounds for the eigensystem of a general complex matrix, "Math. Comput., Vol. 22 (1968), pp. 793-801.

(c) Editorial Work

George E. Forsythe serves as an associate editor of Numerische Mathematik, and writes occasional reviews for Computing Reviews.

Gene H. Golub serves as an associate editor for Numerische Mathematik and the Mathematics of Computation.

Several persons helped J.G. Herriot with refereeing contributions to the Algorithms section of the Communications of the Association for Computing Machinery, until Herriot turned the editorship over to L. Fosdick in July 1969.

STANFORD ARTIFICIAL INTELLIGENCE PROJECT
MEMO AIM-117

April 1970

Project Technical Report

by

John McCarthy and the Artificial Intelligence Project Staff
Edward Feigenbaum, Joshua Lederberg and the
Heuristic DENDRAL Project Staff,

ABSTRACT: 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.

The research reported here was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense under Contract SD-193 and In part by the National Institutes of Mental health under Grant PHS MH 066-45-08.

Table of Contents

1, Introduction

2, Theory

2,1 Representation Theory

2,2 Mathematical Theory of Computation

3, Models of Cognitive Processes

3,1 Heuristic DENDRAL

3,2 Grammatical Inference

3,3 Computer Simulation of Belief Systems

3.4 Automatic Deduction

3,5 Machine Learning

4, Speech Recognition

4,1 Machine Learning

4,2 Analysis and Recognition Of Languages

5, Vision and Control

5,1 Hand-Eye Systems

5,2 Visual Control of a Vehicle

5,3 Visual Identification of People

Appendices

A, Publications

B, Theses

C, Film Reports

D, Artificial Intelligence Memos

E, Operating Notes

1. 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, The information for this study comes in part from observation of human behavior, including self-observation; but mainly from experiments with programs designed to solve problems chosen to require the Intellectual processes under study, Many blind alleys have been and are being followed, and many mistakes are being made, Nevertheless, a body of fundamental knowledge is accumulating,

We divide our work, somewhat arbitrarily, -into four areas:

1. theory
2. models of cognitive processes
3. speech recognition, and
4. vision and control

Except where noted below, support has come from the Advanced Research Projects Agency,

1,1 Theory

Work In Representation Theory is aimed at choosing a suitable representation for situations and the rules that describe how situations change, This description must be general enough to cover all problem solving situations and, even more Important, it must be able to express all likely states of knowledge of the situation and the rules by which It changes spontaneously or by the actions of the proolem solver,

Our work Includes Mathematical Theory of Computation, which treats, computer programs as mathematical objects and attempts to prove or disprove that they have certain properties, While this field Is not strictly a part of artificial Intelligence, there are a number of points of common interest, This work has the applied goal of eventual iy replacing much debugging by computer-checked formal proofs that programs meet their specifications,.

1,2 Models of Cognitive Processes

The largest project concerned with cognitive mode is is Heuristla DENDRAL, This work aims at emulating In a computer Program the inductive behavior of a chemist in tasks such as the identification of an unknown compound from mass spectrum data,

Our research In Grammatical Inference is deveioplng general method8 for inferring grammars from sample strings,

Computer Simulation of Belief Systems is a project with the long range goal of developing more satisfactory theories and models of psychopathological processes, Techniques for computer-mediated lnterviewing are being developed under a grant from the National Institutes of Mental Health,

Research ln Automatic Deduction has developed an increasingly powerful system for interactive theorem-proving, wrltten in LISP,

Machine Learnlng techniques are being applled to several problems, Samuel's Checker program continues tc improve, and a new program has been developed for the game of Go,

1,3 Speech Hecognltjon

Work on computer recognition of human speech initiated by Reddy is being extended in several ways, One current effort Is the appl icatlon of signature-table learning to the identification of speech segments,

Another effort Is concerned wlth the deslgn of special languages for man-machine communication, Careful analysts of prospective grammars wi I I reveal potential phonetic or word-boundary ambiguities, If these are avoided, the construction of reliable recognizers should be much eas I er ,

1,4 Vision and Control

The Hand-Eye Project continues to work on the perception of three-dimensional objects, using data from televlslon cameras, and computer-control led manipulation of these objects by mechanical arms, An applled goal of this research Is to be able to automate certain assembly tasks that currently must be performed by humans,

There is related work on the control of vehlcies, based on computer perception of visual information, Experiments are being performed with an electrically driven cart equipped wlth a televlslon camera and connected to the computer wlth radio control and televlslon return i Inks,

Finally, a program is being developed to recognize people from their television Images, based on measurements of reletive locations of certain features,

The following sections describe this work in a bit more detail, The coverage is uneven for two main reasons, First, the various research projects are at different stages of maturity, so that there is naturally more to say about some than others, Second, these descriptions are the products of the individual researchers, each of whom has a different viewpoint and verbosity level, We have chosen to publish lt as it comes, rather than editing to a uniform viewpoint and depth,

2. THEORY

2.1 REPRESENTATION THEORY [John McCarthy and Erlk Sandewall]

Our research In thls area Is based on the paper by McCarthy and Hayes (1969). An overview of the problems In this area Is glven ln the last Project Tecnnlcal Report (AIM-87).

During thls year, we have concentrated on finding a way of expressing KNOW-like concepts ('knows', 'believes', 'remembers', etc.) In first-order predicate calculus, Thls Is presently one of the Important problems In representation theory, Some of the difficulties are discussed In (McCarthy and Hayes),

Sandewall has lately developed a new approach to this problem, and we believe it will handle some of the previous difficulties. The basic Ideas are as follows,

We use a many-sorted logic where OBJECTS (including persons) and PROPERTIES are two sorts, A relatlon IS asslgns a property to an object, e.g.

IS(peter, tall)

we may have functions from properties to propertles

IS(peter, Very(tall))

or more complicated functions

IS(peter, More(tall, john))

[which Is Intended to mean 'Peter is tall ler than John'], or

IS(321-0578, OF(telephone-number, peter))

[which Is Intended to mean '321-0578 Is the telephone number of Peter!, We can then express 'John knows Peter's telephone number as

IS(John, Knowing(OF(telephone-number, peter)))

where 'Knowing is a function from propertles to propertles, Using a more convenient Infix notation for blnary relations and functions, we can write the same formula as

john IS Knowing (telephone-number OF peter)

Next, In order to handle 'John knows that ...'-type sentences, we Introduce a third sort, EVENTS, and a binary function WERE which maps objects * properties into events, Then 'Dick believes that Peter's telephone number Is 123-4567' can be phrased (using Infix notation) as

dick IS Believing (123-4567 WERE telephone-number OF peter)

We believe that the approach outlined here Is adequate for handling most of the KNOW-like modals, We are now implementing thls approach In Stanford LISP, A file of axloms for various verbs and other functions has been set up and Is being extended, This file uses a convenient, easy-to-read Inf lx nctatlon, llke the one used above, A Program which translates this notation to the Input notation required for the QA3 theorem-proving program has been written, A similar Interface with D, Luckham's proof checker (Memo AIM-1031 is In a late debugging stage, We plan to do experiments wlth question-answering and problem- solving during the spring quarter,

For problem environments where transitions between situtations must be described (as in most Of 'problem-solving'), It may be necessary to add a situation variable as a third argument of the relation IS, With this convention, the notation described above Is a pure descendant of earlier work here on representation theory,

REFERENCES

1. J, McCarthy and P, Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence" In D, Michie (ed), Machine Intelligence 4, American Elsevier, New York, 1969,

2. J, McCarthy, et al, "Project Technical Report", AIM-87, Stanford Artificial Intelligence Project, June 1969,

3. J, Allen and D, Luckham, "An Interactive Theorem-Proving Program", AIM-103, Stanford Artificial Intelligence Project, October 1969,

2.2 MATHEMATICAL THEORY OF COMPUTATION [John McCarthy, Edward Ashcroft, Zohar Manna, Stephan Ness, Shigiru Igarashi, David Wyethe]

The reasons for expecting useful results from research oriented at using proof checkers to assist and replace debugging are briefly as follows,

1, Since a computer program Is a mathematical object, its specifications can often be stated In purely mathematical terms and the fact that it meets them is often a purely mathematical fact,

2, The specifications that the program must meet can usually be stated much more briefly than the program Itself and are much more clearly understandable than the program, Of course, this Is dependent on having a good notatlon for expressing these specif icatlons, Moreover, many partial specifications such as that tne program shall not go into a loop or use storage not asslgned to it are particularly easy to state,

3, The Informal arguments In the mind of a programmer as he writes are never deep and therefore should be readily expressible formally given a suitable language for doing so,

4, For the Same reason, a computer program that checks these proofs of correctness should offer no difficulties of principle,

5, The confidence of a user in a Program that has been proved to be correct will be much greater than that in a program that has merely been tested on a number of cases, We look forward to the tlme when no-one will pay money for a program that is not proved to meet Its specifications,

The road to this goal has quite a few Practlcai difficulties that the proposed research Is aimed at overcoming, Among these are:

1, A convenient formalism for writing the specifications does not yet exist,

2, The formal I anguages of mathematical logic are really designed for provlng metatheorems rather than for convenient use, We sha I! have to devise a language allowing many modes of expression tnat are presently only used Informally,

3, The practical use of a proof checking system by Programmers wi I I require an an interactive system in which each step of the reasoning is checked as soon as possible, It may even be best that the programmer be given the abi I Ity to check assertions about statements or subprograms as soon as he writes the subprogram In question, This may be especially useful for the standard specifications of non-looping and non-interference with other storage,

4, Finally and perhaps most important, there is yet much work to be done to get the proper axioms and rules of Inference and to describe the semantics of the formal system we need,

The Present State of Mathematical Theory of Computation

Verifying that a computer program meets its specifications may Involve verifying that It Interacts wlth the world outside the computer correctly, and so may involve knowledge of the properties of the world as well as of the program itself, However, many of the desired properties of a program are purely mathematical consequences of its structure, Indeed, for most programs all the deslred properties such as termination, affecting only Its own storage, havlng a prescribed relation between its inputs and outputs, are mathematical, Therefore, Instead Of debugging a program, i.e, testing Its operation on a limited number of cases, It should, in principle, be possible to prove that the program meets its specifications, Mathematical theory of computation Is concerned with formalizing the properties of computer programs that constitute their correctness, developing techniques for expressing and proving correctness, and also with computer programs that can verify proofs of correctness, and even with programs that might generate proofs of correctness,

We cannot expect. to be able to find a general technique for generating proofs of correctness of programs, The problems are In general undecidable and can Involve the solution of arbitrary problems of the field of knowledge the program Is concerned with, An extreme example of this would be given by a program for adding four tnteoers that did it in the usual way except when the four numbers came out to be a counter-example to Fermat's last theorem, in which case it returns 0, Whether the program adds correctly depends on whether Fermat's last theorem Is true, Incidentally, we also see that there is no way of debugging the program since it will work correctly on ail cases except for counter-examples to Fermat's last theorem. Although this example Is artificial, the domain of Interesting programs has the property that there Is no general method of proving programs correct and, moreover, no general way of constructing test cases for debugging,

On the other hand, expressing the correctness of programs formally and formally verifying proofs of their correctness seems intuitively to be a feasible goal, When a person writes a program, he has an intuitive idea of what the program is supposed to do, and as he writes It , he has intuitive reasons for expecting the steps of the program to do the right thing, The errors made are general ly oversights and when an overslght is pointed out, the programmer usually understands his mistake even though it may not be apparent to him how to change the program so that It will meet the specifications, Therefore, our problem is merely one of expressing in a formal mechanically checkable way reasonings that, in themselves are not difficult,

There is a close formal relation between mathematical theory of computation as descrl bed above and the older theory of computable functions, In fact, they deal with the same objects since computer programs are entirely equivalent to Turing machlnes or any of the other formallsms they use to describe algorithms, Unfortunately for

US, their objectives are so different from ours that only the most elementary of their results are relevant to proving that particular programs meet their specifications. Namely, the theory of computability is concerned with what classes of problems can be solved by computable functions and various subclasses and generalizations thereof. It is never concerned with the properties of specific algorithms.

Another related field is automata theory. Sometimes its results are relevant for theory of computation, but it too is not often concerned with the properties of specific Programs.

A final relevant theory is formal syntax. This has been helpful in defining the set of admissible strings of programming languages, but the definitions have usually not taken a form that permits an equally formal definition of their semantics. Some recent work of Knuth (1968) deals with the problem of assigning semantics to context-free languages by definitions that parallel the productions that define the syntax. It turns out in some of Knuth's examples, that even when the syntax is context-free, the semantics of an expression depends on the context.

The goal of using mathematical theory of computation to replace debugging was first stated in (McCarthy 1963a), but some results significant for realizing this goal were obtained earlier. In (Yanov 1960), programs were represented by block schemata which are essentially the same as flow charts. A notion of equivalence of block schemata was defined and a decision procedure for this equivalence was given. The Soviet school of "theory of programming", as they call it, has mainly concentrated on developing and elaborating Yanov's notion. Equivalence of schemata is stronger than equivalence of programs in that two equivalent programs (in the sense that for the same inputs they give the same outputs) may have inequivalent schemata. In fact, the transformations that preserve Yanov equivalence are so limited that no one has tried to apply them to actual programs.

Yanov schemata can be regarded as Algol programs using only assignment-statements of the form x := f(x) and go to's of the form

if p(x) then go to a;

with the further restriction that there is only one variable x in the whole program. Equivalence of schemata means input-output equivalence for all domains of the variable and all interpretations of the function and predicate letters. Yanov showed that this equivalence is decidable even with certain additional conditions about the effect of executing the assignment statements on the values of the predicates. See (Rutledge 1964) for details.

The decidability of Yanov schemata goes away, however, if we allow two different variables or impose algebraic relations among the functions. This is shown in (Luckham and Park 1964). Semi-definitive results in this direction were obtained by Manna (1968), who allows first order axioms to relate the functions and prealcates and shows that termination is equivalent to the truth of a

formula of predicate calculus. He also can treat relations between input and output and equivalence under the conditions that both programs terminate for given inputs.

It turns out, however, that almost all interesting results concerning actual programs require mathematical Induction for their proof. In first order formulations this means that axiom schemata and not just finite sets of axioms are required. In fact, the termination of a Program Is equivalent to an axiom schema; the usual induction schema of first order arithmetic is equivalent to the termination of a program that starts with a given integer and counts down until It reaches zero.

The first formalized use of induction for proving equivalence and other properties of programs is in (McCarthy 1963a) wherein a method called- recursion Induction was given for proving equivalence and relations between the arguments and values of functions defined recursively using conditional expressions. (This paper is probably also the first to describe the potential use of mathematical theory of computation to replace debugging.) Examples of proofs were given for the elementary functions of Integers and also for LISP functions of symbolic expressions. The method was further developed and extended to Algol-like programs In (McCarthy 1963b).

Floyd(1967) gave another method of Proving properties of Algal-like programs. The two methods are equally powerful but apparently not equivalent; Floyd's method is more intuitive for Algol-like programs. Neither method treats termination and require that the termination of programs be established by other (at that time unformalized) methods.

Manna (1969a and b) recast Floyd's formalism radically so as to be able to treat termination and McCarthy discovered how to recast his Own formalization so as also to treat termination. Manna and Pnueli (1969c) showed how to unify the recursive function and Algoi Ic program results. Each formalism has Its advantages for different classes of problems. (The recursive function formalism is more easily manipulated mathematically when the problem fits that form, but this is not always the case.)

A second main line of development of mathematical theory of computation is the semantic definition of programming languages. This makes possible proofs of the correctness of translators. This started with (McCarthy 1963b) which introduces the notion of abstract syntax, a syntactic tool that permits the convenient definition of semantics and the description of translators. This paper also shows how to define semantics by a recursive interpreter using a state vector and also gives a definition of the correctness of a translator. This is applied to defining the semantics of a subset of Algol In (McCarthy 1966) and to proving the correctness of a translator for arithmetic expressions In (McCarthy and Painter 1967). Painter (1967) extended the method to Proving the correctness of translators for simple Algolic programs. These proofs Involved rather complicated formalism which discouraged attempts to apply the method to more complicated languages. A new approach by Morris (1970

we hope) seems to avoid these complications,

The ideas of abstract syntax and state vectors were used by the IBM Vienna group (Lucas et, al, 1968) to define the semantics of PL/I, but they got into severe complications In the description, partly due to the many anomalies in PL/I, The mathematical properties of their notation have not as yet been formalized and it is not c I ear that they w I I be able to treat correctness of translators,

Other purely descriptive formalisms are those of Landin (languages are described by giving rules for translation into lambda calculus), Van Wijngarten (languages are described by giving their data as strings and giving Markov-like algorithms for the elaboration of the computation), the formalism of de Bakker (1968), and the lambda calculus formalism of (Ledgard 1969). In our opinion, the usefulness of a descriptive formalism Is, in the long run, determinea by its usability for the description of transia-tion procedures and proofs of the I r correctness, In this connection, the results of Donovan and Ledgard (1967) are interesting in that they apply Donovan's canonic systems to the complete syntact I c description of Programming languages, including the restrictions (such as that an identifier used in a go to statement must appear exactly once as a label) that cannot be described in Chomsky type languages, Donovan's methods a I so al low the relation between a source program and certain translations of it to be defined, We hope to be able to use some of his ideas In connection with abstract syntax,

Another relevant line of work is the development of partial predicate calculus in mathematical logic, The formalisms described In (Wang 1961), (McCarthy 1963_c), and (Hayes 1969) provide a formalism in which proofs Of correctness of programs can be made (Manna and McCarthy, 1970) because computable functions can be substituted In their valid formulas without first determining that they are total, I.e, that the algorithms terminate,

The development Of resolution methods of theorem proving starting with (Robinson 1965) provide a basis for writing proof checkers that can also do Part of the work of constructing the proof, Proof checkers in-general-are discussed in (McCarthy 1944) and one for resolution proofs is described In (McCarthy 1965),

In the last year, a number of new approaches have appeared, Burstall (1970) applies first order logic to describing both the syntax and the semantics of an extensive Algol subset, By using the axiom schema of mathematical induction, he can prove the correctness and termination of simple programs,

Scott (1970) has defined a partial function theory analogous to set theory that contains partial functions of higher types, The undef i ned element of each type and a partial ordering according to relative definedness are introduced, Recursion Is introducea by a combinator that gives the least fixed point of a partial function, The axioms include a direct generalization of recursion induction,

Park (1970) has used the Knaster-Tarski fixpoint theorem for complete lattices to give a fixpoint theorem for predicate calculus formulas, By appropriate choice of formulas, the theorem gives

i) the results of Manna and Pruell (1969c) for recursively defined functions
ii) a statement of recursion induction, and
iii) certain interesting properties of the interpretations satisfying the formula (e.g. the appropriate Induction principle for such interpretations),

Recent work (1970) in the AI Project:

Ashcroft and Manna (1970) have extended the methods of Floyd and Manna to parallel programs. Although the Programs considered are syntactically simple, they do exhibit interaction between asynchronous parallel processes. The formalization can be extended to more complicated programs, The method is based on a transformation of parallel programs into non-deterministic programs, the properties of which have been formalized in Manna (1970a). The nondeterministic programs are in general much larger than the parallel Programs they correspond to, A simplification method is therefore Presented which, for a given parallel program, allows the construction of a simple equivalent parallel program, Whose corresponding non-deterministic program Is of reasonable size,

Manna (1970b) demonstrates conclusively that all properties regularly observed in programs (deterministic or non-deterministic) can be formulated in terms of a formalization of 'partial correctness', Ashcroft (1970) 'explains' this by formuiatlng the notion of an lntuitiveiy 'adequate' definition (in Predicate calculus) of the semantics of a language or a Program, He shows the relationship between a formalization or partial correctness of a program and an 'adequate' logical definition of its semantics,

Manna and Ness (1970) have formalized techniques for proving the termination of algorithms using well-ordered sets, They give effective Sufficient conditions for termination as well as non-effective necessary and sufficient conditions,

Manna end McCarthy (1970) formalize properties of Lisp-like programs using Partial function logic, where the partial functions occuring in the formulas are exactly those computed by the programs, They distinguish between two types of computation rules--sequential and parallel,

Among work In progress, Igarashi is developing further axiomatic methods for the semantics of Algol-like languages, mainly based on his earlier studies, but allowing the methods of Floyd to be carried out within the formalism, A metatheorem is included which can be interpreted as a proof of correctness of a conceptual compiler for the programs treated by the formalism,

In summary, mathematical theory of computation has become a lively discipline, and much of the work is oriented In directions that will eventually enable us to replace most debugging,

References

1. E.A. Ashcroft, (1970), "Mathematical Logic Applied to the Semantics of Computer Programs,,, Ph.D. Thesis, to be submitted to Imperial College, London,

2. E.A. Ashcroft and Z. Manna, (1970), "Formalization of Properties Parallel Programs,,, Stanford Artificial Intelligence Project, Metro AIM-110,

3. R.M. Burstall, (1970), "Formal Description of Program Structure and Semantics In First Order Logic", In Machine Intelligence 5, Edinburgh Unlverslty Press, 79-98,

4. J, W, de Bakker, (1968), "Axiomatics of Simple Assignment Statements,,, Publication of Mathematrsch Centrum, Amsterdam, June 1968,

5. Donovan and Ledgard (1967), "A Formal System for the Specification of the Syntax and Translation of Computer Languages", Proc, FJCC 1967,

6. R.W. Floyd, (1967), "Assigning Meanlng to Programs,,, Proceeding of Symposia In Applied Mathematics, American Mathematical Society, Vol, 19, 19-32,

7. P, Hayes, (1969), "A Machine-Oriented Formulatlon of the Extended Functional Calculus", Memo Al-86, Artificial Intelligence Project, Stanford University, April, 1969,

8. D, Knuth, (1968), "Semantics of Context-free Languages,,, Mathematical System Theory, Vol, 2, No, 2,

9. H, Ledgard, (1969), "A Formal System for Defining the Syntax and Semantics of Computer Languages", Ph.D. Thesis In Electrical Engineering, M.I.T., April 1969,

10, P, Lucas, K, Alber, K, Bandat, H, Bekic, P, Oliva, K, Welk, G, felsel, (1968), "Informal Introduction to the Abstract Syntax and Interpretation of PL/1", Technical Report, IBM Laboratory, Vienna, June 1968,

11, D.C. Luckham and D.M.R. Park, (1964), "The Undecidability of the Equivalence Problem for Program Schemata", Report NO, 1141, Bolt, Beranek, and Newman, Cambridge, Massachusetts,

12, Z, Manna, (1968), "Termination of Algorithms", Ph.D. Thesis, Computer Science Department, Carnegie-Mellon University,

13. Z. Manna, (1969a), "Properties of Programs and the First Order Predicate Calculus", J, ACM, April 1969.

14. Z. Manna, (1969b), "The Correctness of Programs,,, J, System and Computer Sciences, May 1969,

15. Z. Manna, (197Øa), "The Correctness of Non-deterministic Programs,,, Artificial Intelligence Journal, Vol, 1, No, 1,

16. Z. Manna, (197Øb), "Second-order Mathematical Theory of Computation", Proc. ACM Symposium on Theory of Computing (May, 197Ø),

17. Z. Manna and J, McCarthy, (197Ø), "Properties of Programs and Partial Function Logic", In Machine Intelligence 5, Edinburgh University Press,

18. Z. Manna and S, Ness, "On the Termination of Markov Algorithms", Proceedings of the third Hawaii International Conference on Systems Sciences, January 197Ø.

19. Z. Manna and A, Pneuli, (1969c), "Formalization of Properties of Hecursively Defined Functions,,, Proc, ACM Symposium on Computing Theory, May 1969. To appear in the Journal, ACM (July 197Ø),

2Ø. J. McCarthy (1963a), "A Has is for a Mathematical Theory of Computati on", in Braffort and Hirschberg (eds), Computer ProgrammingandFormal Systems, North-Holland, Amsterdam,

21. J, McCarthy, (1963b), "Towards a Mathematical Theory of Computation,', Proc, IFIP Congress 62, North-Holland, Amsterdam,

22. J, McCarthy, (1963c), "Predicate Calculus with 'Undefined' -as a Truth-Value", A.I, Memo, #1, Stanford Artiflclal Intelligence Project, Stanford University, March 1963,

23. J, McCarthy, (1964), "A Proof-Checker for Predicate Calculus", A.I. Memo #27, Stanford Artiflclal Intelligence Project, Stanford University, .

24. J, McCarthy, (1965), "Problems In the Theory of Computation", Proc, IFIP Congress 65,

25, J, McCarthy, (1966), "A Formal Uescriptlon of a Subset of Algol", pp 1-12 of Formal Language Description Languages for Computer Programming, T,B, Steel, Jr, (editor), North-Holland Publishing Co,, Amsterdam,

26. J, McCarthy and J, Painter, (1967), "Correctness of a Compiler for Arithmetic Expressions", In Amer. Math, Soc, Proceedings of Symposia in Appl led Mathematics, Mathematical Aspects Aspects of Computer Science, New York,

27. L. Morris, (1970), Forthcoming Ph.D. Thesis, Stanford University.

28. J. Painter, (1967), "Semantic Correctness of a Compiler for an Algo I - like Language", Ph.D. Thesis In Computer Science, Stanford University.

29. D. Park (1970), "Fixpoint Induction and Proofs of Program Properties", In Machine Intelligence 5, Edinburgh Unlversity Press.

30. Robinson, (1965), "A Machine-Oriented Logic Based on the Resolution Principle", JACM, 12, 23-41.

31. D. Scott, (1970), unpublished memo.

32. J.D. Rutledge, (1964), "On Ianovs Program Schemata", JACM, Vol. 11, No. 1, January 1964.

33. H. Wang, (1961), "The Calculus of Partial Predicates and Its Extension to Set Theory", Zeitschr f. Math., Logik und Grundladen a. Math., pp 283-288.

34. Y.I. Yanov, (1960), "The Logical Schemes of Algorithms", Problems of Cybernetics, Vol. 1, translated from the Russian by Nadler, Griffiths, Kiss, and Muir, Peraamon Press, New York.

3. MODELS OF COGNITIVE PROCESSES

3.1 DENDRAL PROJECT [Edward Feigenbaum, Joshua Lederberg, Bruce Buchanan, Allan Delfino, and Georgia Sutherland]

1. Description of the DENDRAL Project

The DENDRAL project aims at emulating in a computer program the inductive behavior of the scientist in an important but sharply limited area of science, organic chemistry. Most of our work is addressed to the following problem: given the data of the mass spectrum of an unknown compound, induce a workable number of plausible solutions, that is, a small list of candidate molecular structures which explain the data of the mass spectrum. In order to complicate the task, the DENDRAL program performs three steps. First, the program searches the spectrum for clues to the nature of the unknown structure. Second, the program generates a list of all structures with the properties specified by the first step. And, finally, the program deduces the mass spectrum predicted by the computer theory of mass spectroscopy for each of the candidates, and selects the most productive hypothesis, i.e., the structure whose predicted spectrum most closely matches the data.

The program has been completely described in a series of reports (8, 11, 14, 15). The publication of these reports indicates the status of the heuristic DENDRAL program in the artificial intelligence field. Several other reports published in the chemical literature (12, 13, 17, 18) indicate its importance to the field of organic chemistry.

2. Recent Work on Heuristic DENDRAL

Recent work by members of the DENDRAL project demonstrates that the project is expanding on many fronts. Some work has been devoted to improving the internal workings of the program. Other work has expanded the domain of the Program so that more types of structures can be constructed. A graphics program has been written to allow any user to observe the internal workings (heuristic search) of the Structure Generator. A new Inference Maker has been written which, for a limited class of structures, totally replaces the previous program. Chemist/programmers have formalized some of their rules of Inference. A separate effort has been launched to investigate and write a program to perform organic synthesis. A program has been written to converse with chemists to extract their ideas about mass spectroscopy and write actual computer code without the intervention of a programmer. Another program has been written to catalog mass spectra. And a start has been made at describing a new program, colloquially known as "Meta-DENDRAL". All these efforts are given separate treatment in the following paragraphs.

To simplify the modification of the Program's theory of mass spectrometry, the Predictor section of the program has been restructured. The re-organization is along the lines of what is called "situation-action rules", (Ref. 19). The situation-action

rules themselves reside in tables, and are not woven into the actual computer code, This makes them available for Inspection and change in a relatively simple way, In fact, the dialog program described in paragraph 2f below orovjdes a way for chemists (and others) to change these tables, The writing of the Predictor has been undertaken by Dr, Bruce Buchanan, with the assistance of Isu Fang, a graduate stuoent In Computer Science.

The Structure Generator sub-program has been expanded by the addition of a generator for rings. The ring generator constructs a|| simple rings, and passes them to the regular Structure generator, which uses them as "superatoms" jn generating all chemical structures that are either acyclic or monocyclic, Since the large majority of "interesting" chemical structures fali In this class, this expansion of the Structure Generator has made the Program more lnterestlng from the chemists' point of view, This work was completeaby Mrs, Georgia Sutherland.

Tne heuristic search aspects Of the Structure Generator have been the subject of many discussions. It is felt that the search could be Improved, The first step toward thls goal Is to be able to observe the present search process; and with this in mind, a graphic program has been written to display the search tree on a cathode-ray tube, allowing the user quite a bit of freedom to explore this tree at will and to record his suggestions for "smarter" search, This Program Was written by Mike Rogson, a graduate student, nith the collaboration of Georgia Sutherland for corresponding changes to the Structure Generator, The graphic program itself runs on the IBM 360/91 at the Stanford Linear Accelerator Center, Although the Structure Generator program will run on this computer also, there is, at Present, no way for the two programs to be executed simultaneously, Several system programmers are working to make It possible for the, programs to communicate with each other In real time,

The reactlon of chemists who are also programmers to the workings of the Heuristic DENDRAL program has led to the writing of a new program, which, for the class Of "saturated allphaticmonofunctional" compounds, obviates the need to execute the previous Preliminary-Inference- Maker → Structure Geherator → Predictor steps usec by Heuristic DENDRAL In the past, This new Inference Maker works from the mass spectrum directly, and infers only those structures for which there Is direct evidence, The Inference Maker Is a planner, but usual ly its plans specify the structure In sufficient detail that only one structure is implied, Although the class of compounds for which thls can be done is quite limited, it is very lnterestlng that this Is Possible at all, and It represents a formalization of the rules of mass spectroscopy which d/d not exist a year ago, This effort has brought forth two new programs which may be useful in extending this new Inference Maker to other types of structures, One Is a "superatom generator" which constructs a list of unique, mutual ly exclusive, var lants of a functional group (a small but chemically interesting set of atoms) to use as superatoms In the regular Structure Generator, The other Is a "rule generator" which generates the mass spectral rules associated with a limited

class of functional groups, This work was performed by Mr, Allen Delfino of our project, with the assistance of Dr, Armand Buchs of the Chemistry Department,

A project separate from Heuristic DENDRAL, but of great interest, Is the development of a computer program to perform organic synthesis of chemical structures, Dr, Malcolm Bersohn of the University of Toronto, visiting with the DENDRAL project, has undertaken this work, The program will represent a computer simulation of a chemical laboratory in which desired and-product substances are created by the successive reactions of other substances, possibly with the aid of catalysts, Like chess-playing and some other artificial intelligence Problems, synthesis is a graph traversal problem, The graph is of such enormous size that It must be pruned by the program, The breadth vs, depth decision In the graph exploration is made dynamically, This project Is being aided by funds from NIH,

Dr, Bruce Buchanan, with the consultat-lon of Dr, Alan Duffield of the Chemistry Department, has written a program to "converse with a chemist", This program has knowledge of a few primitive chemical concepts and mechanisms, The chemist, interacting with the program, describes complex processes In terms of these primitives, The program, in turn, represents these processes as program statements to be used by the Predictor as additional rules governing the mass spectrometric behavior of new classes of compounds,

Mr-, Allan Delfinohas written a program to catalog the peaks of mass spectra, The Purpose of this program Is to aid chemists, and, hopefully, Inference programs, In finding similar and dissimilar data patterns In order to determine the mechanisms underlying mass spectroscony, Classes of common peaks can be defined In various ways, For examp'le, all peaks at mass X can be collected and compared, or all peaks at mass V-X (where M is the molecular weight, varying from spectrum to spectrum) can be collected, or all peaks at mass M-X-Y, etc, At present, a chemist must Inspect the output of thls program for interesting results; It would be desirable for a computer program to be able to perform this part of the task also,

Meta-DENDRAL will be a program which can generate alternate theories of mass spectroscopy, analogous to -the way In which the Heuristic DENDRAL program generates alternate structures to explain a given mass spectrum, A theory of mass spectroscopy is built upon the primitives described in paragraph 2f, to explain the mechanisms mentioned In paragraph 2g, As achemist Interact6 with the dialog program he Is building hls theory of mass spectroscopy, But there are possibly other theories, and the Job of Meta-DENDRAL will be to generate these automatically by computer,

3, Summary

The work of the Heuristic DENDRAL project has been directed toward the problems of programming a computer to do inductive Inference, The task domain of organic chemistry lends Itself well to these problem8 since it Is a relatively formal and well-defined domain which,

nevertheless, lacks an axlamatlc treatment to make Its problems purely deductive,

In the course of exploring inductive Inference by machine, we have met many well-known problems of artificial intelligence. Among others, these include problems of heuristic search, representation, man-machine communication, and efficient design. The level of sophistication of the program In its task domaln'and the structure of the domain Itself are allowing us to explore these problems, For Instance, now that the program can use heuristics routinely to gulde search through the hypothesis space, we have begun exploring the efficacy of alternative seach strategies, Or, now that the program can use a complex theory (of organic mass spectroscopy), we have begun experimenting with ways of manipulating that theory,

References

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 (10)).
   1a. Part I, Notational algorithm fo'r tree structures (1964) CR-57029
   1b. Part II, Topology of cyclic graphs (19653 CR-68898
   1c. Part III , complete chemical graphs; embedded rings in trees (1969).

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

3. J. Lederberg, "Topological Mapping of f OrganicMolecules", Proc. Nat, Acad. Sci,, 53:1, January 1965, pp. 134-139,

4. J. Lederberg, "Systematics of organic molecules, graph topology ard Hamilton circults. A general outline of the DENDRAL system," NASA CR-48899 (1965).

5, J. Lederberg, "Hamilton Circults of Convex Trivalent Polyhedra (up to 18 vertices)", Am, Math. Monthly, May 1967,

6. J, Lederberg and E.A. Felgenbaum, "Mechanization of Inductive Inference In Organic Chemistry", In B, Kleinmuntz (ed.) Formal Representations for tiuman Judgment, (Wiley, 1968). Also, Stanford Artifical Intelligence Project Memo No, 54, August 1967) .

7. J, Lederberg, "Online computation of molecular formulas from mass number", NASA CR-94977 (1968).

8. B,G. Buchanan, G,L. Sutherland, and E,A. Felgenbaum, "Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses In Organic Chemistry", In Macnlne Intelligence 4 (B, Meltzer and D, Michie, eds,) Edinburgh Unlverslty Press (1969). (Also,

Stanford Artificial Intelligence Project Memo No, 62, July 1968),

9. E.A. Felgenbaum, "Artificial Intelligence: Themes in the Second Decade", In Final Supplement to Proceedings of the IFIP68 International Congress, Edinburgh, August 1968, (Also, Stanford Artificial Intelligence Project Memo No, 67, August 1968).

10. J. Lederberg, "Topology of Molecules", In The Mathematical Sciences- A Collection of Essays, (ed,) 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,

11. G. Sutherland, "Heuristic DENDRAL: A Family of LISP Programs", to appear In D. Bobrow (ed.), LISP Applications, (Also, Stanford Artificial Intelligence Project Memo No, 80, March 1969),

12. 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).

13. A.M. Duffield, A.V. Robertson, C. Djerassi, B.G. Buchanan, G.L. Sutherland, E.A. Felgenbaum, 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).

14. 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 Project Memo No, 99, September 1969),

15. J. Lederberg, G.L. Sutherland, B.G. Buchanan, and E.A. Felgenbaum, "A Heuristic Program for Solving a Scientific Inference Problem: Summary of Motivation and Implementation", Stanford Artificial Intelligence, Project Memo AIM-104, November 1969.

16. 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,

17. 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).

18. 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", submitted to the Journal of the American Chemical Society, March 1970,

19. D.A. Waterman, "Machine Learning of Heuristics", PhD Dissertation (Stanford University Computer Science Department), (Also, Stanford Artificial Intelligence Project Memo No, 74, December 1968).

3.2 GRAMMATICAL INFERENCE [Jerome Feldman and Alan Biermann]

Objectives: To study the problem of grammatical inference. A grammatical Inference system is given a set of strings which nas been chosen in some random way from a formal language, such as a context-free language, (It may also be given strings which are not In the language and so designated). The system Is to make a "reasonable" inference of the grammar for the language,

Some problems under consideration:

1). Develop a criterion for choosing the most "reasonable" grammar for a set of strings among a set of acceptable grammars.

2). Given a criterion from 1)., does an algorithm exist for choosing the most "reasonable" grammar from 2 class of grammars? If SO, find the algorithm.

3). Suppose that strings from some unknown language are sequentially presented to an inference device, and at each instant in time, the device makes an Inference of the most "reasonable" grammar based on current information. Will the guesses that the machine makes converge to the correct answer and, if so, how soon will this happen?

Current Work

Feldman has originated the concepts of grammatical complexity and derivational complexity as vehicles for measuring how well a grammar "fits" a set of strings, The grammar with the smallest combined grammatical complexity and derivatlonal complexity for the glven set of strings Is chosen as the most "reasonable" Inference, The complexity concepts are tied directly to probabillty theory and Bayes Theorem, (See Feldman et al, 1969,)

Feldman (1969) has studied a very general class of complexity measures and shown that the least complex choice Of a grammar from any enumerable class of grammars (which are general rewrltlng systems) can be found, and he gives an algorithm for discovering it,

The problem of learning or converging on the correct grammar for some unknown language as more and more strings from the language are observed has been studied. Feldman, et al (1969) have defined the concept of approachability In the limit, which Is a weaker type of convergence crlterlon than identifiability In the limit studied by Gold (1967) in an earlier paper. It Is shown that for any class of grammars from the class of general rewrltlng systems, there Is a machine which will approach the correct grammar In the limit,

Horning (1969) has applied Bayslan theory to the grammatical Inference problem and has produced an algorithm which finds the "most probable" grammar for a set of strings, He also studied convergence behavior and the problem of inferring stochastic grammars,

Algorithms for grammatical inference which are constructive rather than enuneratlve in nature are currently under investigation, Several algorithms for flnlte-state grammar inference have been developed by

Biermann and Feldman and their properties are being studied. Attempts are being made to develop algorithms for context-free language inference.

REFERENCES:

1. Biermann, Alan W. and Jerome Feldman, "On the Synthesis of Finite-State Acceptors", AIM-114, Stanford Artificial Intelligence Project.

2. Feldman, Jerome A., "First Thoughts on Grammatical Inference!', AIM-55, Stanford University, August 1967.

3. Feldman, Jerome A., James Gips, James J. Horning, Stephan Reder, "Grammatical Complexity and Inference? AIM-89, Stanford Artificial Intelligence Project, June 1969.

4. Feldman, Jerome A., "Some Decidability Results on Grammatical Inference and Complexity", AIM-93, Stanford Artificial Intelligence Project, August, 1969, (To appear in Information and Control>.

5. Gold, E. Mark, "Language Identification in the Limit", Information and Control, vol. 10, 1967.

6. Horning, James Jay, "A Study of Grammatical Inference", AIM-98, Stanford Artificial Intelligence Project, August, 1969.

7. Solomonoff, R., "A New Method for Discovering the Grammars of Phrase Structure Languages", Information Processing, June, 1959.

8. Solomonoff, R.J., "A Formal Theory of Inductive Inference," Information and Control, vol. 7, 1967.

9. Watanabe, Satoei, "Information-Theoretical Aspects of Inductive and Deductive Inference", IBM Journal, April, 1960.

3.3 COMPUTER SIMULATION OF BELIEF SYSTEMS [Kenneth Colby, Frank HI If, Malcolm Newey, Roger Schank, Dave Smith, Larry Tesler, and Sylvia Weber]

Kenneth Mark Colby, M.D., who is a Senior Research Asscclate ln the Computer Sclence Department, terminated hls prlvate practlce of Psychiatry to devote full time to lnvestigations ln thls area of computer slmulation. The National Institute of Mental Health sponsored two projects under Dr. Colby's direction, One of these Is a Research Career Award and the other is a research project whlch continues the Investigations In whlch hls group has been engaged for the past seven years,

A. Introduction and Specific Alms:

The clinical problems of psychopathology and psychotherapy require further investigation since So little Is known about their essential processes. Some of this ignorance stems from a lack at a basic science level of dependable know) edge regarding higher mental processes such as cognltlon and effect, The research of the Project attempts to approach both the cllnlcal and basic science Problems from the vlewpolnt of lnformation-processlng models and computer slmulation technlques, This viewpoint is exemplified by current work in the fields of cognitive theory, attitude change, belief systems, computer slmulation and artificial lntelligence,

The rationale of our approach to these clinical problems lies in a conceptualization of them as lnformation-processlng Problems lnvolvlng higher mental functions, Computer concepts and technlques are appropriate to this level of conceptuallzation, Thelr success In other sciences would lead one to expect they might be of aid ln the areas of psychopathology and psychotherapy,

The specific alms of this project relate to a long-term goal of developing more satisfactory explicit theories and models of psychopathological processes. The models can then be experimented with ln ways which cannot be carr led out on actual patlents, Knowledge gained in thls manner can then be applied to clinical situations,

B. Methods of Procedure:

We have now gained considerable experience with methods for writing Programs of two types. The first type of program represents a computer model of an lndivldual Person's belief system, We have constructed two versions of a model of an actual patient ln psychotherapy and we are currently writing programs which simulate the belief systems of two normal lndlvlduals, We have also constructed a model of a pathological bel lef system in the form of an artificial paranola, A second type of program represents an lntervlewlng program which attempts to conduct an on-line dialogue Intended to col lect data regarding an lndivldual's Interpersonal relations, We have written two such lntevlewlng programs and at

present we arecollaborating with psychiatrists In writing a Program which can conduct a diagnostic psychiatric interview,

Acomputer model of a belief system consists of a large data-base and procedures for processing the information It contains, The data-base consists O f concepts and bel lefs organized In a structure which represents a n individual's conceptuallzation of himself and other persons of importance to him jn his life space, Th 1 s data Is ccl lected from each individual informant by interviews, Verlflcation o f the model is also carried out In jntervlews in which the Informant is asked to conflrm or disconflrm the outcome of experlments on the particular model which represents hls belief system, Because of the well-known effects of human interviewer bias, the Process of data-collection and verifications should ldeally be carr led out by on- ljne man-machine dialogues and thls is a major reason for our attempt to write interviewing programs, However, the djff iculties In machine Utl lization of natural language remaln great and until this problem Is reduced we must use human interviewers,

We have written one tyne of therapeutic Interactive program which Is designed t o aid language development in nonspeaklng autistic children, We have used it for the past two years on elghteen children with considerable success (80% linguistic improvements), We intend to continue using this program and to Instruct professionals in psychiatry and Speech therapy in how to wrlte, operate and Improve such therapy programs for specific conditions,

C, Significance of this Research:

This research has slgnlficance for the psychiatric, behavloral and computer sciences,

Psvchlatry lacks satisfactory classjflcatjons and explanations of psychopathology, We feel these problems should be conceptuallzed In terms of pathological belief systems, Data collection in psychiatry Is performed by humans whose interactlve effects are believed to account for a large percentage of the unrellabillty In psychiatric diagnosis, Diagnostic Interviewing should ldeally be conducted by computer programs, Finally, the process and mechanisms of psychotherapy are not well understood, S I nce experimentation o n computer models is more feasible and control lable than experimentation on patients, this approach may contribute t o our understanding of psychotherapy as an information-processing problem,

It is estimated that 90% of the data collected In the behavloral sciences- is collected through interviews, Again, a great deal of the variance should be reduced by having consistent programs conduct interviews, Also, this research has slgnificance for cognitive theory, attitude change and social psychology,

Computer science is concerned with problems of man-machine dialogue in natural language, with optimal memory organlzatlon and wl th the search problem in large data-structures, This research bears on these problems a s well as on a crucial problem in artificial

Intelligence, i.e., inductive inference by intelligent machines.

D. Collaboration:

We are collaborating with two psychiatric centers for disturbed children and a local VA hospital. We are also collaborating with residents in the Department of Psychiatry and with graduate students in computer science, psychology, education and electrical engineering.

References

1. Colby, K.M., "Experimental Treatment of Neurotic Computer Programs", Archives of General Psychiatry, 10, 220-227 (1964).

2. Colby, K.M. and Gilbert, J.P., "Programming a Computer Model of Neurosis," Journal of Mathematical Psychology, 1, 405-417 (1964).

3. Colby, K.M. "Computer Simulation of Neurotic Processes", In Computers In Biomedical Research, Vol. I., Stacey, R.W. and Waxman, B., Eds., Academic Press, New York (1965).

4. Colby, K.M., Watt, J, and Gilbert, J.P. "A Computer Method of Psychotherapy," Journal of Nervous and Mental Disease, 142, 148-152 (1966).

5. Colby, K.M. and Enea, H, "Heuristic Methods for Computer Understanding In Context-restricted On-line Dialogues", Mathematical Biosciences, Vol. I, 1-25 (1967).

6. Colby, K.M. "Computer Simulation of Change In Personal Belief Systems," Behavioral Science, 12, 248-253 (1967).

7. Colby, K.M. "A Programmable Theory of Cognition and Affect in Individual Personal Relief Systems," In Theories of Cognitive Consistency, (Abelson, R., Aranson, E., McGuire, W., Newcomb, T., Tannebaum, P. Eds.,) Rand-McNally, New York, N.Y. (1968).

8. Colby, K.M. and Enea, H, "Inductive Inference by Intelligent Machines," Scientia, 103, 1-10 (1968).

9. Tesler, L., Enea, H, and Colby, K.M, "A Directed Graph Representation for Computer Simulation of Belief Systems," Mathematical Biosciences, 2, 19-40 (1968).

10. Colby, K.M, "Computer-aided Language Development In Nonspeaking Autistic Children." Technical Report, No, CS 85 (1967), Stanford Department of Computer Science, Archives of General Psychiatry, 19, 641-651 (1968).

11. Enea, H. "MLISP", Technical Report, CS 92 (1968), Stanford Department of Computer Science,

12. Colby, K.M., Tesler, L., Enea,H. "Search Experiments with the Data Base of Human Belief Structure", Proc, of the International Joint Conference on Artificial Intelligence, Washington' D.C., (Walker and Norton, Eds.), 1969,

13. Colby, K.M., Tesler, L, and Enea, H, "Experiments with a Search Algorithm on the Data Base of a Human Belief Structure," Stanford Artificial Intelligence Project Memo AIM-94, August 1969, Proceecings of the Internatlonal Joint Conference on Artificial Intelligence, Walker and Norton (Eds.), 1969'

14. Colby, K.M, and Smith, D.C., "Dialogues Between Humans and An Artificial Belief System," Stanford Artificial Intelligence Project Memo AI M-97, Proceedings of the Internatlonal Joint Conference On Artificial Intelligence, Walker and Norton (Eds.),1969.

15. Colby, K.M. Critical Evaluation of "Some Empirical and Conceptual Bases for Coordinated Research In Psychiatry" by Strupp and Bergin. International Journal of Psychiatry, 7, 116-117 (1969).

16. Colby, K.M., Weber,S. and Hilf,F. "Artificial Paranoia" (in preparation),

17. Colby, K.M., Hilf, F. and Hall,W. "A Mute Patient's Experience with Machine-Mediated Interviews", Stanford Artificial Intelligence Project Memo AIM-113, March 1970.

18. Hilf, F., Colby, K.M., Wittner, W, "Machine-Mediated Interviewing", Stanford Artificial Intelligence Project Memo AIM-112, March 1970.

3.4 Automatic Deduction [David Luckham and Jack Buchanan]

Research effort In thIS area has been devoted to Improving and extending the interactive theorem-proving system which was reported briefly In Technical Report AIM 87, This program Is now more fully described In AIM 1Ø3, [1].

The position regarding the practical capabilities of this system is as follows. The program has been used to prove a variety of theorems in Algebra and Number Theory including some new mathematical theorems contained In a recent research announcement In the Notices of the American Mathematical Society, [2](see also [3]), Since these announcements are made by abstract and do not Include proofs of the stated results, and not Infrequently contain misprints and errors, this represents a very good test of the usefulness of the system, at least in the area of basic axiomatic mathematics.

These experiments with the program- have crovlded information on a number of different points of research:

iI) The improvement (or lack of Improvement) In proof efficiency and computation time resulting from the use of various combinations of the strategies available to the user c4,5,6,7,8,9,10,11,14], and the best way to use these strategies.

(ii) The adequacies and lnaaequacies of the Interactive system, This includes the formal language for expressing the problem to be proved, and the facilities for (a) surveying the progress of a proof search, (b) directing the proof search, and (c) initiating sub-problems,

(iii) Weaknesses in the proof procedure due to lack of strategies for dealing with certain kinds of inefficiencies (e,g, due to additional rules of inference such as Paramodulation [11] or due to the generation of trivial deductions not eliminated by the usual editing strategies),

This information IS summarized In [1], It turns out that Some of the efficiency strategies which are proposed In [6 and 73 play a crucial role not only In finding proof of theorems In [2], but also In cutting down the computation time Involved,

Specific lines of research being pursued at the moment include the following,

(1) Addition of the proof-tree analyzer for extracting solutions to existential statements from proofs of them, This procedure has now been formulated and will be programmed shortly, It has the advantage Over earlier systems (e,g, [12]) of not In any way affecting the run-time afflcfency of the prover, Such procedures provide a basic link between the prover and information retrieval systems uslng It,

(2) Extension of the formal input language to a multi-sorted predicate logic with identity, This Is a necessary step towards applying the program to more complex problems (e,g, checking fairly sophisticated reasoning for correctness),

(3) The use of natural models (as well as Herbrand mode(s) In (a) the model-relative deduction strategy [6], (b) problems

associated with McCarthy's Advice Taker project [13], and (c) eliminating Irrelevant deductions Of the sort mentioned in (ii) above, It turns out from theoretical analysis that formulating a problem in a more sophisticated logical language (as proposed tn (2)) does not always immediately improve proof efficiency, but can usually be used to make the evaluation of truth (of a statement relative to a model) more efficient, It Is therefore natural to pursue (2) and (3) simultaneously,

(4) Addition of more sophisticated on-line interactive facilities,

(5) "End condition" strategies for quick proofs, These include vine-form proofs [7, 143, and decision procedures for certain classes of problems,

This program has been written in LISP and structured In such a way that add'itions and extensions of the sort mentioned above Can be made relatively easily without major revisions to all parts of the program, and can then he tested on practtcai problems, The program Is intended to be an experimental tool, and ease of modification has beer; emphasized at the expense of (some) efficiency, However, In the light of recant results, it seems that we are now a good step nearer the stage when it will be reasonable to expend programming effort and time on writing an optimal version of thls system for practical use, The program is currently being incorporated into some computer-aided instruction programs for high school mathemat ics at the Stanford Institute for Mathematical Studies in the Social Sciences,

References

1. Allen, J, and Luckham, D, "An Interact/w Theorem-Proving Program", Machine Intelligence 5, (Meltzer, B, and Michie, D, eds.), Edinburgh University Press, March 1970, Also, AIM-103.

2. Chinthayamma, "Sets of Independent Axioms for a Ternary Boolean Algebra", Notices of Amer. Math' Soc., Vol, 16, No, 4, June 1969, 69T-A69, pp, 654,

3. Grau, A.A.,"Ternary Boolean Algebra", Bull, Amer, Math, Soc., 53, (1947), pp, 567-572,

4. Robinson, J.A., "A Machine-Oriented Logic Based on the Resolution Principle", JACM, Vol, 12, No, 1, pp, 23-41, January 1965,

5. Luckham, C,, "Some Tree-Paring Strategies for Theorem-Proving", Machine Intelligence 3, D, Michie (ed), Edinburgh University Press, pp, 95-112, 1968,

6. Luckham, D., "Refinement Theorems in Resoiutlon Theory", Proceedings IRIA Symposium on Automatic Demonstration, Versailles, France, December 16-21, 1968, Springer-Verlag, April 1970; also, AIM- 81,

7. Kleburtz, R., Luckham, D., "Compatibility of Refinements of the Resolution Principle", submitted to JACM; and forthcoming AI Memo,

8. Wos, L., Carson, D., Robinson, G., "The Unit Preference Strategy In Theorem Proving", AFIPS Conf, Proc, 26, Washington, D.C., Spartan Books, pp. 615-621, 1964.

9. Wos, L., et.al., "Efficiency and Completeness of the Set of Support Strategy in Theorem Proving", JACM, Vol, 12, No, 4, pp. 536-541, October 1965,

10. Wos, L., et, al., "The Concept of Demodulation in Theorem-Proving", JACM, Vol, 14, No, 4, pp, 698-704,

11. Hoblnson, G., and Wos, L., "Paramodulation and Theorem-Proving in First-Order Theories with Equality", Machine Intelligence IV, D, Michie (ed), Edinburgh Universlty Press, 1.969,

12. Green, C., "Theorem-Proving by Resolution as a Basis fot Question- Answering Systems, Machlne Intelligence 4, D, Michie (ed), American Elsevier, New York, 1969,

13, McCartny, J., "Programs with Common Sense" ln M. Minsky (ed), Semantic Information Processing, MIT Press, Cambridge, 1968,

14. Andrews, P.B., "Resolution with Merging", JACM, 15, No, 3, pp. 367-381, July 1968.

3.5 MACHINE LEARNING [Arthur Samuel, David Barstow, Ken Hanson, Jon Ryder]

The latest version of the checker program is now thought to be substantially better than previous versions. The rate of improvement has, however, begun to taper off and the amount of machine time required to test each new improvement has increased by an unacceptable amount. It is now apparent that some detailed checker-specific information is needed to make further improvements. Accordingly, we are getting some help from Mr. K.D. Hanson, a well known checker master (currently the West Coast Champion). In a very short period of time, Mr. Hanson has been able to point out several serious deficiencies in the program although it is not always apparent as to how these deficiencies can best be rectified.

The Co program is now playing regular games and is exciting considerable interest. It still does not contain any machine-learning features. The entire program is, of course, in an early formative stage and it is still very much easier to make indicated changes in the program by manual means than to effect the changes through machine-learning techniques. This state of affairs should not continue much longer, and serious thought is now being given to the best way to approach the machine-learning aspects.

4. Speech Recognition

4.1 Machine Learning [Arthur Samuel, Mort Astrahan, Ken Sieberz, George White]

While the work In applying machine-learning techniques to speech recognition has. only recently been undertaken, considerable clarification in the basic problem and the potentialities for practical approaches have been formulated, Already three quite cistlnct approaches have been recognized and some work has been done on each,

One procedure initiated by Dr. Mort Astrahan makes use of multi-level signature tables analogous to the procedure used in the checker program, A second procedure studied by Mr, Ken Sieberz uses a single large signature table. A thlrd procedure being worked on by Dr, George White identifies speech segments by making use of a data-grown tree or discriminate net, reminiscent of the "Epam" approach but actually based on the signature table philosophy.

During thls early Phase In the application of learning techniques to speech, several very Interesting approaches to the speech recognition problem have been uncovered which are of interest in their own right quite apart from machine learning. Dlgresslons to study these approaches have temporarily lessened the amount of effort devoted to learning. Dr, Astrahan is currently preparing a paper on one aspect of this work and Dr, White will shortly be In a posltlon to report on another aspect,

REFERENCE

1. M. Astrahan, "Speech Analysis by Clustering, or the Phony Phoneme Method", Stanford Artificial Intelligence Project, forthcoming memo,

4,2 Analysis and Recognition of Languages for Man-Machine Voice Communication [Gary Goodman3

The absence of an effective algorithm for accurately delimiting and classifying phonemes from the ir acoustic signal necessitates the careful design of languages for man-machine voice communication, It IS a wel I krcwn fact that, given appropriate contexts, allophones of two different phonemes may be nearly 100% a'cousticai ly similar, particularly when the recognizer is a machine, Thus, when recognizing spoken sentences of a language with the aid of its grammar, it is possible that two different, but phonetically similar, sentences "match" the acoustic signal quite wel I, It becomes desirable, when designing I anguages for machine perception, to know if such a phonetic ambiguity could exist, either globally or locally,

This problem of phonetic ambiguity may occur either as a word ambiguity or a word-boundary ambiguity, Consider the two sentences "I seem able to block South" and "I see Mable two blocks South," A word ambiguity exists between the words "to" and "two", The phrases "seem able" and "see Mable" present a word-boundary ambiguity, Careful "hand" or "eyeball" design of the grammar and recognizer can eliminate the obvious ambiguites, but the subtle ones remain, And If the grammar is altered, the tedious anai'ysis must be repeated, A more systematic approach to phonetic ambiguity is needed In both the analysis and recognition phases,

OUTLINE OF THE RESEARCH,

The research undertaken can be describea in the following 'terms,

I, ANALYSIS, Given a BNF definition of a language for man-machine voice communication, express algorithms for thoroughly analyzing the grammar looking particularly at the problems one might encounter in recognizing It due to

1, syntactic ambiguity
2, phonetic ambiguity
3, word-boundary ambiguity,

Some indication of how the user might alleviate these problems should also be given,

II, RECOGNITION, Design a recognizer skeleton which, when given a language definition would efficiently recognize the spoken input and display It on a CRT for acceptance or rejection by the speaker, The rnethpd used should be such tnat it reduces the search space greatly enough to permit real time response with a high percentage of correct retrieval, After defining the recognizer there could be some recoginizer dependent analysis of the grammar which could be fed back to the user in order to improve the recognition,

III, LEARNING AND ADAPTING, The recognizer should not be limited to working well for only one speaker, Therefore a tralning program should be written which would ask the speaker to repeat certain phrases or sentences selected from the language, The stored acoustic

versions of each of these utterences would then be available If needed by the recognition procedure, Ideally, only a subset of the total vocabulary would have to be learned; saving greatly on the amount of memory required,

PREVIOUS WORK,

The straight-forward approaches of Bobrow and Klatt [1968] and Reddy [1967b] have shown that limited recognition of words, phrases, sentences, and connected speech can be done by a computer, The model used by Reddy consists of four stages: segmentation, sound description, phrase boundary determination, and phrase recognition (McCarthy, et, al, [1968]). Recent research by Vicens [1969] used this model to recognize spoken commands taken from the sentences of a finite state grammar, The grammar used was carefully selected to reduce the amblgulty and the recognizer was "hand" tailored to the grammar, Thls work produced segmentation and phrase recognition procedures which glve correct results 85-95% of the time, The areas of sound description and phrase or word boundary determination remain as the major trouble Spots, They become furthur complicated by the fact that the Input may contain errors (Incorrect segmentation) causing missing Or extraneous segments,

Alter C19683 describes a system which uses syntactic constraints to analyze spoken utterences, but has not tested it on real speech Input,

Related work In the area of vlsual perception is Shaw's [1968] use of syntactic constraints In the analysts of two-dimensional pictures and Duda and Hart's [1968] US8 Of dynamic programming along wlth context-direction In the analysis of hand-prlnted Fortran,

TOWARDS ANALYSIS,

Goodman C19701 discusses an investigation of this problem developed in the following way, An algorithm for computing a similarity measure between strings of phonemes was developed, A Program was written which would test a grammar-for belng bounded right context for some degree m,n (McKeeman, et, al, [1970] and Floyd [1964]). Normally this program checks stacking and reduction decisions based on the m,n context and reports any confl icts, However, the identity function on strings was replaced by the new similarity function, The program then reports the similarity of the phoneme strings In the m,n context when maklng decisions, SInce there would be many of these, the program is set to save and print only the 20 conflicts wlth highest slmllarlty, After examining these conflicts and the productions of the grammar) the grammar may be altered ln a manner which will reduce the ambiguity causing the conflicts, Thls new grammar may now be tested and the process iterated until all conflicts are below an acceptable limit, The program described has been used to design a desk calculator grammar which will be used in future research.

TOWARDS RECOGNITION.

A recognlzer which uses gross but conslstant features of vowel, frlcltive, and stop-like segments of the acoustic signal together with the grammar to form a "plan" has been wrltten, Prellminary results using hand generated input Indicate the plan created will considerably decrease the search space needed for recognltion, The recognlzer is currently being converted to accept real speech Input,

REFERENCES

Alter, R, [1968], Utilization of contextual constraints in automatic speech recognition, IEEE Transactions on Audio and Electroacoustics AU16, 1 (March), 6-11.

Bobrow, D,, and Klatt, D, [1968], A limited speech recognltlon system, Proceedings of the 1968 AFIPS Fall Joint computer Conference, Thompson Book Company, Washington, D,C,, 305-318,

Duda, R,O,, and Hart, P,E, [1968], Experiments In the recognltion of hand printed text, part II - context analysis, Proceedings of the 1968 AFIPS Fall Joint Computer Conference, Thompson Book Company, Washington, D,C,, 1139-1150,

Floyd, R,W, [1964], Bounded context syntactic analysis, Comm, ACM 7, 2 (February), 62-66,

Goodman, R,G, [1970], Ambiguity in phonetic grammars, Proceedings Of the Third Hawaii International Conference on System Sciences, Granberg, B,S,M, (Ed,), Western Periodicals Company, 560-563,

McKeeman, W,M,, Horning, J,J,, und Wortman, D,B, [1970], A Compiler Generator, Prentice Hall, Englewood Cliffs,

Reddy, D,R, [1967a], Phoneme grouping for speech recognltion, J, Acoust,Soc, Am, 41, 5 (May), 1295-1300,

Reddy, D,R, [1967b], Computer rscognltion of connected speech, J, Acoust, Soc, Am, 42, 2 (August), 329-347,

Reddy, D,R, [1969], On the use of environmental, syntactic, and probabilistic constraints in vision and speech. Artificial Intelligence Memo No, 78, Stanford University, Stanford, California (January),

McCarthy, J,, Earnest, L,D,, Reddy, D,R,, and Vicens, P,J, [1968], A computer with hands, eyes, anti ears, Proceedings of the 1968 AFIPS Fall Joint Computer Conference, Thompson Book Company, WAshington, D,C,, 329-338,

Shah, A, C, [1968], The formal description and parsing of pictures, Ph,D, Thesis, Stanford Univesity, Available as SLAC Report No, 84, Stanford Linear Accelerator Center, Stanford University,

Stanford, California (March).

Unger, S. [1968]. A global parser for context-free phrase structured grammars. Comm. ACM 11, 4 (April), 240-247.

Vicens, P.J. [1969]. Aspects of speech recognition by computer, Ph.D. Thesis, Stanford University. Available as Technical Report No. cs 127, Computer Science Department, Stanford University, Stanford, California (April).

5. VISION AND CONTROL

5.1 Hand-Eye Project [Jerome Feldman, Gilbert Falk, Sundarem Ganapathy, Aharon Gill, Gunnar Grape, Alan Kay, Ugo Montanari, Richard Paul, Karl Pingle, Irwin Sobel, Jay Tenenbaum]

A complete description Of our goals and direction of approach to those goals was outlined in an earlier Project Technical Report [1]. Since that time, substantial progress toward the major goal--the organization of context sensitive visual perception and motor control-- has been realized,

I. Organization

The plan for a system allowing multiple program interaction via both a sub-rronitor and a global associative data structure has been implemented by K, Pingle and R, Sproul [3],[4]. The language SAIL which combines ALGOL, LEAP associative structure, sets, and special string and real time operators has been completed by R, Sproul and D, Swinehart [4]. It is now in genera) use throughout the project, The effect of thls has been to allow separate programs to come together easily In the first trials at interaction for mutual co-operation in scene analysis,

II. Context-sensitive Visual Perception

In 1969, we planned to work on programs to be sensitive to various levels ot visual data and gestalt organization within scenes, This work has proceeded and in some aspects Is near completion. I, Sobel has finished his thesis [5] on camera recalibration after movement of Pan, tilt, or lens change. Using these results, he has created a stereo-depth program capable of resolving the depth of a point to better than 1/10 inch accuracy, J,M, Tenenbaum [6] with K, Pingle has built an accommodative edge detection System, It can apply a variety of strategies to both the TV camera and to methods of processing the TV data so as to optimize the total Information, noise, and cost parameters of edge detection or to provide, upon request, -special information (such as interior edges or very high 'resolution) [7] relevant to the needs of some higher level visual recognition process, It makes use of edge-following routines developed for sensitivity to intensity, color, depth, and, In the future, texture, In addition, it provides facilities for feature and line verificationupon request from other programs,

Making extensive use of these packages of edge detection and verification, G, Grape has continued his work [8] on a line drawing analysis program which deals with the problem of creating a good line drawing In the presence of edge data with missing lines, redundant or false lines, incomplete or chopped-up lines, and lines inconsistent with either the model of the particular scene as it Is being developed by the entire visual perception package or with the known properties of the objects used, (E,g., plane-bounded objects),

Finally, G. Falk is producing the program to Interact with all the above programs to Produce the visual world model of the scene [10]. The program attempts to segment the lines of the line drawing (complete or Incomplete) into groups representing objects, Then it tries to match these partially or wholly seen objects topologically with classes of object prototypes, When some matches are successful, the program attempts to Infer depth lnformatlon from clues In the two-dlmenslonal representation [11]. Thls depth can be used t o resolve the geometric and dimensional relationships of the objects In the scene, His program Is equipped to request lnformatlon from the other vision programs to resolve ambiguity,

III, Arm Control

Work has progressed In two areas; a new arm has been designed and built by Scheinman [12] and Is at present being Interfaced to the computer, The arm has exceptionally hlgh posltlon accuracy (,Ø5 Inch), comparatively fast servo performance (two seconds, point to point), and approximately human arm reach and motlon properties,

A new approach at control ling the arm Is belng undertaken by Richard Paul, combining the work of both Piper [13] (which was largely kinematic) and Kahn [14] (which was largely dynamic), In order to move the arm from one posltlon to another, a trajectory must be defined to avoid colliding with other objects, Starting with this trajectory and the equations of motlon of the arm, posltlon and velocity dependant coefficients relating to the trajectory may be computed, The servo program then utilizes these coefficients to calculate the torques necessary to apply at each joint of the arm such that both the position and velocity errors will be reduced to zero two sample periods later, As this process occurs every sample period, the arm should move along the trajectory,

This approach treats both kinematics and dynamics consistently and provides a natural solution to the servo sampling rate problem, It utilizes the digital computer directly to solve the equation of motion of the arm, not Just to simulate an analog servo system,

In addition, A, Gill is working on visual tracking of the moving arm as a subset of the more general problem of tracking,

IV, Integration of arm and eye

Along the way toward realization of a Program capable Of complex hand-eye behavior, we arepursuing a specific and a general task:

1, Instant-Insanity, We are developing a program to so Ive and construct the Instant-Insanity block stack, This will serve as an initial test of the integration of the visual analysis andmotor control,

2, The Task Language, In attempting to delineate the bounds of the set of construction tasks without limiting either the object types or constructlon operations, we decided to create a language for the description of tasks [15], This language Is defined by production rules and as such can grow as our abilities increase, Right now it

Is capable of representing tasks as complex as "Stack al I the small blue blocks on the right side of the table on top of the red wedge," Asiae from creating a boundary to "task space", another Important effect of the language Is that It provides a structure for task solution and planning, A, Kay is using the task language In his attempt to devise a method of determining strategy for the creation of sub-tasks as well as for discovering the optimum use of the visual-recognition and motor-control Programs to accomplish these subtasks, The strategies will Include considerations of time, space, and cost,

References

1. McCarthy, J. and Staff, "Artificial Intelligence Project Technical Report", June 1969,

3t Feldman, J., et, al, "The Stanford Artificial Intelligence Project", Proc. of the First International Joint Conference on A.I., Washington, D.C., May 1969.

3. Pingle, K. and Sproul, R., Operating Note, forthcoming.

4. Sproul, R. and Swinehart, D., Operating Note, forthcoming.

5._ Sobel, I„ Thesis, forthcoming,

6. Tenenbaum, J.M., Thesis, forthcoming.

7. Heuckel, M., "An Operator Which Locates Edges In Digitalized Pictures", AIM-105, Stanford University, October 1969.

8. Grape, G., "Computer Vision Through Sequential Abstractions", memorandum, June 1969.

9. Grape, G., "On Predicting and Verifying Missing Elements In Line Drawings, memorandum, March 1970.

10. Falk, G., Thesis, forthcoming.

11. Falk, G., "Some Implications of Planarity for Machine Perception", AIM-107, December 1969.

12. Scheinman, V.D., "Design of a Computer Controlled Manipulator," Thesis and AIM-92, June 1969,

13. Pieper, D.L., "The Kinematics of Manipulators Under Computer Control", Thesis and AIM-72, 1968,

14. Kahn, M.E., "The Near-Minimum-Time Control of Open-Loop Articulated Kinetic Chains", Thesis and AIM-106, December 1969,

15, Kay, A., working papers.

5.2 VISUAL CONTROL OF A VEHICLE [Bruce Baumgart, Lynn Quam, Rod Schmidt]

Another attempt to integrate visual-motor activity Is embodied In the cart project. Since the accomplishment last year of a program that allows tne cart to follow a well-defined line, we have installed a new exterior antenna and have collected sequences of actual road pictures taken in the parking lot and around the perimeter drive of the laboratory, From these real world pictures, we have Succeeded ln Isolating the outlines of major road features and obstacles such as the road curbs, cars, grassy hills, the horizon, the sky, and the distant lines of Eucalyptus trees that bound our local world,

In order to meet the goal of a computer driven vehicle, a minimum set of computer capabilities Isas follows:

1) the ability to follow a marked oath, such as a road
2) the ability to recognize the completion of a task, accept direction and begin a new task. (More simply, to decide when to make turns, stop, etc.)

Other abilities are clearly required to practically drive a vehicle, such a s non-destructive error recovery, but a beginning can be made with just the two abl llties llsted, The first capabllity was realized about a year ago with a program which was guided by lines, road edges, etc, Although not extensively tested, the Program was a usable prototype of a vehicle steering system,

The second poI nt involves a step upward in picture processing techniques, since the completion of a task is defined by the vehicle's environment, and Is reflected In the TV Image which the computer has available, Accordingly, a program Is under development which constructs descriptions of scenes, and recognizes a given Scene from Its stored description when it Is seen again, Instructions for action can be associated with the scenes, and thus enable the vehicle to carry out Such missions as "Drive around the bui lding and come back here,"

At present, a prototypical program can recognize as identical two views of the same picture, translated and rotated arbltrarl ly, If the picture contains two or more small- objects, even if not al I the objects are in both views, The program Is being extended to handle features so large that they do not flt in the picture,, and must be described In terms of sub- features, The picture Is currently described in the Image plane of the camera, but wlll be later described in road coordinates, The description Is, and wlll continue to be, essentially two-dimensional, because of the large time penalty In 3-D processing, Time, of course, Is of the essence, since the vehicle Is moving during ail the processing, and hence recognition time cannot exceed afew seconds.

5.3 VISUAL IDENTIFICATION OF PEOPLE [Michael Kelly3

The problem which has been chosen is the following: develop a program which will recognize people standing in front of a TV camera attached to a computer. During an individual's firstappearance, the program is set to request the name of the person whose picture is being processed. The name is associated with features and measurements from the picture which characterize the person. At a subsequent appearance before the TVcamera, the computer will type out the name of the person.

OUTLINE OF THE METHOD.

The general scheme of operation of tne program can be summarized as follows. Two pictures of tne individual to be recognized are read into the computer. One is a picture of the entire body, head to feet. The other is a close-up of the head. The program processes the pictures to locate feature points such as the irises of the eyes, nostrils, the top of the head. Once these points are found, measurements are derived from them such as height, distance between eyes, width of head, etc. These measurements are then used in a pattern classification algorithm to extract the identity of the nerson from a dictionary containing known individuals and their measurements.

The method outlined above appears quite straight- forward: locate features, obtain measurements, classify pattern. Obviously, obtaining measurements once feature points are located is a trivial operation. The final stage of the method, pattern classification, has oeen well studied. Given a good set of features, there are standard classification algorithms which may be used. However, the first step, locating the features in a digitalpicture, is a process about which very little is known.

Thls then is the main effort of this thesis research: accurately locating desired specific points on pictures of people. Such low level picture processing, called variously feature extraction, pre-processing, or characterization of the picture, is generally recognized as the most difficult part and the principal Problem In pattern recognition. (See,' for example, Ho and Agrawala [1968], p. 2102.) It Is worthwhile emphasizing, in vlew of the fact that so much Previous work in pattern recognition has been concerned with mathematical techniques for classification, that classification has received only secondary attention in this work.

Measurerrents from the face and body should provide a good means of identifying people. The reason for this expectation is that physical measurements were the basls of the Bertillon system for identification ofpeople, which had wide use in police work prior to the discovery of the usefulness of fingerprints (Thorwald [1965]). This expectation is confirmed by the results obtalned by Bledsoe which are described below.

PREVIOUS WORK.

A number of papers have appeared In which pictures of faces have been partially processed by computer and the results a/splayed, Examples are the papers by Narasimhan and Forango [1964] and Huecke| [1969]. Both of these papers present computer produced line drawings of faces which were derlved from grey scale Input pictures, Such operations are fine for presenting faces to the human eye but represent only a very small step toward computer description and recognition,

The principal prior work on recognltlon of people by computer has been done by W.W. Bledsoe. This work was begun at Panoramic Yesearch, Inc, and continued with P.E, Hart at Stanford Research Instltute, (See Bledsoe [1964] and [1966]).

There are differences and similarities between the work of Bledsoe and the work reported here, The chief difference Is that Bledsoe created a man-machine system In which a human operator, working with a face projected on a Grafacon or Wand tablet", located the feature points on the face and manually pointed out thelr position for th.e computer to record, In contrast, my work consists primarily of an attempt to automate this feature location step, Bledsoe was concerned with recognition of photographs of faces; I consider both body and face and work with llve subjects, not photographs, Bledsoe permitted a wide variation In head rotation, tllt, lean, photograph quality, and llght contrast; I require much more standardization of pose and can obtaln It since I contro| the picture taking environment, In splte of these differences, the work reported here follows the basic Idea for visual ldentlflcatlon of people first laid out by Bledsoe: find the measurements and use them for identiflcation, Another way of summarizing this is: automate the ldentlflcatlon techniques of Bertll Ion,

Bledsoe's results verified that facial measurements made on photographs could be used effectively for facial recognltlon, In his work, measurements were obtained from 2000 photographs, 2 photographs for each person In the sample, Given a set of measurerrents for an unknown person, the classlflcatlon system attempted to supply a name or a small list of names which Included the unknown individual, Using varlous classification methods Bledsoe found that the average reduction In uncertainty varled from 1/100 to 1/4000,

Bledsoe's group was also concerned to allmited extent with finding features automatically (Bisson [1965a], [1965b]), The results of this were inconclusive: many problems were encountered tryfng to determine the location of feature points, Bledsoe's success with facial classlflcatlon using measurements while leaving open the problem of automatic feature location has been a stimulus for the work reported In this thesis,

Sakai, Nagao, and Fujibayashi C19691 have reported the lr work on finding faces In photographs, Their goal is to detect If a face or faces are present In a picture, They first produce a picture which contains the edges of the Input picture, A large oval template corresponding to the head outline Is then matched with the edge

picture, All reasonable positions and sizes of the template are tried, In those positions where the oval template receives a hlgh response, the head hypothesis is checked by further template matching that expects many edges In the eyes, nose, and mouth and few edges on the forehead, This method appears to be time consuming, and the result Is only an approximate location for the nead In the picture,

Three Russians, El'bur [1967], Yurans [1967], and Rastr lgan [1967], nave presented methods for identifying faces from photographs, The methods assume that a representation of a face as a set of points is available, The papers are mostly on projective geometry; there Is very little mention of application, Hart [1969] has prepared a summary of the content of these papers,

References

Bisson, C.L. [1965a] Preliminary investigation on measurements by computer of the distances on and about the eyes, Report PRI :18, Panoramic Research Inc,, Palo Alto, California (April),

Bisson, C.L. [1965b) Location of some facial features by computer, Report PRI:20, Panoramic Research Inc,, Palo Alto, Callfornia (June),

Bledsoe, W.W. [1964] The mode I method in facial recognition, Report PRI:15, Panoramic Research Inc,, Palo Alto, Callfornia (August),

Bledsoe, W.W. [1966] Man-machine facial recognition, Report PRI:22, Panoramic Research Inc., Palo Alto, California (August),

El'bur, R.E. [1967] Utilization of the apparatus of projective geometry In the process of the identification of individuals by their photographs, In Problems of Cybernetics and Law, Kudryavtsev, V.N. (Ed,), Nauka, Moscow, pp. 321-348, Available through U.S. Department of Commerce, Joint Publications Research Service, No, JPRS: 43,954 (January 10, 1968),

Hart, P.E. C19693 Personal communlcatlon,

Hueckel, M, [1969] An operator wh Ich locates edges ln digitized pictures, Artificial Intelligence Memo AIM-105, Stanford Univ,, Stanford, Calif, (October),

Ho, Y. and Agrawala, A.K. [1968] On pattern classification algorithms, Introduction and survey, Proc, IEEE 56 (December), pp. 2101-2114,

Narasimhan, R. and Forango, J.P. C19643 Some further experiments In the parallel processing of pictures, IEEE Transactions on Electronic Computers 13 (December), pp. 748-750,

Rastrigin, L.A. [1967] About the Identification of images of spatial objects. In Problems of Cybernetics and Law, Kudryavtsev, V.N. (Ed,), Nauka, Moscow, pp. 361-368. Available through U.S. Department of Commerce8 Joint Publications Research Service, No. JPRS: 43,954 (January 10, 1968).

Sakai, T., Nagao, M., and Fujibayashi, S, [1969] Line extraction and Pattern detection In a photograph, Pattern Recognition 1 . (March), pp. 233-248.

Thorwald, J, C19653 The Century of the Detective, Harcourt, Brace, 8 World, New York,

Yurans, V , [1967] Certain questions concerning the theory of identification of objects with the use of the apparatus of projective geometry. In Problems of Cybernetics and Law, Kudryavtsev, V.N. (Ed.), Nauka, Moscow, pp. 349-360. Available through U.S. Department of Commerce, Joint Publications Research Service, No. JPRS: 43,954 (January 10, 1968).

APPENDIX A

PUBLICATIONS

Articles and books by members of the Stanford Artificial Intelligence Project are listed here by year. Only publications following the individual's affiliation with the Project are given.

1963

1. J. McCarthy, "A Basis for a Mathematical Theory of Computation," In P. Blaffort and 3. Hershberg (eds.), Computer Programming and Formal Systems, North-Holland, Amsterdam, 1963,

2. J. McCarthy, "Towards a Mathematical Theory of Computation," In Proc. IFIP Congress 62, North-Holland, Amsterdam, 1963.

3. J. McCarthy (with S. Boilen, E. Fredkin, and J.C.R. Licklider), "A Time-Sharing Debugging System for a Small Computer," In Proc. AFIPS Conf. (SJCC), Vol. 23, 1963,

4. J. McCarthy (with F. Corbato and M. Daggett), "The Linking Segment Subprogram Language and Linking Loader Programming Languages," Comm. ACM, July, 1963,

1965

1. J. McCarthy, "Problems In the Theory of Computation," In Proc. IFIP Congress 65, Spartan, Washington, D.C., 1965.

1966

1. A. Hearn, "Computat I on of Algebraic Properties of Elementary Particle Reactions Using a Digital Computer," Comm, ACM, 9, pp. 573-577, August, 1966.

2. J. McCarthy, "A Formal Description of a Subset of Algol," In T. Steele (ed.), Formal Language Description Languages, North-Ho/ (and, Amsterdam, 1966,

3. J. McCarthy, "Information," Scientific American, September, 196.6,

4. J. McCarthy, "Time-Sharing Computer Systems," in W. Orr (ed.), Conversational Computers, Wiley, 1966,

5. D. Reddy, "Segmentation of Speech Sounds," J. Acoust, Soc. Amer., August 1966.

1967

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

2. J. Campbell' "Algebraic Computation of Radlatlve Correcttons for Electron-Proton Scatterlng," Nuclear Physics, Vol, B1, pp. 238-300, 1967.

3. E. Feigenbaum, "Information Processing and Memory," In Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 4, U.C. Press, Berkeley, 1967,

4. J. 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.

5. J. Goodman, "Digital Image Formatlon from Electronically Detected Holograms," Applied Physics Letters, August 1967.

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

7. J, Lederberg, "Hamilton Circuits of Convex Trivalent Polyhedra," American Mathematical Monthly 74, 522, 1967,

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

9. J. McCarthy, "Computer Control of a Hand and Eye)" In Proc. Thlrd Ail-union Conference on Automatic Control (Technical Cybernetics), Nauka, Moscow, 1967 (Russian),

10. J, McCarthy and J, Painter, "Correctness of a Compiler for Arithmetic Expressions," Amer, Math, Soc., Proc, Symposia in Applied Math., Math, Aspects of Computer Science, New York, **1967,**

11. D, Reddy, "Phoneme Grouping for Speech Recognition," J, Acoust, Soc, Amer.' May, 1967.

12, D, Reddy, "Pitch Period Determination of Speech Sounds," Comm, ACM, June' 1967.

13, D, Reddy, Computer Recognition of Connected Speech," J, Acoust, Soc, Amer., August, 1967.

14, A, Samuel, "Studies in Machine Learning Using the Game of Checkers' II-Recent Progress," IBM Journal, November, 1967.

15. G. Sutherland (with G,W. Evans and G,F. Wallace), Simulation Using Digital Computers, Prentice-Hall, Engelwood Cliffs, N.J., 1967.

1968

1. E. Feigenbaum, J. Lederberg and B. Buchanan, "Heuristic Dendral", Proc, International Conference on System 'Sciences, University of Hawaii and IEEE, University of Hawaii Press, 1968'

2. E, Feigenbaum, "Artificial Intelligence: Themes in the Second Decade", Proc. IFIP Congress' 1968,

3. J. Feldman (with D. Gries), "Translator Writing Systems", Comm, ACM, February 1968,

4. J. Feldman (with P. Rovner), "The Leap Language Data Structure", Proc. IFIP Congress, 1968,

5. R. Gruen and W. Weiher, "Rapid Program Generation", Proc, DECUS Symposium, Fall 1968,

6. A, Hearn, "The Problem of Substitution", Proc. IBM Summer Institute on Symbolic Mathematics by Computer, July 1968,

7, D. Kaplan, "Some Completeness Results in the Mathematical Theory of Computation", ACM Journal, January 1968,

8. J. Lederberg and E, Feigenbaum, "Mechanization of Inductive Inference in Organic Chemistry", in B. Kleinmuntz (ed.), Formal Representation of Human Judgment, John Wiley, New York, 1968,

9. J. McCarthy, "Programs with Common Sense" in M, Minsky (ed,), Semantic Information Processing, MIT Press, Cambridge, 1968,

10. J, McCarthy, L, Earnest, D, Reddy, and P, Vicens, "A Computer with Hands, Eyes, and Ears", Proc. AFIPS Conf, (FJCC), 1968,

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

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

13, D, Reddy, "Computer Transcript/on of Phonemic Symbols", J, Acoust. Soc. Amer., August 1968,

14. D, Reddy, and P, Vicens, "Procedure for Segmentation of Connected Speech", J, Audio Eng, Soc., October 1968,

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

16. A. Silvestri and J. Goodman, "Digital Reconstruction of Holographic Images", 1968, NEREM Record, IEEE, Vol, 10, pp. 118-119, 1968,

17. L. Pester, H. Enea, and K. Colby, "A Directed Graph Representation for Computer Simulation of Belief Systems", Math, Bio, 2, 1968,

1969

1. J. Beauchamp (with H. Von Foerster) (eds.), "Music by Computers% John Wiley, New York, 1969,

2. J. Becker, "The Modeling of Simple Analogic and Inductive Processes In a Semantic Memory System", Proc. International Conf, on Artificial Intelligence, Washington, D.C., 1969,

3. B. Buchanan and G, Sutherland, "Heuristic Dendral: A Program for Generating Hypotheses In Organic Chemistry", In D, Michie (ed.), Machine Intelligence 4, American Elsevier, New York, 1969,

4. B. Buchanan (with C, Churchman), "On the Design Of Inductive Systems: Some Philosophical Problems5 British Journal for the Philosophy of Science, 20, 1969, pp, 311-323,

5. K. Colby, L, Tesler, and H, Enea, "Experiments with a Search Algorithm for the DataBase of a Human Belief System", Proc. Internatlonal Conference on Artificial Intelligence, Washington, D.C., 1969,

6. K, Colby and D. Smith, "Dialogues between Humans and Artlflclal Belief Systems!', Proc, Internatlonal Conference on Artificial Intelligence, Washington, D.C., 1969,

7. A. Duffield, A, Robertson, C. Djerassi, B. Buchanan, G. Sutherland, E. Feigenbaum, and J. Lederberg, "Application of Artlflclal Intelligence for Chemical Interference II. Interpretation of Low Resolution Mass Spectra of Ketones", J. Amer, Chem. Soc., 91:11, May 1969,

8. J, Feldman, G, Feldman,G, Falk,G, Grape, J, Pearlman, I, Sobel, and J. Tenenbaum, "The Stanford Hand-Eye Project", Proc, Internatlonal Conf, on Artlflclal Intelligence, Washington, D.C., 1969,

9. J. Feldman (with P, Rovner), "An Algal-based Associative Language", Comm, ACM, August 1969,

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

11. J. Lederberg, "Topology of Organic Molecules", National Academy of Science, The Mathematical Sciences: a Collection of Essays, MIT Press' Cambridge, 1969'

12, J, Lederberg, G. Sutherland, B. Buchanan, E, Feigenbaum, A, Robertson, A, Duffield, and C. Djerassi, "Applications of Artificial Intelligence for Chemical Inference I, The Number of Possible Organic Compounds: Acyclic Structures Containlnq C, H, O, and N", J, Amer. Chem. Soc., 91:11, May 1969,

13. Z. Manna, "Properties of Programs and the First Order Predicate Calculus", J. ACM, April 1969,

14. Z. Manna, "Formalization of Properties of Programs", J, System and Computer Sciences, May 1969.

15. Z. Manna, "Formalization of Properties of Recursively Defined Functions"' Proc. ACM Symposium on Computing Theory' May 1969.

16. J, McCarthy and P. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence", in' D. Michie (ed.), Machine Intelligence 4, American Elsevier, New York, 1969.

17. U, Montanari, "Continuous Skeletons from Digitized Images", JACM, October 1969.

18. N, Nilsson, "A Mobile Automaton: An Application of Artlficial Intelligence Techniques", Proc, International Conf, on Artificial Intelligence, Washington, D.C., 1969.

19. R, Paul' G. Falk, J, Feldman, "The Computer Hepresentatlon of Simply Described Scenes", Proc, Illinois Graphics Conference, April 1969,

20. R, Schank and L. Tesler, "A Conceptual Parser for Natural Language", Proc, International Joint Conference on Artificial Intelligence, Washington, D.C., 1969.

21. G, Schroii, A' Duffield, C, Djerassi, B, Buchanan, G, Sutherland, E, Feigenbaum, and J. Lederberg, "Applications of Artificial Intelligence for Chemical inference III, Aiiphatic Ethers Diagnosed by Their Low Resolution Mass Spectra and NMR Data", J, Amer. Chem. Soc., 91:26, December 1969.

1970 (to date)

1. J. Allen and D. Luckham, "An Interact/w Theorem-Proving Program" In B. Meltzer and D. Michie (eds.), Machine Intelligence 5, Edinburgh University Press, 1970,

2. B. Buchanan, G. Sutherland, and E, Feigenbaum, "Toward an Understanding of Information Processes of Scientific Inference, In the Context of Organic Chemistry" In 8, Meltzer and D. Michie (eds), Machine Intelligence 5, Edinburgh University Press, 1970,

3. D. Luckham, "Refinement Theorems In Resolution Theory", Proc. 1968 IRIA Symposium In Automatic Deduction, Versailles, France, Springer-Verlag, 1970.

4. Z. Manna and J. McCarthy, "Properties Of Programs and Partial Function Logic" In B. Meltzer and D. Michie (eds.), Machine Intelligence 5, Edinburgh University Press, 1970,

5. Z. Manna, "The correctness of Non-Deterministic Programs", Artificial Intelligence Journal, Vol. 1, No. 1, 1970 (In press).

6. Z. Manna, "Second-order Mathematical Theory of Computation", Proc. ACM Symposium on Theory Of Computing, May 1970.

7. Z. Manna and A. Pnueli, "Formalization of Properties of Recursively Defined Functions", J, ACM, July 1970 (In press),

8. U. Montanari, "A Note on Minimal Length Polygonal Approximation to a Digitized Contour", Comm. ACM, January 1970,

9. U. Montanari, "On Limit Properties in Digitization Schemes", JACM, April 1970,

APPENDIX B

THESES

Theses that have been published by the Stanford Artificial Intelligence Project are listed here, Several earned degrees at institutions other than Stanford, as noted, Abstracts of recent A, I, Memos are given in Appendix D,

AIM-43, R, Reddy, AN APPROACH TO COMPUTER SPEECH RECOGNITION BY DIRECT ANALYSIS OF THE SPEECH WAVE, Ph,D, Thesis In computer Science, September 1966,

AIM-46, S, Persson, 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,

AIM-47, B, Buchanan, LOGICS OF SCIENTIFIC DISCOVERY, Ph,D, Thesis In Philosophy, University of California, Berkeley, December 1966,

AIM-44, J, Painter, SEMANTIC CORRECTNESS OF A COMPILER FOR AN ALGOL-LIKE LANGUAGE, Ph,D, Thesis In Computer Science, March 1967,

AIM-56, W, Wichman, USE OF OPTICAL FEEDBACK IN THE COMPUTER CONTROL OF AN ARM, Eng, Thesis In Electrical Engineering, August 1967,

AIM-58, M, Callero, AN ADAPTIVE COMMAND AND CONTROL SYSTEM UTILIZING HEURISTIC LEARNING PROCESSES, Ph.D. Thesis In Operations Research, December 1367,

AIM-63, D, Kaplan, REGULAR EXPRESSIONS AND THE EQUIVALENCE OF PROGRAMS, Ph,D, Thesis in Computer Science, July 1968,

AIM-65, B, Huberman, A PROGRAM TO PLAY CHESS END GAMES, Ph,D, Thesis in Computer Science, August 1968,

AIM-73, D, Pieper, THE KINEMATICS OF MANIPULATORS UNDER COMPUTER CONTROL, Ph,D, Thesis In Mechanical Engineering, October 1968,

AIM-74, D, Waterman, MACHINE LEARNING OF HEURISTICS, Ph,D, Thesis In Computer Science, December 1968,

AIM-83, R, Schank, A CONCEPTUAL DEPENDENCY REPRESENTATION FOR A COMPUTER ORIENTED SEMANTICS, Ph,D, Thesis in Linguistics, University of Texas, March 1969,

AIM-85, P, Vicens, ASPECTS OF SPEECH RECOGNITION BY COMPUTER, Ph,D, Thesis In Computer Science, March 1969,

AIM-92, Victor D, Scheinman, DESIGN OF COMPUTER CONTRGLLED MANIPULATOR, Eng, Thesis In Mechanical Engineering, June 1969,

AIM-96, Claude Cordell Green, THE APPLICATION OF THEOREM PROVING TO QUESTION-ANSWERING SYSTEMS, Ph.D. Thesis In Electrical Engineering, August 1969,

AIM-98, James J. Horning, A STUOY OF GRAMMATICAL INFERENCE, Ph.D. Thesis In Computer Science, August 1969,

AIM-106, Michael E. Kahn, THE NEAR-MINIMUM-TIME CONTROL Of OPEN-LOOP ARTICULATED KINEMATIC CHAINS, Ph.D. Thesis In Mechanical Engineering, December 1969,

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:

Artificial Intelligence Project Secretary
Computer Science Department
Stanford University
Stanford, California 94305

Alternatively, prints may he purchased at cost (typically $30 to $50) from:

Cine-Chrome Laboratories
4075 Transport St,
Palo Alto, California

1. Art Eisenson and Gary Feldman, "Ellis D. Krontechev and Zeus, his Marvelous Time-Sharing System", 16mm black and white 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 control ling 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 t-he 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 control led Oy 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 Pelper, "Avoid", 16mm silent, color, 5 minutes, March 1969,

Reports an a computer program written by D, Pelper for his Ph,D, Thesis, The problem is to move the computer controlled electrical-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 vqr ious cluttered environments with fairly good success,

Appendix D

ARTIFICIAL INTELLIGENCE MEMOS

These memos report both research results and facility descriptions (computer programs and equipment) in the period 1963-66, From late 1966 on, just research is reported, Facilities are described In internal reports called Operating Notes (Appendix E).

Copies of many of these memos are available to Interested researchers upon request to:

Artificial Intelligence Project Secretary
Computer Science Department
Stanford University
Stanford, California 94385

Alternatively, beginning with memo AIM-69, they are available from

Clearinghouse for Federal Scientific
and Technical Information
Springfield, Virginia 22151

The Clearinghouse charges $3,00 per full size copy and $,65 for a microfiche copy,

Titles only are listed below for 1963-68, Abstracts are given for the more recent memos,

1963

AIM-1, John McCarthy, PREDICATE CALCULUS WITH "UNDEFINED" AS A TRUTH-VALUE, March 1963, 5 paws,

AIM-2, John McCarthy, SITUATIONS, ACTIONS, AND CAUSAL LAWS, July 1963, 11 Pages:,

AIM-3, F, Safier, "THE MIKADO" AS AN ADVICE TAKER PROBLEM, July 1963, 4 pages,

AIM-4, H, Enea, CLOCK FUNCTION FOR LISP 1.5, August 1963, (Out of print),

AIM-5, H, Enea and D, Wooldridge, ALGEBRAIC SIMPLICATION, August 1963, 1 page,

AIM-6, D, Wooldridge, NON-PRINTING COMPILER, August 1963, 2 pages, (out of print),

AIM-7, John McCarthy, PROGRAMS WITH COMMON SENSE, September 1963, 7 pages,

AIM-8, John McCarthy, STORAGE CONVENTIONS IN LISP 2, September 1963, (Out of print).

AIM-9, C.M. Williams, COMPUTING ESTIMATES FOR THE NUMBER OF BISECTIONS OF AN N xN CHECKERBOARD FOR N EVEN, December 1963, 9 pages (Out of print).

AIM-10, S.R. Russell, IMPROVEMENTS IN LISP DEBUGGING, December 1963, 3 pages,

AIM-11, D. Wooldridge, AN ALGEBRAIC SIMPLIFY PROGRAM IN LISP, December 1963, 57 pages.

1964

AIM-12, G. Feldman, DOCUMENTATION OF THE MAC MAHON SQUARES PROBLEM, January 1964, (Out of print).

AIM-13, D. Wooldridge, THE NEW LISP SYSTEM (LISP 1.55), February 1964, 4 pages.

AIM-14, John McCarthy, COMPUTER CONTROL Of A MACHINE FOR EXPLORING MARS, January 1964, 6 pages.

AIM-15, M. Finkelstein and F. Safier, AXIOMATIZATION AND IMPLEMENTATION, June 1964, 6 pages.

AIM-16, John McCarthy, A TOUGH NUT FOR PROOF PROCEDURES, July 1964, 3 pages.

AIM-17, John McCarthy, FORMAL DESCRIPTION Of THE GAME Of PANG-KE, July 1964, 2 pages.

AIM-18, J. Hext, AN EXPRESSION INPUT ROUTINE FOR LISP, July 1964, 5 pages.

AIM-19, J. Hext, PROGRAMMING LANGUAGES AND TRANSLATION, August 1964, 14 pages.

AIM-20, Raj Reddy, SOURCE LANGUAGE OPTIMIZATION OF FOR-LOOPS, August 1964, 37 pages.

AIM-21, R.W. Mitchell, LISP 2 SPPCIFICATIONS PROPOSAL, August 1964, 12 pages.

AIM-22, R. Russell, KALAH - THE GAME AND THE PROGRAM, September 1964, 13 pages.

AIM-23, R, Russet|, IMPROVEMENTS T O THE KALAH PROGRAM, September 1964, 12 pages.

AIM-24, John McCarthy, A FORMAL DESCRIPTION Of A SUBSET OF ALGOL, September 1964, 4 3 pages.

AIM-25, R. Mansfield, A FORMAL SYSTEM OF COMPUTATION, September 1964, 7 pages.

AIM-26, Raj Reddy, EXPERIMENTS ON AUTOMATIC SPEECH RECOGNITION BY A DIGITAL COMPUTER, October 1964, 19 pages,

1965

AIM-27, -John McCarthy, A PROOF-CHECKER FOR PREDICATE CALCULUS, March 1965, 7 pages.

AIM-28, John McCarthy, PROBLEMS IN THE THEORY Of COMPUTATION, March 1965, 7 pages.

AIM-29, C.M. Williams, ISOLATION O f IMPORTANT FEATURES OF A MULTITONED PICTURE, January 1965, 9 pages,

AIM-30, E. Feigenbaum and R.W. Watson, AN INITIAL PROBLEM STATEMENT FOR A MACHINE INDUCTION RESEARCH PROJECT, April 1965, 8 pages,

AIM-31, John McCarthy, PLANS FOR THE STANFORD ARTIFICIAL INTELLIGENCE PROJECT, April 1965, 5 paws,

AIM-32, H. Ratchford, THE 138 ANALOG DIGI TAL CONVERTER, May 1965, 9 pages.

AIM-33, B, Huberman, THE ADVICE TAKER AND GPS, June 1965, 10 pages,
AIM-34, P. Carah, A TELEVISION CAMERA INTERFACE FOR THE PDP-1, June 1965, 8 pages.

AIM-35, F, Safier, SIMPLE SIMON, June 1965, 17 pages (Out ofprint).

AIM-36, J, Painter, UTILIZATION Of A TV CAMERA ON THE PDP-1, September 1965, 6 pages,

AI+37, K, Korsvold, AN ON LINE ALGEBRAIC SIMPLIFICATION PROGRAM, November 1965, 36 pages.

1966

AIM-38, D. Waterman, A FILTER FOR A MACHINE INDUCTION SYSTEM, January 1966, 1 9 pages,

AIM-39, Karl Pingle, A PROGRAM TOFIND OBJECTS IN A PICTURE, January 1966, 2 2 pages,

AIM-40, John McCarthy and J, Painter, CORRECTNESS OF A COMPILER FOR ARITHMETIC EXPRESSIONS, April 1966, 13 pages,

AIM-41, P, Abrams and D, Rode, A PROPOSAL FOR A PROOF-CHECKER FOR CERTAIN AXIOMATIC SYSTEMS, May 1966, 10 pages, (Out of print),

AIM-42, Karl Pingle, A PROPOSAL FOR A VISUAL INPUT ROUTIME, June 1966, 11 pages,

AIM-43, Raj Reddy, AN APPROACH TO COMPUTER SPEECH RECOGNITION BY DIRECT ANALYSIS OF THE SPEECH WAVE, September 1966, 144 pages,

AIM-44, J, Palnter, SEMANTIC CORRECTNESS OF A COMPILER FOR AN ALGOL-LIKE LANGUAGE, Revised March 1967, 130 pages,

AIM-45, D, Kaplan, SOME COMPLETENESS RESULTS IN THE MATHEMATICAL THEORY OF COMPUTATION, October 1966, 22 pages,

AIM-46, S, Persson, SOME SEQUENCE EXTRAPOLATING PROGRAMS: A STUDY OF REPRESENTATION AND MODELING IN INQUIRING SYSTEMS, September 1966, 176 pages, (Out of print),

AIM-47, Bruce Buchanan, LOGICS OF SCIENTIFIC DISCOVERY, December 1966, 210 pages, (Out of print),

1967

AIM-48, D, Kaplan, CORRECTNESS OF A COMPILER FOH ALGOL-LIKE PROGRAMS, July 1967, 46 pages,

AIM-49, Georgia Sutherland, DENDRAL - A COMPUTER PROGRAM FOR GENERATING AND FILTERING CHEMICAL STRUCTURE% February 1967, 34 pages,

AIM-50, Anthony C, Hearn, REDUCE USERS' MANUAL, February 1967, 53 pages,

AIM-51, Lester D, Earnest, CHOOSING AN E.YE FOR A COMPUTER, April 1967, 154 pages,

AIM-52, Arthur L.Samuel, SOME STUDIES IN MACHINE LEARNING USING THE GAME OF CHECKERS II - RECENT PROGRESS, June 1967, 48 pages,

AIM-53, B, Welher, THE PDP-6 PROOF CHECKER, June 1967, 47 pages,

AIM-54, Joshua Lederberg and E,A, Feigenbaum, MECHANIZATION OF INDUCTIVE INFERENCE IN ORGANIC CHEMISTRY, August 1967, 29 pages,

AIM-55, J, Feldman, FIRST THOUGHTS OF GRAMMATICAL INFERENCE, August 1967, 18 pages,

AIM-56, W, Wichman, USE OF OPTICAL FEEDBACK IN THE COMPUTER CONTROL OF AN ARM, August 1967, 69 pages,

AIM-57, Anthony C. Hearn, REDUCE, A USER-ORIENTED INTER- ACTIVE SYSTEM FOR ALGEBRAIC SIMPLIFICATION, October 1967, 69 pages.

AIM-58, M.D. Callero. AN ADAPTIVE COMMAND AND CONTROL SYSTEM UTILIZING HEURISTIC LEARNING PROCESSES, December 1967, 161 pages.

1968

AIM-59, D.M. Kaplan, A FORMAL THEORY CONCERNING THE EQUIVALENCE OF ALGORITHMS, May 1968, 20 pages,

AIM-60, D.M. Kaplan, THE FORMAL THEORETIC ANALYSIS OF STRONG EQUIVALENCE FOR ELEMENTAL PROGRAMS, June 1968, 263 pages, (out of print).

AIM-61, T, Ito, NOTES OF THEORY -OF COMPUTATION AND PATTERN RECOGNITION, May 1968, (Out of print),

AIM-62, B. Buchanan and G. Sutherland, HEURISTIC DENDRAL: A PROGRAM FOR GENERATING EXPLANATORY HYPOTHESES IN ORGANIC CHEMISTRY, July 1968, 76 pages.

AIM-63, D.M. Kaplan, REGULAR EXPRESSIONS AND THE EWIVALENCE OF PROGRAMS, July 1968, 42 pages,

AIM-64, Z. Manna, FORMALIZATION OF PROPERTIES OF PROGRAMS, July 1968, 18 pages,

AIM-65, B. Huberman, A PROGRAM TO PLAY CHESS END GAMES, August 1968, 168 pages,

AIM-66, J. Feldman and P. Rovner, AN ALGOL-BASED ASSOCIATIVE LANGUAGE, August 1968, 31 pages,

AIM-67, E, Feigenbaum, ARTIFICIAL INTELLIGENCE: THEMES IN THE SECOND DECADE, August 1968, 39 pages, (out of print).

AIM-68, Z. Manna and A, Pnueli, THE 'VALIUITY PROBLEM OF THE 91-FUNCTION, August 1968, 20 pages,

AIM-69, John McCarthy, PROJECT TECHNICAL REPORT, September 1968, 90 pages.

AIM-70, Anthony C. Hearn, THE PROBLEM OF SUBSTITUTION, December 1968, 14 pages.

AIM-71, P, Vicens, PREPROCESSING FOR SPEECH ANALYSIS, October 1968, 33 pages,

UN-72, D.L. Pieper, THE KINEMATICS OF MANIPULATORS UNDER COMPUTER CONTROL, October 1968, 157 pages.

AIM-73, John McCarthy, SOME PHILOSOPHICAL PROBLEMS FROM THE STANDPOINT OF ARTIFICIAL INTELLIGENCE, November 1968, 51 pages.

AIM-74, D. Waterman, MACHINE LEARNING OF HEURISTICS, 1968, (Out of Print).

AIM-75, R.C. Schank, A NOTION OF LINGUISTIC CONCEPT: A PRELUDE TO MECHANICAL TRANSLATION, December 1968, 21 pages.

AIM-76, R.C. Schank, A CONCEPTUAL PARSER FOR NATURAL LANGUAGE, December 1968, 22 pages, (out of print).

AIM-77, J.D. Becker, THE MODELING OF SIMPLE ANALOGIC AND INDUCTIVE PROCESSES IN A SEMANTIC MEMORY SYSTEM, January 1969, 21 pages,

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.R. Reddy, ON THE USE OF ENVIRONMENTAL, SYNTACTIC AND PROBALISTIC CONSTRAINTSINVISION AND SPEECH, January 1969, 23 pages,

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 heurlstlcs, 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, D.R. Reddy and R.B. Neely, CONTEXTUAL ANALYSIS OF PHONEMES OF ENGLISH, January 1969, 71 pages,

It is now well known that the acoustic characteristics of a Phoneme depend on both the orecedlng and following phonemes, Thls paper provides some needed contextual and probabilistic data about trigram phonemic sequences of spoken English, Since there are approximately 40+3 such sequences, one must dlscover and study only the more

commonly occurrlng sequences, To this purpose, three types of tables are presented, vlz., a, Commonly occurrlng 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 occurrlng word boundary sequences of the form /-ab/ and /ab-/ where /-/ represents the silence phoneme, Entrles of the above tables contain exam I es of usage and probabilities of occurrence for each such sequence.

AIM-80, Georgla Sutherland, HEURISTIC DENDRAL: A FAMILY OF LISP PROGRAMS, March 1969, 46 pages,

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, Dav I d Luckham, REFINEMENT THEOREMS IN RESOLUTION THEORY, March 1969, 31 pages, (out of print),

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 ln conjunction, The results o f some preliminary experiments wlth the reflnements are given, Presented at the IRIA symposlum on Automatic Deduction, Versalles, France, December 16-21, 1968,

AIM-82, Zohar Manna and Amir Pneuli, FORMALIZATION OF PROPERTIES OF RECURSIVELY DEFINED FUNCTIONS, March 1969, 26 pages, (out of print),

This paper I s concerned wlth the relationship between the convergence, correctness and equivalence of recursively defined functions and the satisfiability (or unsatisfiability) of certain first-order formulas,

AIM-83, Roger C, Schank, A 'CONCEPTUAL REPRESENTATION FOR CUMPUTER-ORIENTED SEMANTICS, March 1969, 281 pages, (out of print),

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 underlylng conceptual structure, It Is desirable to have the machine structure Its own experience, botn linguistic and nonlinguistic, In a manner concomitant wlth the human method for dolng 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 under lying conceptual structure, The conceptual dependencies are utilized as the hlghest 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 fi ies tnat serve to spel I 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 Tart, 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 fi I i ln the semantic file, It Is possible to reverse the realization rules to operate on sentences and produce a conceptual parse, Ail potential parses are checked with the conceptual Semantics in order to eliminate semantic and syntactic ambiguities, The system has been programmed; coherent sentences have beer: generated and the parser is operable, The entire system Is posited as a viable linguistic theory,-

AIM-84, David Canfield Smith, MLISP USERS' MANUAL, January 1969,5 7 pages, (out of print, revision In preparation),

MLISP Is a LISP pre-processor designed to facilitate the wrltlng, use, and understanding of LISP progams, This is accomplished through Parentheses reduction, comments, introduction of a more vlsual 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, Pierre Vicens, ASPECTS OF SPEECH RECOGNITION BY COMPUTER, April 1969, 210 pages,

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, seqnent classification, recognition o f words, recognition of sentences, We present implemented solutions to each of these problems which achieved accurate recognition In all the trial cases,

AIM-86, P,J, Hayes, A MACHINE-ORIENTED FORMULATION OF THE EXTENDED FUNCTIONAL CALCULUS, June 1969, (out of print),

The Lxtended Functional Calculus (EFC) a three-valued predicate calculus Intended as a language In which to reason about the results of computat ions, is described In some detail, A forma) 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 cf Semantic Trees, Finally some remarks are made on efficiency,

AIM-87, John McCarthy and the A,I, Project Staff, PROJECT TECHNICAL REPORT, 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 Percept/on 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, Roger C, Schank, LINGUISTICS FROM A CONCEPTUAL VIEWPOINT (ASPECTS OF ASPECTS OF A THEORY OF SYNTAX), April 21, 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, J,A, Feldman, J, Gips, J, J, Horning, and S, Reder, GRAMMATICAL COMPLEXITY AND INFERENCE, June 1969,

The problem of Inferring a grammar for a Set of symbol strings I s considered and a number of new decldabil ity results obtalned, Several notlons of grammatical complexlty and their properties are stucied, The question of learnlng 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 general lzation through the grammatical Inference question,

AIM-90, Anthony C, Hearn, STANDARD LISP, May 1969, (out of print),

A uniform subset of LISP1,5 capable of assembly under a wide range of existing compliers and Interpreters Is described,

AIM-91, Anthony C, Hearn and J, A, Campbell, SYMBOLIC ANALYSIS OF FEYNMAN DIAGRAMS BY COMPUTER, 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 ln terms of a Hamiltonian of Interaction or Feynman diagrams to the derlvatlon of an absolute square of the matrix element for the process,

AIM-92, Victor D, Schelnman, DESIGN OF A COMPUTER CONTROLLED MANIPULATOR, June 1969, 52 pages,

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 var lous manipulator

configurations, force sensing methods, and suitable components and hardware was first performed, Based on thls study, a general design concept was formulated. This concept was then developed into a detalled manlpulator design, having slx degrees of freedom, all electric motor powered. The manlpulator has exceptionally hlgh position accuracy, compartitively 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, Jerome Feldman, SOME DECIDABILITY RESULTS ON GRAMMATICAL INFERENCE AND COMPLEXITY August 1969, 26 pages, (out of print, revision In preparation),

The problem of grammatical inference Is considered and a number of positive answers to decldablllty questions obtalned, Condltlons are prescribed under which it is possible for a machine to Infer a grammar (or the best grammar) for even the general rewriting systems,

AIM-54, Kenneth Mark Colby, Lawrence Tesler, Horace Enea, EXPERIMENTS WITH A SEARCH ALGORITHM ON THE DATA BASE OF A HUMAN BELIEF STHUCTURE, August 1969, 28 pages.

Problems of collecting data regarding numan bel lefs are considered, Representation of thls 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, tohar Manna, THE CORRECTNESS OF NON-DETERMINISTIC PKOGRAMS, August 1969, 44 pages,

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 wlde variety of possible applications of the results,

AIM-96, Claude Cordell Green, THE APPLICATION OF THEOREM PRUVINC TO QUESTION-ANSWERING SYSTEMS, August 1969, 166 pages,

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 constructlon 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.

AIY-97, Kenneth Mark Colby and David Canfield Smith, DIALOGUES BETWEEN HUMANS AND AN ARTIFICIAL BELIEF SYSTEM, August 1969, 28 pages.

An artificial belief system capable of conducting on-line dialogues with humans has been 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, James Jay Horning, A STUOY OF GRAMMATICAL INFERENCE, August 1969, 166 pages.

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, B,G, Buchanan, G.L. Sutherland and E.A, Feigenbaum, TOWARD AN UNDERSTANDING O F O F INFORMATION PROCESSES OF SCIENTIFIC INFERENCE IN THE CONTEXT OF-ORGANIC CHEMISTRY, September 1969, 66 pages,

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 that utilize it effectively. The initial, rather straightforward, 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 and John McCarthy, PROPERTIES OF PROGRAMS AND PARTIAL FUNCTION LOGIC, October 9969, 21 pages.

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, K. Paul, G. Falk, J.A. Feldman, THE COMPUTER REPRESENTATION OF SIMPLY DESCRIBED SCENES, October 1969, 16 pages.

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 slmiiar objects can be represented In terns of a common part and a number of lndlvlduai parts, If It Is necessary to simulate moving an object then only the Individual information need be saved,

AIM-102, D, A, Waterman, GENERALIZATION LEARNING FOR 'AUTOMATING THE LEARNING OF HEURISTICS, July 1969, 74 pages,

This paper investigates the problem of implementing machine learning of heuristics, First, a method of representing heuristics as production rules is developed which facl i itates 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 dec I sion in draw poker, Finally, problems wh I ch merit further investigatlon are discussed, including the problem of defining the task environment and the problem of adapting the system to board games,

AIM-103, John Allen and David Luckham AN INTERACTIVE THEOREM-PROVING PROGRAM, October 1969, 27 pages,

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, Thls 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 L, Sutherland, Bruce, G, Buchanan, Edward A, Felgenbaum, A HEURISTIC. PROGRAM FOR SOLVING A SCIENTIFIC INFERENCE PROBLEM: SUMMARY OF MOTIVATION AND IMPLEMENTATION, November 1969, 15 pages,

The primary motivation of the Heuristic DENDRAL project Is to study and model processes of inductive inference In science, In particular, the formatlon of hypotheses which best explaln g I ven 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 th I s stage, The paper concludes with a comprehensive list of publications of the project,

AIM-105, M, Heuckel, AN OPERATOR WHICH LOCATES EDGES IN DIGITIZED PICTURES, October 1969, 37 paws,

This paper reports the development of an edge finding operator (subroutine) which accepts the digitized light intensities withln 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 unlque best operator wnIch satisfies a certain set of criteria for a local edge recognlzer, The main concerns of the criteria are speed and reliability in the presence of noise.

KEY WORDS AND PHRASES: artificial Intelligence, picture processing, pattern recognition, edge recognition, edge Operator, primitives of pictures, computer vision, scene analysis, feature extracting, information reduction, noise elimination, heuristic process, Hilbert space.

AIM-106, Michael E. Kahn, THE NEAR-MINIMUM-TIME CONTROL OF OPEN-LOOP ARTICULATED KINEMATIC CHAINS, December 1969, 171 pages.

The time-optimal control of a system of rigid bodies connected in Series by single-degree-0 f-freeqom jolnts 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 snitching curves for each of the system controls, These curves are obtained from the linearized equations of motion for the system, Approxlmatfons are made for the effects of gravity loads and angular velocity terms in the ronilnear equations of motion.

Digital simulation is used to obtaln a comparison of response times of the optimal and suboptimai 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 I S applied to the control of three joints of a mechanical manipulator, Modifications of the suboptlmal 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, December 1969, 27 pages,

The problem of determining the shape and orientation of an object based on one or more two-dimensional Images Is considered, F-or a

AIM-108, Michael D.Kelly, EDGE DETECTION IN PICTURES BY COMPUTER

USING PLANNING, January 1970, 28 pages

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. A 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, and Sylvia Weber, SPINOZA II: CONCEPTUAL CASE-BASED NATURAL LANGUAGE ANALYSIS, January 1970, 107 pages.

This paper presents the theoretical changes that have developed In Conceptual Dependency Tneory and the I r 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 and Zohar Manna, FORMALIZATION OF PROPERTIES OF PARALLEL PROGRAMS, February 1970, 58 pages,

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 formallzatlon can easl ly be extended to more complicated programs,

Alsc 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 formallzatlon gives a practical method for proving properties of such programs,

AIM-111, Zohar Manna, SECOND-ORDER MATHEMATICAL THEORY OF COMPUTATION, March 1970, 25 pages,

In this work we show that It is possible to formalize ail properties regularly- observed in (deterministic and non- deterministic) algorithms In second-order predicate calculus,

Moreover, we show that for any given algorlthm it suffices to know how to formalize Its "partial correctness" by a second-order formula In order to formal lze all other propertlas by second-order formulas,

Thls 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 Symposlum on Theory of Computing (May 1970),

AIM-112, Franklin D, Hi if, Kenneth Mark Colby, David C. Smith, and hliliam K, Wittner, MACHINE-MEDIATED INTERVIEWING, March 1970, 27 pages,

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 M, Colby, Franklin R, Hilf, William A, Hall, A MUTE PATIENT'S EXPERIENCE WITH MACHINE-MEDIATED INTERVIEWING, March 1970, 19 pages,

A hospitalized mute patient participated in seven machine-mediated Interviews, excerpts of which are presented, After the flfth 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, A,W, Biermann and J,A, Feldman, ON THE SYNTHESIS OF FINITE-STATE ACCEPTORS, April 1970, 31 Pages,

Two aigorithms 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 givesamethod 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 algorithmsand their relationship to the problem of grammatical inference,

AIM-115, Ugo Montanari, ON THE OPTIMAL DETECTION OF CURVES IN NOISY PICTURES, March 1970, 35 pages,

A technique for recognizing systems of lines Is presented, In which the heuristic of the problem is not embedded ln the recognition algorithm but is expressed in a figure of merit, A multistage decision process is then able to recognize the Input picture the optimal system of lines according to the given figure of merit, Due to the global approach, greater flexibility and adequacy to 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 lg_2N if an approximate method Is used),

AIM-116, Kenneth MarkColby, M.D., MIND AND BRAIN, AGAIN, March 1970, 10 pages,

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, restricted class of projections it is shown that monocular information Is often "nearly" sufficient for complete specification of the object viewed,

Appendix E

OPERATING NOTES

The Stanford Artificial Intelligence Laboratory has a dual-processor (DEC PDP-10/PDP-6) time-shared computer, It has 131 thousand words of care memory backed by a fast disk (20 mIII ion bits Per second transfer rate) and an IBM 2314 disk file' Numerous display consoles and Teletype terminals are connected,

The Operating Notes (SAILONs) below describe the operation of computer programs and equipment and are intended for project Use' This annotated list omits obsolete notes,

SAILON-2.1, W, Welher, "Calcomp Plot Routines", September 1968, Describes use of basis plot routlnes from elther FORTRAN IV or MACRO,

SAILON-3.1, B, Baumgart, "HOW to Do It and Summaries of Things", March 1969, An Introductory summary of system features (obsolescent),

SAILON-8, S, Russell, "Recent Additions to FORTRAN Library", March 1967,

SAILON-9, P, Petit, "Electronic Clock", March 1967, Electronic clock attached to the system gives time In micro-seconds, seconds, minutes, hours, day, month, and year,

SAILON-11, P, Petit, "A Recent Change to the Stanford POP-6 Hardware", March 1967, The PDP-6 has been changed so that user programs can do their own I/O to devices numbered 700 ,and above,

SAILON-21, A, Grayson, "The A-D Converter", June 1967,

SAILON-21 Addendum 1, E, Panofsky, "A/D Converter Multiplexer Patch Panel and Channel Assignments as of 1-9-69", January 1969,

SAILON-24, S, Russell, "PDP-6 I/O Device Number Summary", August 1967,

SAILON-25, S, Russell, "The Mlsceiianeous Outputs", August 1967, Gives bit assignments for output to hydraulic arm and TV camera positioning,

SAILON-26.2, P, Petit, "FAIL", Apr|1970, Describes one-pass assembler that is about flve times as fast as MACRO and has a more powerful macro processor,

SAILON-28.3, L, Quam, "Stanford LISP 1.6 Manual", September 1969, Describes the LISP interpreter and complier, the editor ALVINE, and other aspects of this venerated list processing system,

SAILON-29, W. Weiher, "Preliminary Description of the Display Processor", August 1967, III display system from the programmer's viewpoint,

SAILON-31, J, Sauter, "DiscDiagnostic", October 1967, A program to test the Librascope Disk and its Interface,

SAILON-35,2, K, Pingle, "Hand-Eye LibraryFile", April 1970,

SAILON-36, G, Feldman, "Fourier Transform Subroutine", June 1968, FORTRAN subroutine performs one-dimensional Fast Fourier Transform,

SAILON-37, S, Russell and L, Earnest, "A,I, Laboratory Users Guide", June 1968, Orientation and administration procedures,

SAILON-37, Supplement 1, J, McCarthy, "A,I, Laboratory Users Guide", June 1968, Hard-line administration,

SAILON-38, P, Vicens, "New Speech Hardware? August 1968, Preprocessor for input to speech recognition systems,

SAILON-39, J, Sauter and D, Swinehart, "SAVE", August 1968, Program for saving and restoring a single user's disk files on magnetic tape,

SAILON-41, L, Quam, "SMILE at LISP", September 1968, A package of useful LISP functions,

SAILON-42, G, Falk, "Vidicon Nolsa Measurements? September 1968, Measurements of spatial and temporal noise on Cohu vidicon camera connected to the computer,

SAILON-43, A, Moorer, "DAEMON - Disk Dump and Restore", September 1968, Puts all or selected files on magnetic tape, New version described in SAILON-54,

SAILON-44, A, Moorer, "FCROX - MACROX to Fall Converter? September 1968, Converts MACRO programs to FAIL format, with a few annotated exceptions,

SAILON-45, A, Hearn, "REDUCE Implementation Guide", October 1968, Describes the procedure for assembling REDUCE (a symbolic computation system) In any LISP system.

SAILON-46, W, Weiher, "Loader Input Format", October 1968,

SAILON-47, and 47 Supplement 1, J, Sauter and J, Singer, "Known Programming Differences between the PDP-6 and PDP-10" November 1968,

SAILON-49, A, Hearn, "Service Routines for Standard LISP Users% February 1969,

SAILON-50.2, S , Savitzky, "Son of Stopgap", April 1970, A line-number-oriented text editor with string search and substitution commands and hyphenless text justification,

SAILON-52.1, A , Moorer, "System Bootstrapper's Manual", February 1969, How to bring back the system from various states of disarray, ,

SAILON-53, R, Neely and J, Beauchamp, "Some FORTRAN I/O Humanization Techniques", March 1969, How to live with FORTRAN crockery'

SAILON-54, A , Moorer, "Stanford A-I Project Monitor Manual! Chapter I a Console Commands", September 1969, How to talk to the timesharing system.

SAILCN-55, A, Moorer, "Stanford A-I ProjectMonitorManual! Chapter II - User Programming", September 1969, Machine language commands to the timesharing system,

SAILON-56, T, Panofsky, "Stanford A-I Facility Manual? Computer equipment features (in preparation),

SAILON-57, D, Swinehart and R, Sproull, "SAIL", November 1969, ALGOL-60 compiler with LEAP constructs and stringprocessing,

SAILON-58, P , Petit, "RAID", September 1969, Display-oriented machine language debugging package,

SAILON-59, A,, Moorer, "MONMON", October 1969, Lets YOU peer Into the TS monitor,

SAILON-60, L, Earnest, "Documentation Services", February 1970, Text preparation by computer is often cheaper then typewriters, Facilities for text preparation and reproduction are discussed,

Contribution from the DENDRAL project

towards the summary of REsearch in the Computer Science Dept, 1970

E. A. Feigenbaum

J. Lederberg

B. G. Buchanan

G. L. Sutherland

The "scientific method" involves two very different kinds of intelligent behavior, sometimes called induction and deduction respectively. A theory is somehow "induced", sometimes out of sheer speculation, in order to account for some hitherto baffling or provocative observations of nature. Then, the theory is applied deductively, i.e., logically or mathematically rigorous conclusions are made. If the theory is true, then certain results must be obtained.

The process of logical deduction follows rules which, at least at an elementary level, are well understood. Correspondingly, computers have been extensively used for deductive calculations. Scientific induction, on the other hand, remains a mysterious process, connected to the most "creative" aspects of human thinking, and is difficult to implement on a computer.

The DENDRAL project aims at emulating in a computer program the inductive behavior of the scientist in an important but sharply limited area of science, organic chemistry. Most of our work is addressed to the following problem: Given the data of the mass spectrum of an unknown compound, induce a workable number of plausible solutions, that is, a small list of candidate molecular structures. The Heuristic DENDRAL program does this, and, in order to complete the task, the program then deduces the mass spectrum predicted by the theory of mass spectrometry for each of the candidates, and selects the most productive hypothesis, i.e., the structure whose predicted spectrum

most closely matches the data.

We have designed, engineered, and demonstrated a computer program that manifests many aspects of human problem-solving techniques. It also works faster than human intelligence in solving problems chosen from an appropriately limited domain of types of compounds, as illustrated in the cited publications. (13,17)

Some of the essential features of the DENDRAL program include:

1) Conceptualizing organic chemistry in terms of topological graph theory, i.e., a general theory of ways of combining atoms.

2) Embodying this approach in an exhaustive hypothesis generator. This is a program which is capable, in principle, of "imagining" every conceivable molecular structure.

3) Organizing the hypothesis generator so that it avoids duplication and irrelevancy, and moves from structure to structure in an orderly and predictable way.

The key concept is that induction becomes a process of efficient selection from the domain of all possible structures. Heuristic search and evaluation is used to implement this "efficient selection". Most of the ingenuity in the program is devoted to heuristic modifications of the generator. Some of these modifications result in early pruning of unproductive or implausible branches of the search tree. Other modifications require that the program consult the data for cues (pattern analysis) that can be

used by the generator as a plan for a more effective order of priorities during hypothesis generation. The program organization is aimed at facilitating the entry of new ideas by the chemist when discrepancies are perceived between the actual Functioning of the program and his expectation of it.

The Heuristic DENDRAL process of analyzing a mass spectrum* consists of three phases. The first, preliminary inference (or planning), obtains clues from the data as to which classes of chemical compounds are suggested or forbidden by the data. The second phase, structure generation, enumerates chemically plausible structural hypotheses which are compatible with the inferences made in phase one. The third phase, prediction and testing (or hypothesis validation), predicts consequences from each structural hypothesis and compares this prediction with the oripinal spectrum to choose the hypothesis which best explains the data. Corresponding to these three phases are three sub-programs. The programs have been described in previous publications, primarily in the books Machine

**********footnote*******

* For this discussion it is sufficient to say that a mass spectrometer is an instrument into which is put a minute sample of some chemical compound and out of which comes data usually represented as a two dimensional histogram. This is what is referred to here as the mass spectrum. The instrument itself bombards molecules of the compound with electrons, thereby producing ions of different masses in varying proportions. The points on the abscissa of the spectrum represent the masses of ions produced and the points on the ordinate represent the relative abundances of ions of these masses.

****

Intelligence IV and V (8,14) and in a series of Stanford Artificial Intelligence Project Memos (8,11,14,15).

The attached references report the practical application of DENDRAL as an aid in solving problems of chemical structure. While our main interest in DENDRAL is as a prototype of scientific induction, it may have specific application in guiding the closed-loop automation of an analytical mass spectrometer or general chemical fractionation system.

Three papers have appeared in the Journal of the American Chemical Society (12,13,17) and a fourth has been submitted. The first paper describes the Heuristic DENDRAL program and tabulates numbers of chemical 1 y plausible isomers for many compounds. This is of particular interest to chemists because it indicates the size of the search space in which structures must be found to match specific data. The second paper explains the application of the program to ketones: the subclass of molecular structures containing the keto radical. The whole process from preliminary inference (planning) through structure generation and prediction of theoretical spectra was applied to many examples of ketone spectra. The results, in terms of actual structures identified, are encouraging. The third paper explains the application of the program to ethers. A second kind of data, NMR spectroscopy, was introduced to the process and aided considerably in the successful processing of ether compounds. The fourth paper (18) describes the application of Heuristic DENDRAL techniques to the class of chemical compounds called amines. In this case, even with

both mass spectral data and NMR data, the Heuristic DENDRAL program was not able to limit the list of potential hypotheses to a small number. However, by using additional information available within the NMR data, an additional program was written, which performs the hypotiwsis selection and validation tasks independently of the previous programs. This technique worked very well for amines, and is thought to be applicable to a limited, but significant, class of other compounds as well.

One reason for the high level of performance of the Heuristic DENDRAL program is the large amount of mass spectrometry knowledge which chemists have imparted to the program. Obtaining this has been one of the biggest bottlenecks in developing the program. It should be understood that there presently is no axiomatic or even well organized theory of mass spectrometry which we could transfer to the program fron a text hook or from an expert. Most of the chemical theory has been put into the program by a programmer who is not a chemist but who spent many hours in eliciting the theory from the chemist-expert. In many cases the chemists's theory was only tentative or incompletely formulate& so that many iterations of rule formulating, programming, and testing were necessary to bring the DENDRAL program to its present level of competence.

A few general points of strategy have emerged from the DENDRAL effort. With regard to the theoretical knowledge of the task domain in the program, we believe that the following considerations are important.

1) It is important that the program's "theory of the real world" be centralized and unified. Otherwise, during the evolution of this theory, the program will inevitably accumulate inconsistencies.

2) It would be advantageous for the program to derive planning cues from its own theory, by introspection, rather than from external data which may not yet have been assimilated into its theory. The success of the program depends in every case on the validity of the theory, so there is no use going beyond it. It is more efficient for the computer to generate hypothetical spectra and search for the relevant "diagnostic" patterns in them than to wait for experimental data. The theory should be responsive to the data; then the list of inference cues should be generated from the theory.

3) Separating the theory from the routines which use it facilitates changing the theory to improve it, on the one hand, or to experiment with variations of it, on the other. Although scattering the theory in the program's LISP code increases running efficiency, it seems more desirable, at this point, to increase the program's flexibility. This has led us to design the programs in a form we refer to as "table-driven".(14)

4) The idea for a new, "Meta-DENDRAL", program is emerging. Meta-DENDRAL will be a program which can generate alternate theories of mass spectroscopy, analogous to the way in which the Heuristic DENDRAL program generates alternate structures to explain a given mass spectrum. A theory of mass spectroscopy is built upon primitive operations to explain the mechanisms thought to be underlying different spectra. The present Heuristic DENDRAL program contains one such theory of mass spectroscopy. But there are possibly other theories, and the job of Meta-DENDRAL will be to generate these automatically by computer.

5. The consistency problem mentioned in #1 above would evaporate if there were just one representation of the chemical theory which could be read by all parts of the system which use the theory. But it may be unreasonable to expect to find one representation which is suitable for all purposes. A solution is to add either (a) A program which can read both representations of the theory to check for inconsistencies, or (b) a different representation to which modifications will be made and a program which writes the other two representations from the third after each set of changes. In either case, the program which will result will be part of Meta-DENDRAL, since it deals with more global problems of science than does the Heuristic DENDRAL program which deals specifically with the problems of mass spectrometry.

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 (10)).

(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) 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 Project Memo No. 54, August 1967).

(7) J. Lederberg, "Online computation of molecular formulas from mass number." NASA CR-94977 (1968)

(8) 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 Project Memo No. 62, July 1968).

(9) 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 Project Memo No. 67, August 1968).

(10) J. Lederberg, "Topology of Molecules", in The Mathematical Sciences - A Collection of Essays, (ed.) 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.

(11) G. Sutherland, "Heuristic DENDRAL: A Family of LISP Programs", to appear in D. Bobrow (ed), LISP Applications (also Stanford Artificial Intelligence Project Memo No. 80, March 1969).

(12) 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).

(13) 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).

(14) 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 Project Memo No. 99, September 1969).

(15) 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 Project Memo No. 104, November 1969.

(16) 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.

(17) 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).

(18) 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", submitted to the Journal of the American Chemical Society, March 1970.

COMPUTER SIMULATION OF BELIEF SYSTEMS

Kenneth Mark Colby, M.D., who is a Senior Research Associate in the Computer Science Department, terminated his private practice of psychiatry to devote full time to investigations in this area of computer simulation. The National Institute of Mental Health sponsored two projects under Dr. Colby's direction. One of these is a Research Career Award and the other is a research project which continues the investigations in which his group has been engaged for the past seven years.

Research Plan

A. Introduction and Specific Aims:

The clinical problems of psychopathology and psychotherapy require further investigation since so little is known about their essential processes. Some of this ignorance stems from a lack at a basic science level of dependable knowledge regarding higher mental processes such as cognition and affect. The research of the project attempts to approach both the clinical and basic science problems from the viewpoint of information-processing models and computer simulation techniques. This viewpoint is exemplified by current work in the fields of cognitive theory, attitude change, belief systems, computer simulation and artificial intelligence.

The rationale of our approach to these clinical problems lies in a conceptualization of them as information-processing problems involving higher mental functions. Computer concepts and techniques are appropriate to this level of conceptualization. Their success in other sciences would lead one to expect they might be of aid in the areas of psychopathology and psychotherapy.

The specific aims of this project relate to a long-term goal of developing more satisfactory explicit theories and models of psychopathological processes. The models can then be experimented with in ways which cannot be carried out on actual patients. Knowledge gained in this manner can then bc applied to clinical situations.

B. Methods of Procedure:

We have now gained considerable experience with methods for writing programs of two types. The first type of program represents a computer model of an individual person's belief system. We have constructed two versions of a model of an actual patient in psychotherapy and we are currently writing programs which simulate the belief systems of two normal individuals. We have also constructed a model of a pathological belief system in the form of an artificial paranoia. A second type of program represents an interviewing program which attempts to conduct an on-line dialogue intended to co-llect data regarding an individual's interpersonal relations. We have written two such interviewing programs and at present we are collaborating with psychiatrists in writing a program which can conduct a diagnostic psychiatric interview.

A computer model of a belief system consists of a large data-base and procedures for processing the information it contains. The data-base consists of concepts and beliefs organized in a structure which represents an individual's conceptualization of himself and other persons of importance to him in his life space. This data is collected from each individual informant by interviews. Verification of the model is also carried out in interviews in which the informant is asked to confirm or disconfirm the outcome of experiments on the particular model which represents his belief system. Because of the well-known effects of human interviewer bias, the process of

data-collection and verifications should ideally be carried out by on-line man-machine dialogues and this is a major reason for our attempt to write interviewing programs. However, the difficulties in machine utilization of natural language remain great and until this problem is reduced we must use human interviewers.

We have written one type of therapeutic interactive program which is designed to aid language development in nonspeaking autistic children. We have used it for the past two years on eighteen children with considerable success (SO% linguistic improvements). We intend to continue using this program and to instruct professionals in psychiatry and speech therapy in how to write, operate and improve such therapy programs for specific conditions.

C. Significance of this Research:

This research has significance for the psychiatry, behavioral and computer sciences.

Psychiatry lacks satisfactory classifications and explanations of psychopathology. We feel these problems should be conceptualized in terms of pathological belief systems. Data collection in psychiatry is performed by humans whose interactive effects are believed to account for a large percentage of the unreliability in psychiatric diagnosis. Diagnostic interviewing should ideally be conducted by computer programs. Finally, the process and mechanisms of psychotherapy are not well understood. Since experimentation on computer models is more feasible and controllable than experimentation on patients, this approach may contribute to our understanding of psychotherapy as an information-processing problem.

It is estimated that 90% of the data collected in the behavioral sciences is collected through interviews. Again, a great deal of the variance should be reduced by having consistent programs conduct interviews. Also, this

Recent Publications

1. Colby, K. M. Experimental treatment of neurotic computer programs. Archives of General Psychiatry, 10, 220-227 (1964).

2. Colby, K. M. and Gilbert, J. P. Programming a computer model of neurosis. Journal of Mathematical Psychology. 1, 405-417 (1964).

3. Colby, K. M. Computer simulation of neurotic processes: In Computers in Biomedical Research, Vol. I, Stacey, R. W. and Waxman, B. Eds., Academic Press, New York (1965).

4. Colby, K. M., Watt, J. and Gilbert, J. P. A computer method of psychotherapy. Journal of Nervous and Mental Disease, 142, 148-152 (1966).

5. Colby, K. M. and Enea, H. Heuristic methods for computer understanding in context-restricted on-line dialogues. Mathematical Biosciences, Vol. I, 1-25 (1967).

6. Colby, K. M. Computer simulation of change in personal belief systems. Behavioral Science, 12, 248-253 (1967).

7. Colby, K. M. A programmable theory of cognition and affect in individual personal belief systems. In Theories of Cognitive Consistency, Abelson, R., Aranson, E., McGuire, W., Newcomb, T., Tannebaum, P. Eds., Rand-McNally, New York (1967). In Press.

8. Colby, K. M. and Enea, H. Inductive inference by intelligent machines. Scientia (1967). In Press.

9. Tesler, L., Enea, H. and Colby, K. M. A directed graph representation for computer simulation of belief systems. Mathematical Biosciences, 1 (1967). In Press.

10. Colby, K. M. Computer-aided language development in nonspeaking autistic children. Technical Report, No. CS 85 (1967), Stanford Department of Computer Science.

11. Enea, H. MLISP. Technical Report, CS 92 (1968), Stanford Department of Computer Science.

12. Colby, K. M., Tesler, L., Enea, H., Search Experiments with the Data Base of Human Belief Structure, (paper to be delivered at International Joint Conference on Artificial Intelligence, Washington, D. C., May '69).

13. Colby, K. M., Tesler, L. and Enea, H. Experiments with a Search Algorithm on the Data Base of a Human Belief Structure. Stanford Artificial Intelligence Project Memo A-I 94. Computer Science Department, Stanford University, August, 1969, (To appear in Proceedings of the International Joint Conference on Artificial Intelligence, Walker and Norton (Eds.), in press)

14. Colby, K. M. and Smith, D. C., Dialogues Between Humans and An Artificial Belief System. Stanford Artificial Intelligence Project Memo A-I 97. Computer Science Department, Stanford University. (To appear in Proceedings of the International Joint Conference on Artificial Intelligence, Walker and Norton (Eds.), in press)

13. Colby, K. M. Critical Evaluation of "Some Empirical and Conceptual Bases for Coordinated Research in Psychiatry" by Strupp and Bergin. International Journal of Psychiatry, 7, 116-117 (1969).

16. Colby, K. M., Weber, S. and Hilf, F. Artificial Paranoia (In preparation)

17. Colby, K. M., Hilf, F. and Hall, W. A Mute Patient's Experience with Machine-Mediated Interviews. Stanford Artificial Intelligence Project Memo A-I 113. Computer Science Department, Stanford University.

18. Hilf, F., Colby, K. M., Wintner, W. Machine-Mediated Interviewing (In preparation).

research has significance for cognitive theory, attitude change and social psychology.

Computer science is concerned with problems of man-machine dialogue in natural language, with optimal memory organization and with the search problem in large data-structures. This research bears on these problems as well as on a crucial problem in artificial intelligence, i.e., inductive inference by intelligent machines.

D. Collaboration:

We are collaborating with two psychiatric centers for disturbed children and a local VA hospital. We are also collaborating with residents in the Department of Psychiatry and with graduate students, in computer science, psychology, education and electrical engineering.

Programming Models and the Control of Computing Systems

Work to Date

The first phase of this research was concerned with the development of the graph program model and the demonstration of its suitability for representing a broad class of computational problems (Duane A. Adams, "A Model for Parallel Computations", June 1969, Proceedings of Symposium in Parallel Processor Systems Technology and Applications, Monterey, California). The model, as developed by Adams, is concerned with free running programs, that is, programs for which one could assume there were sufficient resources that the computation does not come to a halt for lack of resources. Such a model is useful for examining the representation of parallel computations and studying the maximum resource requirement.

The second phase of this study is concerned with resource allocation on graph program computations in two different environments: (1) the free running and (2) resource limited.

The graph program model, as developed by Adams, is one in which the control of the computing system is governed by the flow of data through it. The model includes the definition of data types and primitive nodes that define the possible classes of computation. It also includes a hierarchical structuring of programs including graph procedures which permit recursive computations. The data item has no permanent location as representation, but rather "travels" along the edges of the graph to the

operations which are performed on it. This model permits one to program sophisticated algorithms such as matrix inversion in a way which allows both the single construction stream type of parallelism and multiple construction stream parallelism down to a very low level. The rules for creating an Adams' graph include implicitly the sequencing control necessary to permit parallelism of computation. This approach is to be contrasted with approaches for which the programmer must explicitly specify the parallelism in the program. The latter cases are characteristic of programs for the ILLIAC IV, systems that include the fork and join statements, and systems employing Dijkstra semaphore primitives as devices whereby the programmer can represent parallel computations. In the graph program model operations are assumed to be implicitly simultaneous unless they are logically dependent upon one another, as indicated by the precedence in the directed graph representation of the computation. In this directed graph model, the nodes of the graph represent operations performed on data stored on it as directed into the node.

The graph program simulator has been written (Edward Nelson, Graph Program Simulation). This simulator interprets Adams's graph programs, carries out the computations specified by the graph procedures, keeps statistics on the timing and resource usage, and thereby permits one to-do studies on resource allocations for such parallel computations. The simulations were run on a number of small programs including a matrix multiply program, a quadrature program and a sort program. The programs were-run using varying amounts of data and varying assumptions about speeds of the primitive operations. Another variation was to run with and without vector parallelism, that is, the multiple execution of instances of a given

p node. All of the simulations run were done so under the assumption that the machine specified by the simulator had an unlimited number of processors to carry out the operations specified by the primitive nodes and an unlimited amount of memory. Of course, that is an unrealistic assumption. These simulations were run in an attempt to discover the inherent resource usage characteristics of programs operating under the constraints of the Adams' program graph model.

The simulator, which is described in Nelson's draft report along with the experiments run on it, has the following characteristics. It may be thought of as a parallel computer with the following components: (1) storage for graph procedures, (2) storage for data (edges), initiation queues and that status of nodes and edges in executing graph procedures, (3) a pool of processors with input and output registers, (4) logic for performing the operations specified by the primitive nodes, (5) control logic for determining which nodes are ready to execute, assigning processors to these nodes, recognizing when a process is done, and putting the results on the output edges in the order dictated by the initiation queue. Besides these components, the simulator also has the code necessary to gather statistics on the simulation and provide a trace of the computation.

Two distinct machine models are possible for the simulator. One in which each processor is a specialized functional unit able to execute only a single type of primitive node and one in which the processors are all general processors so each can execute all the primitive nodes.

In terms of an actual implementation the first node has the advantage that it is not necessary to duplicate the decoding and control circuitry required to decode operations in each processor. It has the disadvantage

of limiting the flexibility of the processor allocation algorithm; in addition, if the mix of functional units available on the machine does not closely match that required by a given program many of the functional units will be idle much of the time. The savings gained by not duplicating control circuitry may thus be lost by decreases in efficiency. The distinction between the two models is not too important in the unlimited resource environment since it does not make sense to ask what the optimum ratio of adders to multipliers is if one has infinite supply of both. In the finite environment, however, the simulator can be used to determine the cost in functional unit idleness of the first mode; these costs could then be weighted against the cost of duplicating control circuitry.

The two issues studied to present with the simulator are: (1) the dependence on problem size of computation time and the amount of parallelism during execution, and (2) the dependence of these measures on relative processor speed. Even for such an inherently sequential procedure as the square root, the average number of nodes executing per execution step was 1.7. This indicates that even in such sequential programs, a great deal of the operation can go on in parallel. In the matrix multiplication computation, a very clearly parallel program, the n cubed operations are required to multiply two n by n matrices.. The program represented by a program graph executes in a time proportionalto n ; that is, on the average there are n^2 operations executing in parallel.

We have made changes in the simulator to treat the resource limited cases, that is, cases which are processor limited and edge limited program graph terminate, and still guarantee that the execution is determinate.

Edward Nelson	"Graph Program Simulations"
Henry Bauer and Harold Stone	"The Scheduling of N Tasks with M Operations on Two Processors," STAN-CS-70-165, July 1970
Victor Lesser	"A Computer Model for the Definition and Execution of Complex Information Processing Systems", May 1969
Victor Lesser	"A Variable Control Structure Micro-Computer Architecture or Emulation Via An Instantiation Language", July 1970
Harold S. Stone	"Parallel Processing with the Perfect Shuffle"', STAN-CS-70-158, March 1970
Harold S. Stone	"The Spectrum of Incorrectly Decoded Bursts for Cyclic Burst-Error Codes", STAN-CS-70-154, February 1970

GRAPHICS RESEARCH

Currently, there are two research projects in computer graphics at the Linear Accelerator Center sponsored by NSF.

One involves the implementation and use of a graphical meta-system. This meta-system permits economical experimentation with graphical applications of which Shaw's PDL is an example. It allows the application to be developed interactively or in a batch mode and the final version can be used interactively or in a slave mode, with device independence.

To date, the applications have included a two-dimensional expression display system which can be used interactively on the IBM 2250 Graphics Display or with a card‘reader and printer. It can also be used in the slave mode and has been utilized in this slave mode by-a printing formatting program, thereby allowing a mixture of normal text and two-dimensional expressions to be generated on output.

The other application allows test cases to be constructed which are then converted to digital form to be used in pattern recognition experimentation.

The second project is concerned with the design of an interactive system for displaying large time dependent data files. This has included the development of the techniques to produce high quality computer generated films including color and 3D. An experimental program allows one to visually correlate multivariate geophysical time series; this program has been used to view past seismic activity in various ways.

Accelerator Control

The disk monitor operating system for the PDP-9 has been completed and checked out. The various user programs such as klystron replacement, logging, quadsetting, are rewritten and will be incorporated during the accelerator down-time June 25-July 10. The next cycle then should run under the new system. This system has a 32K virtual memory in 512 word page increments, which will allow new functions (such as CCR/DAB consolidation) to be added without requiring additional hardware core memory.

The software I/O handler for the fast A/D system (providing the computer with pulse-to-pulse values of log Q, x, y for all 30 sectors) was also written and checked out during this period. However, as yet, this data is not being used for any particular function by the computer.

Work is continuing on the software aspect of the CCR/DAB consolidation project. Hardware specs for the computer link (SDS 925 in DAB, PDP-9 in CCR) are firm enough to being programming the I/O handlers. Formats for link messages have been agreed upon and data structures for the internal represent+%;: of the touch panel images are being developed.

Graphics Interpretation Facility

During this period the GIF hardware configuration was completed with the acceptance of a magnetic tape unit and drum memory unit, We also acquired an auxiliary keyboard and an Arriflex 16 mm..motion picture camera with associated equipment. The interface for the camera was also built and debugged.

An interactive debugging program was developed that allows a user to trace through a 360 Assembler language routine examining registers and making symbol ic references to his own program. This routine is described in CGTM #92.

From March through June', the following projects were initiated on the Graphics Interpretation Facility:

1. A general Varian 620/I I/O expansion unit was designed, built, and checked out. The function of the I/O expansion unit is to allow an unlimited number of I/O devices to be connected to the Varian 620/I computer.

2. A motion picture camera controller from the Varian 620/I was designed, built, and checked out. The function of this device is to allow the motion picture camera to be driven in synchronization with pictures generated from an IDIIOM display system.

3. A controller for a solid-state keyboard to be interfaced with the Varian 720/I computer was designed and is being built presently.

The SLAC Scope Package for the IDIIOM

This scope package is a set of PL/1 procedures which a programmer may use to write interactive programs for the IDIIOM display console. During the past 6 months this package has been expanded to make it more versatile and easier to use.

The major addition has been the ability to synchronize display regeneration with a 16 mm. Arriflex motion picture camera. It is possible to use the camera in two basic modes.

1. Frame Synchronized Mode: The camera runs at its normal speed while the IDIIOM synchronizes the display with the open-shutter periods. This should enable one to take flicker-free movies of most display programs using this package.

2. Animation Mode: The IDIIOM sends signals to the camera telling it when to open and close its shutter. This mode should enable one to generate color and 3-D movies using the IDIIOM.

SLAC Spiral Header

During the period between January through February 1970, final tests were made on the spiral reader before it was officially turned over to the Conventional Data Analysis group on March 1, 1970. A comparative test was made between the spiral reader and the NRI measuring machines, and the results showed the spiral reader to be as accurate and reliable as the NRI measuring machines. With a full staff of operators and the spiral reader working normally, one could expect the spiral reader to put out well over 1200 events per day. Normally, the expected average output from the spiral reader is about: 900 events per day.

Numerical Analysis

A [illegible] subroutine for the integration of ordinary differential equations was completed, tested, and placed in the SFSCC library. This program will handle both non stiff and stiff equations. It selects both the step size and order of the method automatically.

An experimental version of a program which solves simultaneous systems of algebraic and differential equations has been written and tested, The theory behind the new techniques in this program has been written up in SLAC publication 723, "The Simultaneous Numerical Solution of Differential-Algebraic Equations".

Access Control and Integrity of Data Files

This project is concerned with the access control and integrity of data files such as physics data files, management information, and personnel files. It includes problems of the organization of the control program for implementing access control procedures as well as problems of reliability,

cost, and logical soundness of the procedures themselves. It includes investigation of both batch processing and interactive systems. In May 1970, the basic machine-independent, file structure-independent model was described in a Ph.D. dissertation issued as SLAC Report 117, "The Formulary Model for Access Control and Privacy in Computer Systems". A more application-oriented companion report, "The Engineering of Access Control Mechanisms in Physics Data Bases" (SLAC Report 118) has been completed and is currently being printed by the Publications Group of SIX. The model used in these reports has been tested by 360/91 programs and proved sound.

Controls of these kinds are vital to systems such as the storage ring control and data analysis system or the large magnetic spectrometer data acquisition and analysis system. Whenever two different kinds of programs are interacting with a common data base, the problem of the access control and the passage of the control from one user type to another requires precise definition and formal procedures to prevent unauthorized access or improper synchronization of accesses.

This project and the Accelerator Control project has been interacting in the design of a sophisticated general resource allocation system, including a scheduler.

This work has fallout in other areas such as personnel, medical records, and behavioral science data bases. Experiments run on the 360/91 have demonstrated the general applicability and soundness of the model, and have indicated that the additional CPU overhead introduced by use of the model and its inherent flexibility do not necessarily lead to any decrease in system throughput.

SMEDIT, A Simple-Minded Editor

The editor contains a number of improvements over the IBM utilities for maintaining files of source statements, particularly when it is not possible to place sequencing information on the records of the file.

This Simple-Minded Editor was subsequently rewritten to include a number of new features, the most powerful of which is the ability to move pieces of text around in a card file, and to include sequences of statements from external sources. This makes such tasks as the inclusion of standard COMMON-DIMENSION-EQUIVALENCE decks in a program a very simple matter. The program was written up in CGTM 88, which contains a large number of examples of the use of the editor.

Multiple Precision Divide Routine

The Multiple Precision Divide Routine was rewritten to incorporate tests for a special case of divisor-dividend incompatibility.

FORMAT Program

The FORMAT program was modified on two separate occasions: the first time was to add a control card for controlling the field scan on control statements: this allowed FORMAT to be used from CRBE terminals, and permitted

text-input files to be edited using the internal edit feature of FORMAT. The second set of additions and modifications was more extensive: the underlining algorithm was placed under the control of the user, who may now specify whether he wants to underline leading and training special characters; the "COPY" facility was rewritten to reduce the overhead (the time for two copies of a large file was reduced from 82 to 18 seconds); a control statement was added to allow the user to specify whether sentences are to be separated by two spaces, or a single space; and a long-standing bug in the editor facility was found and corrected. Also, a catalogues procedure was written for FORMAT and it is now in the system procedure library. Appropriate accompanying Newsletter articles were written to describe each of the revisions mentioned above.

SPASM Fast Assembler

The SPASM fast assembler was upgraded to provide a macro capability, with a number of differences in language and capability over the IBM Macro-Assembler Level E. Among the most important are: the ability to define macros anywhere in the source stream; macros may be defined by macros; a greatly improved control is given to the user over the evaluation of terms appearing in conditional-assembly statements; restrictions on syntax and ordering of statements were relaxed; assembly speeds on the Model 67 remain at about 8000-9000 statements per minute, and about 35000 statements per minute on the Model 91. The ability to define DSECTs and CSECTs was added, making it possible to handle most of the common program structures processed by the IBM linkage editor; this capability will make it very simple to produce relocatable text if such is desired later.

SLAC System Programming Language

Preliminary design work on a higher level language for the construction of operating systems has been initiated. The goals include development of succinct constructs for the control, synchronization, scheduling, etc. functions of the system.

A Draft Report on SSPL (April 14, 1970) emphasized the area of structures and access to them, and extensions of Dijkstra semaphores to permit more flexible interprocess communication as well as synchronization. The feasibility of an efficient implementation of these features was investigated to ensure that they could be sufficiently quick to be used as general mechanisms throughout the system.

Although the language is in no sense completely specified or totally refined, we expect to code an interactive text editor using the present version in order to uncover difficulties and/or strengths.

MLP900 Assembler Implementation

The micro processor assembler language for the MLP900, Octavia, is being implemented as a single pass assembler. The assembler is designed to translate a source program coded in Octavia into a MLP900 machine language program. The assembler also performs other auxiliary assembler functions

as designated by the programmer. The assembler itself is coded in PL/1 in total with. the only lower level language routines being those of the PL/1 library system.

The design of the assembler consists of a set of procedures or sub-routines which perform the various functions. They are in general:

A Main Criver: This routine.obtains the source code, extracts . the operation code and calls upon a specific routine to process the statement. The driver also performs various initialization and cleanup procedures.

Machine Operation (lode Routines: These routines function according to the specific operation code encountered and construct the corresponding machine language instruction.

Assembler P. . . o-operation Code Routines: These routines process the assembler pseudo-operations, such as space, title, etc.

Scanner Routine. This routine locates a file on the source statement.

Symbol Table Routine: This routine uses a hash code based on the symbol to insert symbols into the symbol table and locate previously defined symbols. Chaining is used to locate or enter synonyms.

Expression Evaluation Routine: This routine evaluates expressions when the terms have been previously defined.

Fix-up Table Routine: This routine enters fix-up information for partially defined expressions.

Print-out Routine: This routine prepares a temporary print image data set during assemble and outputs same at end of assemble process.

Cross Reference Routine: This routine prepares a temporary data set containing the symbol and statement number of symbols referenced.

Machine Code Output Routine: This routine prepares a data set containing the-assembled machine code in a format acceptable to the loader.

The bulk of the above routines have been coded. Checkout is now in process. The cross reference routine lacks a sort procedure but if sufficient core is available then the IBM supplied sort will be linked to it; otherwise we will have to construct a sort routine. The machine code output routine has not been coded yet.

Several pseudo operations, such as DATA and the initialization of the register groups instruction INITR have not been coded yet.

Currently, the symbol table will handle a maximum of 4096 distinct symbols. The machine code generated is stored in core.

The requirements to run the assembler are a 300K region, standard input and output, two temporary data sets, one for the intermediate print data set, and the other for the temporary cross-reference entries, and one data set for the machine code output.

Two-Dimensional Pattern Recognition

Experiments have been carried out to determine the usefulness of Fourier boundary descriptors for discriminating among various two-dimensional shapes. Fortunately, it seems to require only very few of the low order Fourier descriptors to identify distinct shapes accurately. The most significant aspect of this family of shape features is their generality. The theory makes no assumptions on what sort of picture the figure shapes were derived from; any simple closed polygonal curve represents an admissible shape.

The paper (SLAC Publication 672) entitled "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters" has been revised and expanded by the'inclusion of several computer experiments to test the theory. It is being resubmitted to IEEE Transactions on Computers where it will be published later this year. Several ideas for using these cluster techniques to organize the data from a bubble chamber photograph have been developed and we hope to make some computer tests of their validity.

A Simulation Package for Computer Design

The implementation of this package of programs has progressed in the first half of 1970 in the following ways:

- The interpreter which'is the core of the simulator has been programmed and is in the process of being tested. Documentation of the interpreter is under way.

A "load format" has been defined for the descriptions to the input to the interpreter, a load phase has been programmed, tested and appended to the interpreter program. The loader and interpreter currently make up a 1150-statement PL/I program which is now being tested as a unit.

A description of the MICRO-800 computer has been written out in the "high-level" (PL/I-like) language, and partially translated (by hand) into the load format for the interpreter. This description is being used as the first sample program for testing of the-loader-interpreter program.

- Work has begun on writing a compiler program for translation of "high-level" language descriptions directly into the load format.

Microprogrammed Implementation of SNOBOL4

Since December 1969, 94 of the 128 SNOBOL4 macro instructions have been implemented in an interpreter written in IBM 360 machine code. The remaining 34, however, are very important because they deal with stack management, construction of tables, and control of calling sequences to internal routines. In other words, the static parts of the SNOBOL4 compiler-interpreter

can now be implemented within the 360 interpreter, but the dynamics remain undone. Our interest is now shifting away from this approach, at least temporarily.

A possible better approach

Increased familiarity with SNOBOL4 has made it less clear that direct microprogramming of the macro language is the best way to go about increasing efficiency. Certain procedures within the program such as pattern construction, pattern matching, string construction and storage allocation and regeneration consume a great deal of the running time. These procedures have no compact symbolic representation like that of the macros, but they could be micro-programmed. It might well be that greater efficiency would be achieved by microprogramming the pattern matching procedure than by microprogramming many macros. Consequently, an alternative approach is to examine these larger scale operations and attempt to determine how they could be micro-programmed,. how much this would increase efficiency, and how to program the macros displaced from the micro-computer.

Unfortunately, these more complex procedures are not as well documented as the macros; they are not as easily microprogrammed. To answer the questions raised in the previous paragraph; it is necessary to probe and more thoroughly document the innards of the SNOBOL4 compiler-interpreter. Because the prospects seem good that this approach will lead to a more efficient micro-programmed SNOBOL4 compiler-interpreter than simply translating the macro-instructions into microcode, we plan to devote the next two months to further documenting such internal procedures as pattern construction and matching, string construction and manipulation, storage allocation and regeneration, and general behavior of the interpreter.

As we discern and document the structures at the lower levels of SNOBOL4, we hope to begin designing implementation of these structures in microcode. An MLP900 will serve for experimentation along these lines (CGTM 91). At present there is a simulator of the MLP900 available at SLAC. In the next months both macros and less well defined structures of SNOBOL4 will be coded and run on the simulator. However, in the immediate future emphasis will be on documentation and design rather than implementation.

OCTAVIA - A Microprogramming Assembly Language for the MLP900

-The main goal in designing OCTAVIA was to maximize the readability of programs written in it and to make the very complex MLP900 miniflow as easy to use as possible.

An OCTAVIA source text consists of statements subdivided into the conventional fields: name, operation, operands, comments. The fields are separated by blanks, and requirements on formats of the fields are minimal. The statements specify MLP900 ministeps or one of the assembler pseudo instructions mentioned below. Comment and blank statements are also provided. The last statement of any program is an END statement.

Data types and pseudo instructions:

Most concepts of OCTAVIA are close to those of conventional assembly languages. It allows for symbolic addressing, the specification of'binary, decimal, octal, hexadecimal and character self-defining terms of 36-bit length, and for expressions evaluating to 36-bit absolute values or to 16-bit control memory addresses.

A new data type "RF designators" is introduced to denote'operands which refer to MLP900 registers and state flip-flops,' Basic RF designators are assembly defined RF constants, but they may also be specified symbolically.

Symbolic specifications are done in general via identifiers. An identifier may be defined by its appearance in the name field of an instruction or by an EQU pseudo instruction assigning to it the value of an expression or of an RF designator. In order to provide identifier localization, the assembly can be divided into sections by SECT statements. Identifiers are local to the section where they are defined unless they are declared in that section to be common by a COMSY statement. Any section that references a common identifier must also contain a COMSY statement for it.

The statements ORG and ALIGN serve to manipulate and align the location counter; BES and BSS will reserve a block of storage beginning (ending) with a specified symbolic name; DATA can be used to initialize a block of data to certain expression values. INITR will initialize MOP900 registers at run time; MASK allows for specifying a default mask register used as an operand in several ministeps. Conditional assembly is made possible by the statements SKIPT and SKIPF. There are various statements like EJECT, PRINT, SPACE, etc. to control the format of the assembler listing.

Coding of ministeps:

The operation codes for ministeps are based on mnemonics, which can be extended in many instances by one or two letters allowing the control of individual bits in the resulting ministep. Arithmetic and logic codes as well as test modes are in general not specified in the operation code but rather, more naturally, by special expressions combining the operands of the ministep.

Wherever meaningful, default operands are provided.

Branch addresses can be specified by the symbolic addressing mentioned above, by location counter references ("*") and by so-called program points allowing the introduction of local symbols which do not have to be unique. All these primitives can be combined in expressions, of course.

So far, three versions of the language specification manual have appeared and are available for the Computation Group.

Mobile Programming System

SLAC has obtained a Mobile Programming System from the University of Colorado (Poole and Waite) and has implemented it on the 360/91. Program mobility is defined as a measure of the ease with which the program can be implemented on a new machine. The mobility of a program is completely dependent upon the mobility of the programming system on which it rests. The Mobile Programming System rests on a bootstrap called SIMCMP which is a simple substitution macro processor written as a 91 statement FORTRAN program. SIMCMP accepts statements written in a language called FLUB (First Language Under Bootstrap) and produces (in the version sent to SLAC from the University of Colorado) FORTRAN statements.

The second stage, called STAGE2, is a fairly general macro processor written in 997 FLUB statements. Passing STAGE2 through SIMCMP, one obtains 1581 FORTRAN statements which are then compiled to produce a STAGE2 processor. STAGE2 allows conditional expansion, iteration, etc.

The following observations are in order in assessing the soundness of this approach:

The 1581 FORTRAN statements could not be compiled under H-FORTRAN because of the approximate 500 statement limit for single programs.

G-FORTRAN required 3 minutes, 42 seconds of CPU time on the 91, and

the resulting STAGE2 processor did not work due to the fact that the 360 hardware uses base registers for addressing memory. (When SIMCMP was modified to produce 360 Assembler instructions, STAGE2 was assembled in about 14 seconds.)

The STAGE2 language is awkward, unforgiving, and unreadable.

Computation Group Technical Memos - January 1 through June 30, 1970

88. Smedit2: A Simple-Minded Editor for Card Files
J. Ehrman, April 1970

89. IBM 360 - PDP9 Assembler Manual
M. Hu, March 1970

90. A Brief Tour Through the Micro-Programmed Research Facility
H. Saal, May 1970

91. CONC : A SNOBOL4 Program for Generating a Selective Concordance from Short Texts
C. Holbrow, June 1970

92. An Interactive Debugging Program
M.A. Fisherkeller, June 1970

93. Model for Deadlock-Free Resource Allocation - Preliminary Version - Section 1: The Basic Schedule and Linear Algorithms for-Deadlock Detection
B. Russell, June 1970

94. Section 2: Linear Algorithms for Deadlock Prevention
B. Russell, (not yet printed)

95. FCONC: A Faster Concordance Generator which Reads Input Text from Tape
C. Holbrow, July 1970

SLAC Publications

723 The Simultaneous Numerical Solution of Differential-Algebraic Equations
C.W. Gear, March 1970

742 On Direct Methods for Solving Poisson's Equations
G. Golub (et.al.), May 1970

760 Numerical Techniques in Mathematical Programming
G. Golub (et.al.), May 1970

774 Bounds for the Error of Linear Systems of Equations Using the Theory of Moments
G. Golub (et.al.), October(1969)

SLAC Reports

111 SLAC Spiral Reader Control System Reference Manual
M. Hu, January 1970

114 An APL Machine
Phil Abrams, February 1970

117 The Formulary Model for Access Control and Privacy in Computer Systems
Lance Hoffman, May 1970

RESEARCH IN PROGRAMMING LANGUAGES

In order to help find long-range solutions to the "software problem", Professors Robert Floyd and Donald Knuth are directing a series of investigations of important aspects of compiler writing and closely related subjects.

The major unknown factor in contemporary compiler design is the lack of reliable data about what users do with programming languages; it would be beneficial to know how frequently various constructions are used, in order to be able to make a rational choice between different compiler methods. It is felt that it should be possible to measure many of the critical parameters with reasonable accuracy. It should be possible to estimate, for example, how much time is spent in various kinds of lexical and syntactical analysis, as well as to determine how often various optimizable constructions occur (statistically and dynamically). This investigation involves theoretical as well as empirical considerations.

While statistics like these are being gathered, it is also likely that some general aids, by means of which the users of a language can gather similar statistics about the running of their own programs, can be developed; moreover, these aids can be extended to some new kinds of debugging tools that may have a significant impact on the debugging process.

Typical areas of research which the investigators are pursuing or planning to pursue are:

a) Development of theories relevant to code generation

b) Development of new algorithms for global optimization

c) Design of languages to facilitate real-time compiler writing (this includes "structured assembly programs" such as Wirth's PL360)

d) Design of two levels of intermediate languages: one approximately at a Polish notation level, one approximately at an "infinite register machine" level, both suitably extensible that they can adapt to a variety of source languages, and suitably clean that they can support research of classes a) and b) above.

e) Design and development of experimental programming systems.

RESEARCH IN THE ANALYSIS OF ALGORITHMS

Professors Floyd and Knuth are working on "analysis of algorithms", a field of study directed to an understanding of the behavior of particular algorithms. Two kinds of problems are usually investigated.

A. Quantitative analysis of an algorithm. In this case the goal is usually to determine the running time and/or memory space requirements of a given algorithm. The determination of running time can be done in an essentially machine-independent manner by expressing the algorithm in some machine-independent language (nor necessarily a formal language) and counting the number of times each step is executed.

B. Determination of "optimal" algorithms. In this case the goal is usually to find the "best possible" algorithm in a given class of algorithms. We set up some definition of "best possible" which reflects, as realistically as possible, the pertinent characteristics of the hardware which is to be associated with the algorithm.

The following list of reasearch topics typifies the investigations into algorithmic analysis that are in progress:

1) Study of the solution of special types of recurrence relations, both in closed form and from an asymptotic viewpoint.

2) Type A analyses of important algorithms for storage allocation, arithmetic, sorting, information retrieval, language analysis, scheduling, etc., for the many cases where published analyses are incomplete.

3) Further work on Type B analyses for several unsolved problems.

4) Extensions to programming languages and compilers intended to facilitate making empirical analyses of algorithms; and to facilitate making use of theoretical analyses by computing the running time when approximate parameters are supplied.

5) Development of a "verifying compiler" which facilitates the construction of rigorous proofs that an algorithm is correct. (A proof of validity may be considered as the first step in the analysis of an algorithm.)

The Hewlett-Packard 2116 Computer

The HP 2116 Computer has become a vital teaching tool for CS 111, Introduction to Computer Organization and Data Structures. The computer is programmed and operated by the students themselves as part of normal course activity, and is the only course in the CS curriculum in which the student has the opportunity to use a computer "hands on." Programming assignments are designed to introduce the student to several concepts that are easily developed with the HP 2116 but would otherwise be very difficult to teach. For example, the student does his own input/output programming and exercises the interrupt system of the computer in order to understand the problems of timing and control.

The computer is now used throughout the year exclusively for CS 111. Formerly, student projects other than classroom projects had been done on the computer, but these projects have been moved to other computers as the instructional demands for the computer grew.

In October 1969, Hewlett-Packard donated a card reader to the Computer Science Department that was added to the computer, and increased the capacity of the computer significantly. By spring quarter, 1970, a fast assembler for the HP computer that executes on the 360/67 campus facility became available.for student use. This assembler is used as a syntax checker and high-speed listing program to relieve some of the load on the HP 2116. As a result the computer is devoted almost entirely to console debugging of programs.

Research activities for Harold Stone

The primary direction of the research has been study of parallel computer organizations and algorithms for parallel processing. Activities have included the study of computers of the Illiac IV variety and of a processor that is known as an "orthogonal processor.'! This study brought to light the importance of a processor-to-memory interconnection scheme that is called the perfect shuffle, When a perfect shuffle is available, it is possible to do Fourier transforms, sorting, polynomial evaluation, and matrix transposition with very high efficiency.

Other areas of interest include algorithms for scheduling parallel processors, and redundancy techniques for enhancing reliability.

Reports

The Spectrum of Incorrectly Decoded Bursts for Cyclic Burst Error Codes by Harold S. Stone, STAN- CS-70- 154

Parallel Processing with the Perfect Shuffle by Harold S. Stone STAN-CS-70- 158

MATHEMATICAL PROGRAMMING LANGUAGE and RELATED OPERATIONS RESEARCH ACTIVITY.

Since 1967 work has been underway on a mathematical programming language (MPL). This work is now supported by the National Science Foundation with Professor George B. Dantzig as Principal Investigator.

1. The Need for MPL:

The purpose of RPL is to provide a language for writing mathematical algorithms, especially mathematical programming algorithms, that will be easier to write, to read, and to modify than those written in currently available languages (e.g. FORTRAN, ALGOL, PL/1, APL). It is believed that MPL's most important use (until fully implemented) will be as a communication language.

The need for a highly readable mathematically based computer language has been apparent for some time. Generally speaking, standard mathematical notation in a suitable algorithmic structure appears best-for this purpose. The reason is that most researchers are familiar with the "language" of mathematics having spent years going to school and taking many courses on this subject. For the mathematical programming application, the availability of such a tool is deemed essential.

Mathematical programming codestend to be complex. (Some commercial codes have over a hundred thousand instructions.) They are developed by persons formally trained in mathematics using, for the most part, standard matrix and set notation. Recently, research has been directed toward structured large-scale systems. These systems have great practical potential especially for planning the growth of developing nations.

To date many methods have been proposed for solving large-scale systems, hut few have been experimentally tested and compared because of the high cost and the long time it takes to program them, and because it is difficult to debug and to modify them quickly after they are written. It is believed that highly readable programs would greatly facilitate experimentation with these proposed methods and would speed up the time when they can he used for finding optimal growth patterns of developing nations and industries. Moreover, experimentation is a valuable way to develop ones intuition and test conjectures prior to developing theoretical proofs.

General features of MPL:

Research on MPL to date has been directed towards developing a highly readable language adherinq as closely as possible to standard mathematical notation. Considerable attention has been given to keeping the definition structure of MPL as general as possible.

Matrix notation is required for the mathematical programming applications and this has been given special emphasis in MPL including partitioned matrices and matrices with special structure.

Set notation is universally used in mathematical proofs. However in statements of algorithms, as found in theoretical

papers, one finds wha? appears to he set notation, but which turns out to he, on closer examination, an ordered set concept i.e. there is an assumed underlying ordering of the elements of' a se+. A convenient set-like notation is part of MPL. Typically it is used with the such-that construct which allows one to restrict or extend the definition of a set through logical expressions.

Other important features of mathematical notation are the "let" and "where" concepts. As commonly used, they serve as either a symbol substituter (macro) or as a short subroutine whose parameters are evaluated and the results substituted for the symbol. LET and WHERE constructs are also part of MPL.

Generally speaking, the literature of mathematics has been devoted to proofs of theorems. Algorithms as such, when they do appear, are often par+ of a constuctive proof and have an ad-hoc organizational structure. MPL has adopted instead the formal block stucture 0f ALGOL with minor variations. Alternatives are provided for those who prefer not to see the words BEGIN and END used as punctuation marks for blocks throughout a program. The user can optionally use less obtrusive special bracket symbols to conveniently group several statements forming a block or to group statements which follow and are subject to IF and POP clauses. It is also possible in MPL to conveniently identify by labels parentheses pairs, complex statements and algebraic expressions and thereby greatly increase readability.

In mathematics it is often desirable to change the meaning of symbols (e.g. variable names). In computer languages a formal structure for "declaring" (defining) symbols is used and also for stating the "scope" (the set of instructions) where these definitions are to he applied. For example, in ALGOL names Of variables defined within a block cannot be used outside the block without redefinition. In MPL, definition of a symbol canbemade anywhere inside the block up to its first appearance in a statement; moreover, it can also be implicitly defined by the statement itself. Implicit definition is an important feature of MPL. Provision is made for conveniently specifing the scope of A variable if it extends outside the block. Finally, it is possible to release the storage space assigned for the values of a symbol when no longer needed.

In defining a language it is natural to worry about whether or not it is possible to reasonably implement it. For example, *he present form of MPL uses linear character strings for exponents, superscripts or summations in place of two dimensional notation like: .

$$S = \sum_{i=1}^{n} a_i$$

Thus, a is written a(i). However, one of the members of our task force group (V. Nicholson) has recently completed a Ph.D. dissertation on this subject and we plan to incorporate features of his already Implemented two dimensional notation into MPL.

Except for special functions like sin(X), mathematicians avoid the use of multiple character variable names. The reason for this historically appears to be two-fold: First, it is easier to visualize algebraic manipulation of symbols when they appear as single characters. Second, it avoids possible confusionwith implicit multiplication e.g. sin(x) meaning s·i·n.(x). However, by requiring in MPL the explicit use of the multiplication symbol, multiple character names are allowed as

in most computer languages.

2. DETAILED PROJECT REVIEW

The first goal of the project was to specify the language in implementable form. The language outlined in a preliminary proposal to NSF as of May 1968 was systematically developed; the syntax was more closely aligned with standard mathematical notation and kept as general as possible. Many of the earlier constructs were extended and improved, for example:

- The vector construct was extended to include set notation in the form of ordered sets with logical qualifiers.
- Fore complex data structures were introduced, including multidimensional arrays, partitioned matrices and reference arrays.
- The domain of numeric constants was made the extended real numbers (-00,+00).
- In response to user requests, blocks were introduce4 as a primary means of defining scope of variables.
- The principle of dynamic allocation of storage was adopted for all non-scalar quantities.
- Both dynamic and static symbol substitution were introduced into the language.
- Subscripting was generalized to include subscripting of expressions.
- Function Variables which allow a general function name to be replaced by a specific name were introduced.
- Parameter passing for procedures was greatly extended by developing several different types of procedures. In particular, a function procedure was introduced which acts exactly as a function in mathematics, (i.e. without any side effects).

This draft of the MPL specifications in Backus Naur Form was prepared under the general guidance of the principle investigator by a work group at Stanford currently consisting of Stanley Eisenstat, Thomas Magnanti, Steven Maier, Michael McGrath, Vincent Nicholson and Christiane Riedl. Miss Riedl of the Stanford Linear Accelerator Center actively served as general coordinator. The other members are graduate students in Operations Research and Computer Science.

This phase of the project is nearing completion, therefor the draft is now being readied for general review by a committee probably consisting of Rudolf Bayer, Paul Davis, David Gries, Robert Floyd, Donald Knuth, and Christoph Witzgall. During the fall quarter the language will be used as a teaching tool to obtain feedback from potential users. By the end of calendar year 1970 it is hoped that the specifications can be frozen, so that implementation and use as a communication language can begin in earnest.

The second goal of the project was to implement these specifications. This involved development of a PL/1 translator and made possible evaluation of the language by Operations Research graduate students and researchers from both the academic and industrial communities.

A preliminary test version of MPL, referred to as "MPL-McGrath", was at the suggestion of Paul Davis implemented in 1969 by Michael McGrath using PL/1 as a translator into PL/1 instructions. This version included those features of MPL that

were easiest to translate into available PL/1 constructs. PL/1 was used in order to produce a somewhat environment free system. It was felt that this would provide the widest possible circulation for HPL, since any installation with a PL/1 compiler could then be used.

The current version of the MPL/PL1 translator encompasses many of the unique constructs available within MPL. The translator was successfully used in a large scale systems optimization seminar with enthusiastic student response. Much valuable information was obtained from this exchange, and it is hoped that this practice can be continued. Of particular note, is that many students found the language easier to use and less tricky than either FORTRAN or ALGOL.

The language was presented to the industrial community through the Stanford "Computer Forum" by M. McGrath in 1969 and C. Riedl in 1970; to the academic community through lectures by G. Dantzig; and to the professional community by R. Bayer and C. Witzgall in talks on their matrix calculus which is expected to play a role in the generation and manipulation of special matrix structures. Some work was also done on using MPL as a tool in developing new algorithms and in presenting some of the existing algorithms in the field of Operations Research. It is hoped that this will become one area of future concentration in the further development of the MPL language. This has particular importance in gaining wider acceptance for the language.

3. RESEARCH PROGRAM

3.1. MPL as a Communication and Programming Language

To date the primary objective of the MPL project has been the formal language definition. The result of this effort is the Language Specification Manual written in Backus Naur Form which should function as a basis for an implementation. Since this manual is intended for computer specialists, it is not very suitable for an applied mathematician not trained in computer science. Accordingly, a next step for the project (and its proposed continuation) is the development of an MPL user's manual. This document would serve in two capacities:

(i) by giving an introduction to MPL for a wide spectrum of possible users, and

(ii) by expanding and interpreting the more involved features of the language found in the MPL specification manual.

To accommodate both of these objectives, the user's manual would endeavor to present MPL in a simplified form and at a level in which most of its constructs are explained. In this manner, the reader at the beginning or intermediate level, knowing only a subset of the language, would nevertheless be able to write MPL programs compatible with the full language.

With a user's manual available, the project would proceed into a testing and evaluation phase. An important contribution to this phase would be feedback from potential users. From this feedback we would be able to ascertain what modifications, if any, are required to give us the "best" language for the user. It is probable that HPL will be equally useful for statistical and numerical analysis applications, particularly in conjunction with special subroutines useful in these fields. Though we would promote investigation into a wide range of areas, a thorough study of such a scope would be impractical. Instead, we propose to concentrate

upon applications to Mathematical Programming.

Testing MPL as a Language for Mathematical Programming would proceed along two fronts. First, standard algorithms, such as Generalize? Upper Rounding, would be programmed using MPL. This would allow us not only to evaluate MPL as a programming tool but also to assemble a library of algorithms for use in further research. Second, MPL would he used to write andtest new algorithms, consequently, &valuating its potential as a research tool. We hel ieve that the language could have a qreat impact in this area - especially in academic research where the time and expense in programming for large scale systems has been prohihitive in other languages.

As a user's tool, MPLhas been developed to parallel much of standard mathematical notation. Thus most algorithms written in mathematics could almost as easily be written and read in MPL. This aspect of the language makes it attractive as a standard communication language for algorithms. As one furtherphase of this project, we hope to explore this fact in greater depth. In particular, we would investigate whether it would he plausible to use MPL as a standard vehicle for presenting algorithms in journals especially for the newly proposedMathematical Programming Journal. Not. only would this have the beneficial effect of standardization, but it would also mean that published algorithms could be easily tested or implemented via PPL.

Some ofthe objectives outlined above can be partially met with the current version of the MPL translator. In order to fully test the language and implement. it as a user% tool, however, the translator will have to he expanded or a compiler written. An investigation of these possibilities constitutes the next major task of the continuing project.

B. Implementation Considerations

A complete, "machine-independent" implementation seems essent ial in gaining broad acceptance of MPL as a mathematical programming language. Such an implementation could take two directions:

(i) Extending the current translator to encompass those MPL concepts not presently handled (e.g. subscripting as an operator, partitioned data structures, concatenation, and set generators).

(ii) Writing a full-scale compiler into some ideal machine language (e.g. three address code or reverse Polish).

The translator would be less work but the more efficient rode produced by a compiler would makethe solution of large scale problems more practical. However, of equal, if not. greater, importance is a "How to Implement" manual, a compendium of suggestions on implementingsome of the more powerful MPL constructs as well as techniques for handling large scale data structures and codes involving many thousands of instructions on a computer.

For the most part, the techniques would be machine independent, i.e., the method of implementation outlined in the manual should he of help in implementing any large-scale mathematical pragraraing system.

Part of the manual would be concerned with the analysis of an MPL program. Ttems included would be parsing techniques, symbol table organization, a precedence grammar if possible, suggestions for the internal represent.ation of the program after analysis, and an outline of code emitted for advanced

features of MPL (e.g. function variables, indexing sets, dynamic LET statements).

Runtime organization which is essentially MPL independent would require a study of data structures necessary for large scale systems, dope vectors, algorithms for handling the non-first-in-first-out data structures of MPL.

If an easily modifiable translator were written, experiments could be made with different runtime data structures, data handling algorithms, and computational algorithms (such as matrix expression evaluation).

OPERATIONS RESEARCH ACTIVITY

Most of Professor Dantzig's Operations Research work is conducted in the Operations Research Department.

1. Professor Dantzig is organizing a NATO conference on "Applications of Optimization Methods for Large-Scale Resource Allocation Problems", to be held in June 1971.

2. Professor Dantzig is the principal investigator of the following projects:

Optimization Theory
Sponsored by the National Science Foundation

Mathematical Programming Language [MPL]
Sponsored by the National Science Foundation

Stochastic Mathematical Programs
Sponsored by the Atomic Energy Commission

Mathematical Models in Operations Research and Computer Science
Sponsored by the Office of Naval Research

Research in Mathematical Biology
Sponsored by the National Institutes of Health

Professors Richard W. Cottle and Alan S. Manne of the Operations Research Department are associated with Professor George Dantzig in the Mathematical Methods in Operations Research and Computer Science project. Professor Robert Wilson of the Graduate School of Business is associated with him on the Stochastic Mathematical Programs research project.

RESEARCH REPORTS

CS 119	Mathematical Programming Language	Bayer/Bigelow/Dantzig/ Gries/McGrath/Pinsky/ Schuck/Witzgall
CS 126	Complementary Spanning Trees	Dantzig
OR 66-2	All Shortest Routes from a Fixed Origin in a Graph	Dantzig/Blattner/Rao
OR 66-3	All Shortest Routes in a Graph	Dantzig

17 July 1970

DIGITAL SYSTEMS LABORATORY

SUMMARY OF RESEARCH 1969-70

Prof. E. J. McCluskey, Director

The research of the Digital Systems Laboratory has been chosen to provide a balance between topics concerned with computer software and hardware topics. In fact, there has been a trend towards research areas which integrate both programming and logic organization. Some topics where this integration has-been possible are:

(1) The study of the control of processes operating in parallel. Here a representation has been developed from which it is possible to systematically design either a logic circuit or a program realization of the control algorithm (component of an operating system). Problems such as the mutual exclusion problem and the buffer problem are treated as examples.

(2) The study of parallel implementation of a single-assignment language in which a computer organization for maximum parallelism and a suitable programming language for describing parallel algorithms were developed concurrently.

(3) The development of a system consisting of a general purpose computer with a specially designed digital differential analyzer as a peripheral device.

-Another aspect of the DSL research arises from the fact that two of the most critical problem areas of contemporary computer systems are those of reliability and operating system design. Much of the research of the Digital Systems Laboratory is related to one of both of these. A major

effort is underway aimed at developing a theory of the effects of component failures on digital systems. This theory has already led to insights' on new testing algorithms and on design of networks so that they 'are easily maintained. Work on control of parallel processes and on computer system evaluation represent approaches being taken to improveoperating system design.

From another point of view the attempt to study parallelism in digital systems is common to many of the DSL activities. This ranges from work on large combinational networks for tasks such as sorting to studies on parallel computation.

EJMc: sb(7-17-70)

Project: 3208

Contract: NASA, NGR-05-020-014

Principal Investigator: A. M. Peterson

Staff: R. Koralek, E. Schulz, and B. Parasuraman

Title: NONLINEAR CODING THEORY

Work has been progressing on developing a systematic theory of nonlinear error-correcting codes. These codes, used to transmit data over noisy channels (e.g., a spacecraft data link), will automatically facilitate correction of transmission errors. Present codes use linear parity check bits to provide error-correcting capability; Such linear codes are, fairly easy to encode but rather difficult to decode. Nonlinear codes should be more easily decoded than present codes, Also, nonlinear codes can be more efficient than linear codes, For example, the best known g-bit distance-3 (single-error correcting) linear code contains 32 words; the corresponding nonlinear code has 38 words. For longer block lengths, the difference becomes even greater: the best linear 15-bit distance-5 code contains 128 words, but a similar nonlinear code has 256 words. For a channel like a spacecraft data link, this improvement results in a higher bit rate for the same probability of an uncorrected error.

Research in the past few months has concentrated on the weight and distance properties of maximal code sets (sets of code words with the maximum number of words for a given word length and minimum distance). An upper bound has been found on the minimum distance of a codeset of given parameters, and this calculation can be extended to give the average distance. A theorem shows that the distance spectrum of a maximal code set is an exact multiple of the weight spectrum; thus the average weight is known. A study of "weight moments about the average"

Puts strong constraints on the numb& of words of each weight that a maximal codc can have. Two other theorems, concerning the number of words of a given weight that can be distance d apart, specify exactly the weight structure of a class of maximal nonlinear codes, It is then an easy matter to write down the code words.

In the next few months a formal, systematic way to construct these and other nonlinear codes will be sought. Encoding and decoding methods will be investigated, in terms of both algorithms and. hardware. Preliminary results indicate that coding and decoding should be very fast, especially using certain medium-scale integrated circuits and unorthodox logic organizations such as cellular nets. Thus nonlinear codes will be able to increase data rates not only by coding efficiency but also by faster decoding hardware,

Title: MATCHING A DIGITAL DIFFERENTIAL ANALYZER TO A GENERAL PURPOSE COMPUTER

This study is aimed at developing configurations for a special-purpose hybrid computer which is intended primarily for the solution of all kinds of differential equations. The composite machine would consist of an electronic digital differential analyzer (d.d.a.) and a general purpose digital computer. With *such a combination, the extremely high solution speed of the d.d.a. can be used effectively with the memory and logic-capabilities of the digital computer. .

In contrast to a block of successive instructions in a computer program, the d.d.a. uses a set of patching or interconnection instructions. By implementing these interconnections with digital hardware, communication between the d.d.a. and the computer can be simplified. Such hardware implementations are now economical because of the abundance of fast and compact semiconductor circuits.

The first phase involves putting together a system of integrators using available MSI circuits and connecting them to a suitable mini-computer. The hp 2116 Bhas been selected on account of its availability, convenient size and adequate instruction repertoire. Since MSI circuits arc generally available in multiples of 8 bits, the 16 bit word length of the hp 2116 B is well suited to this application. The d.d.a. organization centers around two banks of semiconductor random-access memories and a high-speed parallel arithmetic unit. Each word in the memory bank represents one integrator, and since only one word can be accessed at any time, the d.d.a. is organized serially. As a consequence, the solution time increases with the number of integrators used.

When connected to the computer, the d.d.a. passes information via the I/O- channels_, and it therefore appears as an I/O device to the computer. The computer performs the following functions:'

1. Setting up initial conditions
2. Inserting scale factors
3. Reading in program interconnections
4. Initiating the compute cycle
5. Accessing any integrator to transfer data
6. Reversing computations if necessary
7. Interrupting to transfer control to some other device
8. Doing "table look-ups" to check known solutions
9. Any other general housekeeping functions,

On the other hand the d.d.a. would be generating rapid solutions to differential equations, and these numerical solutions would be stored in the memory units,

The next phase of the project deals with extending this concept to larger computers and developing more general rules for d.d.a. interfacing. Specifically, some kind of assembly language program needs to be developed to make the composite machine more user-oriented, In this way the d.d.a. can be made completely automatic without the past problems of patchboard wiring and complicated programming.

Another aspect being currently investigated is the possibility of realizing d.d.a. integrators in the form of cellular arrays. These arrays lend themselves easily to large scale integration. Other researchers have demonstrated these techniques successfully in the case of logic networks.

Project: 6902

Contract: Tri-Services N00014-67-A-012-0044

Principal Investigator: E. S. Davidson

Staff: T.F. Chang, H.P. Lee, and S. S. Reddi

Title: ALGORITHMIC DESIGN OF COMPUTING SYSTEMS

A. Computer System Evaluation

In computer systems there is a continuing trend toward integrated circuit components of lower cost and higher density and speed. This trend has been evident in recent systems announced by computer manufacturers and is reflected in higher performance machines as well as in larger machines with more complicated internal structure and system complcxity,and in networks of small and large machines with distributed memory and processing capability. For a user with a well characterized job mix, a wide variety of systems is available and within each system many configurations and operating strategies are possible. The user has available the tools of simulation and measurement to aid him in making this decision, and in modifying that decision as the job mix changes in the course of time. The computer designer has the same tools at his disposal, but his freedom to specify the system is much broader. Thus, while the user must select his system by choosing between several available parameter values for ccrtain'spccified system parameters, the designer must investigate parameter values of a broader range and finer degree, as well as many additional system parameters.

Both simulation and measurement have developed without a cohesive, analytically-oriented system science for computers. The selection of the system parameters for a simulation or measurement study, as well as the appropriate level of detail for the study, depends upon the intuition of the designer of the study. The consequences of the lack of an analytical model are that parameters which are functionally related are often measured or simulated as independent parameters causing a computational inefficiency in the study; that the level of detail may be inappropriately selected causing inefficiency if the selected level is too low and causing the omission of essential effects if the level is too high; and that results of such studies cannot be rigorously applied toward system improvement, except by studying all cases and simply selecting the one which appears best for the sample operating environment selected. The goal of of this study is the development of such an analytic approach which can be applied independently or in conjunction with simulation and measurement to solve these problems.

Preliminary investigation is centering on several simulation and measurementstudies. A simulation study which yields the address referencing behavior of some kernel program has investigated the probability of a consecutive string of references to the same memory bank. Probabilities were reported for four numbers of memory banks and five string lengths. It was found that these twenty independently measured probabilities could be closely predicted by a model involving ten parameters. A least-square fit was used to evaluate the ten parameters in terms of the simulation-produced probabilities. Further research is directed toward improving the model, treating address referencing behavior in general, and evaluating the model pramctcrs directly

from an assembly language program.

Future research will be concerned with replacement of an address reference list, such as used in paging studies, with a suitable parameterization which can be evaluated from an assembly language or higher level program. Additional research will be concerned with predicting specific system improvement based on results of measurement or simulation studies.

B. A Model for Pipeline Systems

Recent computer architecture is tending more in the direction of pipeline systems to overcome speed -limitations in arithmetic computations and memory accessing. The pipeline system can be described as one in which the initiation of an operation is begun before the previous operation has been completed. A model has been developed for a pipeline function generator which operates in a pipeline fashion, but is restricted to performing a single type of calculation, e.g., addition, matrix multiplication, or effective address computation.

The model divides the pipe into temporal segments. During each segment a piece of the computation is performed. The hardware involved in computation is divided into physical segments, each of which consists of the logic necessary to perform Its computation and a buffer required to store partial results. The architecture of the pipe is then specified by a reservation table which allocates the temporal segments of the pipe to the physical segments. Such a reservation table allows the re-using of physical segments at two different points in time in the pipe. This. capability is a generalization of what is generally found in existing pipeline function generators.

A characteristic vector may be computed from the reservation table. This vector is used to avoid "collision" of operands, i.e., attempted usage of the same physical segment at the same time by sets of operands at different temporal segments in the pipe. The characteristic vector;may be stored in a shift register which is shifted to the left one position in each temporal segment. Collision is avoided 'by initiating new operands only when a zero is found in the left-most position of the shift register. Each time a new operand is initiated the characteristic vector is ORed into the shift register,

Scheduling of the pipe may thus be accomplished by initiating new operands in the pipe, after their arrival in an input queue, at the first appearance of a zero in the left-most pbsition of the shift register. While this is a very simple strategy to follow and requires very little hardware for implementation, it is not in general an optimum scheduling strategy. That is to say, for certain pipeline architectures and certain arrival times of operands in the input queue, a scheduling strategy with higher throughput can be found. Investigation of optimum scheduling strategies is carried out by generating a state diagram for the states of the shift register. The minimum weighted self-loop in the state diagram produces the maximum steady-state throughput of the pipe, i.e., the minimum constant time interval between initiation of operands which can be sustained indefinitely. Furthermore, a computationally efficient algorithm has been found for determining the maximum throughput cycle in the state diagram which corresponds to initiating operands at non-constant time intervals in an indefinitly repeatable cycle.

Further research is directed toward deriving cost-efficiency considerations for alternative pipeline architectures performing the same computation. Research is also directed toward establishing the efficiency of a pipeline architecture by analysis of fundamental cycles in the characteristic vector. Adaptation of the pipeline function generator model to general pipeline(function generator model to general pipeline)processing will be considered.

C. Combinatorial Problems

This research has been directed toward solving some problems of a combinatorial nature in switching theory. The first problem concerns "maximum sized" equivalence classes induced by a group acting on a set of objects. Golomb [1] gives methods for determining these equivalence classes and most of his methods assume that the group in question is cyclic. A method is found which is applicable to completely solvable groups. This method naturally suggests a way of enumerating "maximum sized" equivalence classes of switching functions of up to 4 variables.

Another, yet unsolved, problem investigated is to enumerate Hamiltonian circuits on an n-cube for $n > 4$. Gilbert[2] enumerated these circuits by an exhaustive search on a computer up to $n = 4$. A method for generation of Hamiltonian circuits by choosing certain faces on an n-cube is found (which is similar to the method of Hermary). This method consists of defining two graphs called the "face adjacency graph" and the "face tangency graph". The set of faces to be chosen for a Hamiltonian circuit is characterized in terms of these graphs. This method can be easily adapted by using a computer to generate Hamiltonian circuits.

Certain arithmetic properties of the number of Hamiltonian circuits on an n-cube are also investigated. It is found, for example, that if H_n is the number of Hamiltonian circuits on an n-cube then

$$H_n \equiv 0 \pmod{n} \text{ if } n \text{ is odd}$$

$$H_n \equiv 0 \pmod{n/2} \text{ if } n \text{ is even}$$

The problem of classification of Hamiltonian circuits into equivalence classes is also studied. The method mentioned above for generation of Hamiltonian circuits is particularly suitable for algebraic manipulation of equivalence classes. Hamiltonian circuits are required in many engineering applications for example the discovery and evaluation of unit-distance or Gray codes.

Another problem, posed in the SIAM Review [3], which has been solved is the enumeration of the number of ways in which n identical balls can be distributed in h boxes in a row such that each pair of adjacent bdxes contains at least 4 balls. A recursion formula is developed for the above enumeration.

References:

(1) Golomb, S. W., A Mathematical Theory of Discrete. Classification, Proc . Fourth London Symp . Inform. Theory, Butterworth and Co. (Publishers) Ltd., London, 1961.

(2) Gilbert, E.N., Gray Codes and Paths on the n-cube, Bell System Tech. J., Vol. 37, No. 3, 1958

(3) SIAM Review, Vol.. 10, No.4, October 1968, pp. 451, Prob. 68-16.

Project: 6903

Contract: NSF GK-4852

Principal Investigator: E. S. Davidson

Staff: H. P. Lee

Title: DESIGN OF NAND NETWORKS

An interactive algorithmic approach to the design of NAND networks by transformson the interconnection topology of a given network is being developed. With the advent of LSI technology, the need for optimum solutions has diminished. Furthermore, historically justified definitions of optimality no longer apply. In order to provide a realistic goal, a wide variety of conventional cost criteria such as gates and gate inputs, and a wide variety of network contraints such as fan-in, fan-out, levels of logic, and interconnection cross-overs must be compatible with the approach taken. Most existing algorithms fall short of this goal and all require more computation time than justifiable in their pursuit of an optimum network. One approach which overcomes these difficulties is to generate efficiently a starting, possibly non-optimal network which is presented to the designer for his approval. The designer then uses an interactive program for solution improvement. He can then identify the areas of the network which do no meet his cost or contraint criteria. Violations of certain specified constraints can easily be flagged for his attention by the interactive program.

A single transform which operates on the interconnection topology, of the network has been found. The output connecting a designated gate to the network is deleted and is connected instead to a number of other gates in the network. The entire transform may be specified by designating a "transformed gate" and a "modified gate". The new connections are made and the resulting network is then simplified logically.

Since the transform manipulates the interconnection topology of the network directly, it is rather easy for the designer to deal directly with network constraints. For example, he may remove an input from a gate which exceeds fan-in limitations. He may reduce the number of levels in a subnetwork by creating an internal invertcr in that network. This is done by removing all inputs but one from a gate. The internal invcrter is then removed by the automatic network simplification following the last transform. Cross-overs may be eliminated by transforming a gate to another gate whose subnetwork shares inputs with the transformed gate. Transforms which might add gates or gate inputs to the network can readily be identified. 'Transforms which reduce the number of gates or gate inputs can usually only be identified after the network simplification following a test execution of the transform. The desirability of executing test transforms, and the possibility of curing one undesirable feature of a network by creating another, implies that a file of solutions be kept in storage as they are generated, allowing the designer to return to any previous solution and apply transforms upon it.

The power of the transform is vindicated by the fact that many previously known simplification heuristics may be viewed as special cases of the transform. The transform has been rigorously defined and its validity has been proved.

Project: 7101

Contract: Tri-Services, Contract N00014-67-A-0112-0044 ,

Principal Investigator: E.J. McCluskey

Staff: D. Siewiorck .and R. Betancourt

Title: RELIABILITY THROUGH REDUNDANCY.AND FAULT DETECTION

A. Redundancy Techniques

Protective redundancy schemes (systems which can tolerate faults because of additional components, programs, or time used for the computational task) can be classified into two general categories: massive redundancy and selective redundancy. In massive redundancy the effect of the faulty element is masked instantaneously by permanently connected and concurrently operating replicas of the faulty component. Selective redundancy encompasses the remaining protective redundancy techniques, i.e., the presence of a fault must be detected and then corrected (by error correcting codes, switching of stand-by spares, or system reconfiguration).

The overall goal of this research is to develop tools a designer can apply to evaluate' the various fault-tolerant techniques, that might be incorporated into a system.

-In the area of massive redundancy the method most commonly considered is triple modular redundancy (TMR), whereby a nonredundant network is divided into modules which are triplicated and separated by majority gates (voters). A method for rapidly estimating the

reliability of a TMR version of a cascade network has been dcvclopcd. The trade-offs between the number of modules or the use of single versus triplicated voter stages are easily evaluated. For the more general network (for which the cascade formulas have been shown to be significantly inaccurate, thereby warranting another approach) a cellular method has been formulated which yields an upper bound on reliability. This procedure is more accurate than the generalized cascade approach and involves much less calculation (cell reliabilities are derived by a simple combinational formula) than the more accurate methods. In the initial stages of an iterative design-procedure the loss in accuracy induced by this method is considered of secondary importance (especially when module reliability is not yet accurately known) with respect to speed of calculation.

A mathematical model for the diagnostic procedure used in selective redundancy (stand-by) is being developed. Once completed the model will be directly applicable to nonredundant new machine testing and diagnosis. Several parallelsbetween the multi-processor scheduling problem and the multiprocessor diagnostic sequence have been drawn and work is in progress on the solution to the latter problem.- Also a model and some theorems concerning the problem of "checking the checker " as applied to a system's hard core (that portion of a system considered to be fault free and containing the minimum subsystems required to initiate self-diagnosis) 'are being developed.

B. Detection of Faults in Logical Circuits.

This project is concerned with finding test procedures to detect faults in non-redundant combinational digital circuits, with studying the effects that faults have in the behaviour of the circuits, and with finding guidelines to help in the design of switching circuits in order to make them more easily testable (i.e., to reduce the complexity and length of the testing procedure).

The kind of faults considered comprise the so-called "stuck-at-0" and "stuck-at-1" faults, which are of a more or less permanent nature (the character or existence of a failure does not change during the testing procedure). Their logical effect in the circuit is to tie one or more lines to a logical 0 or to a logical 1, irrespective of the input signals applied to the network'.

We have proved that, for a given unate function, (functions that can be represented by a normal form in which no variable appears both complemented and uncomplcmented), there exist sets S_0 and S_1 of input combinations that detect, respectively, a stuck-at-0 or stuck-at-1 fault in any line in any irredundant rcalizat ion built with AND

and OR gates (or with appropriate modifications, built with NAND or NOR gates). These disjoint sets are minimum for the class of realizations of the given function in the sense that there are no smaller sets that are sufficient to test completely all the different realizations. In particular, a two level AND-OR (OR-AND) network needs all the S_0 (S_1) tests.

The sets S_0 and S_1 are easily obtained from the minimal sum and minimal product forms for a function. We have proved that a fault that can be detected by a test not belonging to S_0 or S_1 can be detected by a test in S_0 or S_1, and so we can restrict ourselves to study only such sets.

A multilevel realization may or may not need all the S_0 and S_1 tests. To find in this case a minimum set of tests, we refer the multilevel circuit to a 2-level formula and do a process of minimization by the well known covering method. The advantage of this procedure over others previously used is that we perform the minimization using only the S_0 and S_1 sets instead of all possible tests.

Extending slightly the method used for unate functions, we can find also minimal sets for non-unate functions that have only one minimal sum and one minimal product (the essential prime subcubes cover the entire function). We have not been able to extend effectively the same procedure for functions that do not meet the above requisite. Our research efforts are directed toward the solution of this problem.

Knowing the tests for a given member of a class of functions, it is usually very 'easy to find tests for the whole class. The Classes we have studied are: Complementing Symmetry Class, Duality, Permutation of Variables, and Partial Symmetry.

Project: 7102

Contract: NSF CJ-165

Principal Investigator: E.J. McCluskey

Staff: F.W. Clegg, K.Y. Mei, and D.J. Chesarek

Title: EFFECTS OF FAILURES IN LOGIC NETWORKS .

A. Algebraic Properties of Faults in Logic Networks

A general study of the effects of so-called "stuck-at" faults on the structural and functional characteristics of combinational logic networks has been undertaken. It has been shown that some of the possible faults which can occur in a given network bear relations to certain other possible faults in that network. Knowledge of these relations greatly facilitates consideration of networks in the presence of failures.

The two types of relations which were considered are those of covering and equivalence. The covering relations-introduced reflect the mechanisms whereby the presence of certain faults in a network renders the occurrence of other failures to some extent unobservable. The equivalence relations which were studied reflect the varying degrees to which distinct faults in a network can be indistinguishable.

Any equivalence relation partitions the set on which it is defined into disjoint equivalence classes. The equivalence relations introduced between the faults which can occur in a network thus permit one, for the first time, to treat these faults not individually, but in classes.

This greatly simplifies work involving faulty networks since the number of different faults which can occur in a given circuit is typically many times greater than the number of fault equivalence classes,

Several different fault equivalence and covering relations have been considered. Some are defined in terms of the effects of faults on network structure--others in terms of the effects of faults on the network output function.

A modelling technique has been developed whereby the structure of a given network is represented by a labelled, directed graph. The effects of faults on this structure are modelled by appropriate transformations applied to this graph. These models and the associated algebraic techniques which are developed provide a particularly convenient means of characterizing the relations between, and other aspects of, the faults which can occur in a network. Key theorems have been proved which establish necessary and sufficient conditions for the existence of the various covering and equivalence relations which permit one to determine these relations directly from a systematic inspection of the network under study. Other results demonstrate how one may infer certain properties of the faults in a network and the relations between them directly from certain characteristics of the network's structure (such as the existence of reconvergent fanout paths) and of the function it implements (such as symmetry). Such inference has been greatly facilitated by the introduction of a new means of characterizing both network structure and output function in a single algebraic expression. Enumeration techniques have also been developed which permit one Co quickly establish upper bounds on, and in

some cases exact counts of, the number of classes into which the faults of a network are partitioned under the various equivalence relations.

The application on the knowledge obtained by these techniques of the properties of, and relations between, faults have been explored. In particular, it has been shown that knowledge of the relations between faults has important and immediate usefulness in the area of failure detection and diagnosis.

B. Fault Dominance in Combinational Circuits

Fault detection and diagnosis of logical circuits has become more important, due to the increasing demand for higher reliability and the additional complexity of computers. Many testing and diagnosing methods have been devised to deal both with circuits in general and with specific machines. The theory of faulty circuits is still being developed.

Clegg [1] and Shertz [2] have both studied the equivalence relations of faults in combinational circuits. These studies were motivated by the fact that equivalent faults are indistinguishable; thus the entire equivalence class may be treated as a simple fault, drastically reducing the complexity of fault testing,

Fault dominance assumes the fact that detection of some faults will automatically lead to detecting some others. Only dominated faults need to be considered in fault detection.

Important theorems developed to date [3] concerning simple fault dominance include:

(1) The dominance relationship holds between input and output , faults of an irredundant circuit if, and only if, the function is unate in that input variable.

(2) Input faults can either be equivalent or show no dominance between two faults if the function is partially symmetric in the two variables of interest.

Due to the transitivity of dominance, we also have:

(3) If a fanout-free logic circuit is realized by symmetric, unate gates, tests designed for detecting all stuck-at faults associated with first-level input leads can also detect all other simple faults in the circuit.

Shertz's [2] graphical display of fault relations is extended to cover the dominance relation as well. This extended fault diagram serves as a visual aid to disclose fault relations.

It has been found that the direction of dominance at the fanout point is the reversal of that at the gates; therefore, theorem 3 does not hold for circuits with fanout.

An algorithm is being developed to select fault leaders which are responsible for multiple faults. With fanout-free circuits realized by AND, OR, NAND, NOR, NOT gates, we can prove that any multiple fault must dominate at least one of its component single-fault leaders. Therefore, detection of all single faults of such circuits will detect all multiple faults as well.

The inhibition of detection due to fanouts is under investigation. -Dominance in multiple output networks is expected to be a generalization of former results.

REFERENCES

[1] Clegg, F. W. and E. J. McCluskey, "Algebraic Properties of Faults in Logic Networks," Technical Report no. 4, SU-SEL 69-078, Digital Systems Laboratory, Stanford University, Stanford, California, 1970.

[2] Shertz, D. 'R., "On the Representation of Digital Faults," R-418, Coordinated Sciences Laboratory, University of *Illinois, Urbana, Illinois, 1969.

[3] Mei, K. C. Y., "Fault Dominance in Combinational Circuits," (to be published as a Technical Report-of the Digital Systems Laboratory, Stanford University).

C. Fault Detecting Experiments for Sequential Machines

This research is directed toward the development of an algorithmic procedure for the generation of test sequences for digital logic circuits which contain clocked memory elements.

The specific algorithm under investigation assumes only a knowledge of the logical behavior of a circuit and is therefore independent of the specific hardware used to implement the circuit. A sequential machine model of the Mealy type is used to describe the logical behavior of the circuit for which the test sequence is to be generated. No assumptions are made about the number of failures which may exist in the circuit being tested. The types of failures are restricted to those which do not increase the number of internal states of the original machine. This is important from the theoretical standpoint but may not be significant in practice since it is known that randomly generated test sequences do a fairly good job of testing for failures.

The rationale behind the algorithm under investigation comes from the use of special synchronizing symbols which are commonly used in several areas of computer design. The algorithm uses a particular input/output symbol pair as a synchronizing symbol to delimit test subsequences and to reinitialize the internal state of the machine being tested.

Bounds on the minimum and maximum test lengths for the algorithm have been derived. The upper bound on the test length

compares favorably with the other algorithms which solve the same problem. In certain cases the new algorithm results in a very significant saving in both test length and in the time required to generate the test.

The current investigation centers on the characterization of those machines for which the test is guaranteed to detect all errors. To date it has not been possible to construct a failed machine which will pass the test generated by the algorithm.

Project: 7111

Contract: NASA, NGR-05-020-337

Principal Investigator: E.J. McCluskey

Staff: T.H. Bredt

Title: PARALLEL COMPUTING

A parallel computer system is a complex combination of circuits and programs known as "hardware" and "software". These systems are parallel in the sense that different components in a system are operated or executed at any instant. One example of a part of a computer system in which parallelism is very important is the operating system, that part of the system responsible for the processing of programs submitted by users of the computer. The operating system performs functions such as the scheduling of jobs to be run, the allocation of resources (such as memory space, input devices, and output devices), the definition and maintenance of files of information, recovery from errors which occur in user programs, and the determination of what it. costs to use the computer. At the present time, computer systems,-(in particular, opera-ting systems) are designed in very ad hoc ways. Not only are there few, if any, tools which can be used in the initial construction of the system, but once the system is built, there is no way, other than by actually running the system, to determine if it operates correctly. This situation is very wasteful because the systems may operate inefficiently, because tremendous amounts of time must be spent in testing or, so-called, "debugging," and because it is difficult to convey to other programmers a description of how the system is constructed.

There are then fundamental difficulties involved in the design of computer systems. The major goal of our research is to develop a formal model for the study of parallel computer systems, and in particular, operating systems. This formal model will be used in the initial design of the system and in the analysis of existing systems to verify that they operate as intended. In addition, the model will provide a concise and unambiguous definition which can be used in describing the system.

Our work in this area has been underway for almost two years and considerable progress has been made. Initial attention was focused on a particular problem which occurs in the design of parallel computer systems: the mutual exclusion or interlock problem. In this problem, components in a system, which are operated concurrently, must be controlled so that it is impossible for them to perform some prespecified operations simultaneously. It must also be guaranteed that if a component wants 'to perform its special operation, it is eventually allowed to do so. Our model is based on the use of flow tables to describe the operation of programs and circuits. The use of these tables in the design of switching circuits is well known but their use in the design of programs is new. The model makes it possible to design programs for solving the mutual exclusion problem. It also makes it possible to analyze the efficiency or complexity of the-control mechanism used to prevent operations from being performed simultaneously, to establish minimal requirements for all correct solutions to the problem, and to give solutions which use these minimal requirements. Program and circuit implementations can be

considered simultaneously, making it possible to consider in a precise way the relative efficiencies of hardware and software implementations.

In the immediate future, we plan to study the design of other portions of the computer system including the specification of interrupt facilities and job schedulers. In addition, the design of a complete, although perhaps modest, operating system will be attempted with the hope that it will be possible to establish the logical correctness of the design without need for costly testing and debugging as is presently required.

We believe the consideration of system design problems such as have been mentioned is very important. Such research is seldom conducted outside the university because the time pressures found in industry, which require the development of a working system within some fixed and usually short period of time, make it necessary to consider only the design of a particular system and make the investigation of tools which could be useful in the design of all future systems a luxury that is nearly always not considered.

Project: 7151

Contract: Tri-Services, N00014-67-A-0112-0044 .

Principal Investigators: T. H. Bredt/E. J. McCluskey

Staff: D. Chamberlin, T. G. Price, S. H. Fuller and J. M. Cadiou

Title: PARALLELISM IN COMPUTING SYSTEMS

A. Parallel Implementation of a Single-Assignment Language

The purpose of this project is to investigate a particular means of parallel computation, i.e., a way in which many computing resources can be focused simultaneously on a single program in order to speed up its execution. The parallel system under investigation is distinguished from other parallel systems by the following properites:

1. Programs are originally given to the system in a high-level --language, but the programmer need not explicitly indicate to the system the opportunities for parallelism in his program.
2. Parallelism is discovered and utilized on the statement level or even within a statement, rather than on the level of blocks of statements.
3. The system contains many processors which operate independently and asynchronously oh a program residing in a common memory.

The approach to parallel processing used in this project was inspired by L. G. Tesler's proposal of a class of single-assignment languages[1]. A single-assignment language requires that each variable be assigned a value only once during the execution of a program. This property enables

the sequencing of statements in a program to be implicitly determined by their data flow. As soon as all the input variables of a statement are defined, that statement can be executed; during execution it may, in turn, define another variable which will release other statements for execution. If many statements have their input variables defined at the same time, they may be executed simultaneously.

The first task of this project was development of a high-level single-assignment language, called SAMPLE for Single-Assignment-Mathematical-Programming-Language. SAMPLE was designed for numeric applications, and it incorporated most of the features of Algol, including block structure and recursive procedures. The language proved to be particularly well adapted to matrix manipulation algorithms because of the high degree of potential parallelism in the algorithms.

The next step was development of a machine-level language which could be directly executed by hardware, and into which SAMPLE could be translated. This was accomplished along with some suggestions about how the translation process itself might be treated as a parallel process, although this was not the principal focus of the project.

The next step was the description of specific hardware modules to execute the translated programs, taking full advantage of their potential parallelism. The system contains three passive memory devices, each organized into several independent banks. The types of access requests which each memory device can service are described in detail. The system also contains a variable number of processors whose behavior is described in detail on a memory-cycle level.

The final step of the project, which is still underway, is evaluation of the proposed system. For this purpose, a simulator program has been written to investigate the behavior of the system. By executing SAMPLE programs on the simulator, it is hoped that data can be gathered on the dependence of execution time on the number of available processors, and other parameters of the system.

REFERENCE

[1] Tesler, L. G. and H. J. Enea, "A Language Design for Concurrent Processes," Proc. of the Spring Joint Computer Conference, 403-408, 1968.

B. Scheduling and Resource Allocation in Computer Operating Systems

This research is largely concerned with computer operating systems and their effect upon overall system performance. There is a need to develop better tools for locating system "bottlenecks". The measurement programs which are currently available give, at best, only an indirect indication of where the problem is, and frequently almost no information about the cause. The parts of an operating system which have a major effect upon system performance need to be identified so that they can be studied more carefully. For example, the job selection procedure which determines the job mix, the central processor dispatching algorithm, the I/O dispatching algorithm, and the main storage management all have significant effects on performance. Recently, a study of central processor dispatching algorithms was begun and a simple rule was found which is optimal for an idealized model. It is hoped that these results can be extended and possibly a similar approach can be taken to study other critical parts of operating systems.

C. Performance Measurements and Optimization of a Computing System

After observing the operation of several computing systems, it becomes obvious that the performance of these systems is very opaque. Even gross measures of systems utilization are hard to come by. In response to this situation work has been done on the IBM360/91 at SLAC, using the several hardware and software monitors already in existence there to accurately determine the spectral characteristics of several parameters of the computer so that subsequent measurements of the system can be done cheaply, as well as accurately.

In addition, a scheduling algorithm is being developed which can determine the optimal (minimum time) sequence in which to access a set of data records form a computer's storage drum. The algorithm has the very nice property of having its execution time grow as $N(\log N)$, where N is the number of records to be accessed.

D. Computer Architecture

A comparison has been made between an orthogonal processor and an array processor of similar complexity. The results of this analysis are contained in a forthcoming DSL Technical Note.

An analysis into the feasibility of using "cache" or buffer memories in a mini-computer, specifically, the Hewlett-Packard 2116, has been conducted. As a secondary result, a simulator for the HP 2116 now exists which is able to save a trace of the execution sequence of any program written for the HP 2116.

E. The Transformation of Sequential Programs into Parallel Programs

One of the goals of this research is to produce compilers which will translate programs written in the usual sequential fashion into efficient parallel codes.

It can be argued that codes produced by such compilers cannot be as efficient as codes directly obtained by using special purpose parallel algorithms.

The following theoretical problems remain to be solved:

(1) Choosing an adequate representation of parallel programs

(2) Defining such aspects as the "degree of parallelism: of a program, and the "maximal parallel form" of a program

(3) Obtaining sufficiently large classes of program schemata for which there exists an algorithm to derive a maximal parallel form of a program

(4) Obtaining results to show that wider classes of program schemata will lead to fundamental undecidability

(5) Devising transformations that will improve parallelism in a large number of cases, where best maximal form cannot be obtained.

Reasonable models and definitions for (1) and (2) have been adopted. We have proved that it is undecidable whether two processes in a flowchart schema (as defined by Luckham, Park, and Peterson [1]) are executable in parallel, bringing us one step towards (4). We are currently considering a rather large subclass of schemata (non-repetitive schemata), but have not yet been able to solve (3) for it.

This research can yield as a byproduct interesting theoretical insights into such topics as representation of programs and equivalence of programs.

REFERENCE

[1] Luckham, D. C., J. C. Park, and G. H. Petersen, "On Formalized Computer Programs," Programming Research Group, Oxford University, Oxford, England, August 1967.

Unsponsored

Principal Investigator: E. J. McCluskey

Staff: G. N. Shapiro

Title: A FUNCTIONAL APPROACH TO STRUCTURED COMBINATIONAL-LOGIC DESIGN

This work treats the problem of designing large combinational-logic networks. The adjective , combinational-logic , refers to directed networks of switching elements where all of the outputs are uniquely-determined combinations of the present inputs. Acyclic, or loop-free, combinational-logic designs, in particular, have been pursued.

Our new approach to loop-free logic design advocates the use of standard-patterns for interconnecting the basic logic units. Each such interconnection pattern, or structure, applies to a particular class of functions. A range of structures exist for each function class. Thus a specific logic design task is solved here by fitting that task into a function category, and then choosing an associated structure.

The particular novel element of our approach is the generality of the proposed standard structures. Each standard structure can be used with a variety of basic logic units. Since these basic logic units form functions, we call these structures which are functions of functions, functionals. That is, each functional is a generalized description of a -collection of loop-free logic nets.

This generality allows a departure from the current practice of assuming a particular logic technology. Here the logic composing the basic units, or modules, can be chosen after the interconnections are fixed. The emphasis is placed on using a minimum of module types in a regular interconnection pattern. Consequently, this work is adaptable to a wide range of logic design technologies, with particular application to large-scale integrated design(L.S.I.).

The work is divided into analysis of structures for three different situations:-

1) The most general class of switching functions,

2) A particular class of switching functions,namely those symmetric about all arguments.

3) A specific task, namely loop-free sorting.

From the survey of sorting networks emerge several new and efficient sorting algorithms. Of primary importance, however, is the framework used for categorizing such networks according to certain construction features.

Standard-structures are presented for forming any switching function of n variables to show the range of logic tasks covered by our ideas. These structures are derived by adapting the linear algebra concepts of vector space and bases to switching functions and Boolean logic. Expansion theorems play an important role in the analysis of the allowed choices of module types in that functional.

Analysis of structures for the particular class of symmetric switching functions is then undertaken to show that our functional approach can apply there too. The functionals derived are related to the generalized encoders and decoders for arbitrary functions. Two particularly important instances of symmetric function sets are those for counting and for thresholding. The networks for the lacer are directly related to those for sorting,

The cost of these standard structures is shown to be quite reasonable. The generalized encoders and decoders require on the order of: 2^n modules for arbitrary functions, and n^2 modules for symmetric functions, of n variables. More complex arrangements are also described . to reduce this cost to within the theoretical 'worst-case least-upper-bounds!' Namely, the module count can be on the order of : $2^n/n$ for arbitrary functions , and n for symmetric functions.

The functional approach advocated in this work also illuminates many unsuspected relationships between diverse logic design tasks. One example is the connection between counting and sorting. Other examples fall out of the ties made between linear algebra and logic design. This ecumenical approach may help the designer perceive more of the relationships between specialized fields of knowledge. And with this general outlook, perhaps the designer will not be obsolete on the day his higher education cease?.

Contract: AEC, AT(04-3)326 PA23

Principal Investigator: W. F. Miller

Staff: H. S. Stone

Title: PARALLEL COMPUTER ORGANIZATION

Major research effort is directed to the study of parallel computers and algorithms for parallel computation. The objective of the research is to invent new ways of organizing computers that can perform parallel computation with high efficiency.

In the past year a memory-to-processor interconnection pattern called the perfect shuffle was studied, and it has been found that the most efficient parallel algorithms known today for sorting and for Fast Fourier Transform make use of the perfect shuffle. The study suggests that the perfect shuffle is a fundamental interconnection that heretofore has not been considered for use in a parallel computer. Several other efficient parallel algorithms that depend on the perfect shuffle were found, as well as general characteristics of perfect shuffle algorithms.

Other effort has been directed to scheduling parallel computation. Substantial progress has been made in finding an efficient solution to the n-task, 3-machine scheduling problem that is one of the "classical" combinatorial scheduling problems. The problem can be broken into four subproblems, for which efficient solutions for three have been found. A partial solution to the fourth subproblem has also been found, but it is still an open question as to whether there exists a complete efficient solution for it.

COMPUTING AND BUSINESS EDUCATION

Norman R. Nielsen, Associate Professor of Operations and System Analysis
Graduate School of Business

In 1968 it became clear to the faculty of the Graduate School of Business that they were not taking appropriate advantage of computing in the training of masters and doctoral candidates. Computing was playing an ever expanding role in the conduct of business in the country, yet the role of the computer in the business school had not enjoyed such a growth. Accordingly, a faculty task force was convened to examine the role of the computer in the curriculum, in pedagogy, and in research.

During the course of the subsequent deliberations, more than 50% of the faculty became actively involved. The most far reaching of the task force's conclusions came in the area of curriculum. It was recommended that more be taught about the computer per se as well as its applications in the business world. It was recommended that the various functional areas (such as marketing, finance, accounting) place greater emphasis not only upon using the computer but upon the effect of the computer in that functional field. In other words, a significant and comprehensive revision of the entire curriculum was recommended.

In the 1968-69 and 1969-70 school years the task force recommendations were implemented in the first year of the two year MBA (Master of Business Administration) programs. A computer laboratory was constructed in the basement of the business school, specifically designed to house 15 time-sharing

terminals connected to the Stanford 360/67. This facility serves a student population of 600 masters candidates and 100 doctoral candidates. In addition two more terminals are located in the business school library. The ready availability of access to the computer as well as the time-sharing mode of operation permit a close and immediate student-machine interaction.

In addition, a new course has been developed, Management and the Computer, which is required of all entering students. This course provides the students with a background knowledge of computer hardware and software, a practical skill in BASIC programming, and some insights into the various types of computer applications in the business world. This course brings all of the school's students up to a common level of computer background. Although the students are by no means programmers upon completing the course, they do have a familiarity with the system and can develop their own programs to solve problems.

The remainder of the curriculum developments are taking place within the existing course structure. Not only is a greater emphasis being placed upon the effect of the computer in the various subject areas, but the computer is also being used to aid in the education process itself. Library or "canned" programs are used by students to automate routine but lengthy computations. Not only does this save drudgery but it also permits students to devote their time to studying the implications of the results thereby obtained rather than upon obtaining those results. In other instances, programs have been developed by the faculty to assist

the student in analyzing some of the case problems used in the business school courses. (The cases are real life situations and form an important part of the school's program.) Students working with the computer in this fashion are able to obtain not only a better mastery of concepts involved but they are able to gain that mastery in a shorter period of time.

In still other situations the student will have to develop his own programs to analyze problems or complete assignments. The computer is also used to carry out the computations necessary for various types of business "games". (In such a game student teams or "companies" compete in a simulated industry, making periodic decisions about production, marketing, finance, etc. The computer evaluates the relative effect of each set of decisions upon the simulated firms, and then provides updated information to each team. Then another set of decisions is made, etc.)

The making of the above types of curriculum changes, the development and documentation of the necessary computer programs, and the testing of new teaching materials requires a substantial amount of faculty, doctoral research assistant, and computer time. Consequently, most of the developments have been restricted to the first year of the MBA program.

However, funding for a new effort has been provided by the National Science Foundation under a two year curriculum development grant. This new effort is intended to institute the above type of curriculum revision throughout the second year of the MBA program as well as the Ph.D. program.

INFORMATION RETRIEVAL AND LIBRARY AUTOMATION: SPIRES/BALLOTS

The Problem Context

The publication explosion, the compelling need for access to information, and rapid library growth are not unique to Stanford University. At Stanford, a commitment has been made to deal with the information problems of the university by improving library service and developing a campus based bibliographic retrieval system. Using the tools of computing technology and library systems analysis, computer specialists have joined with librarians and behavioral scientists in exploring the problems and creating the systems to meet the bibliographic requirements of a major university community.

Library automation requires a major system development effort and sir&able expenditures for computer equipment. Computerized information storage and retrieval requires an equally large investment in hardware and software. Both efforts have common conceptual problems in such areas as bibliographic file organization and online searching. Each effort derives benefits from the other. Bibliographic files created in the process of library automation are available for general ized retrieval uses, and complex retrieval routines are available for search of library bibliographic files

SPIRES/BALLOTS Project

At Stanford, two major projects have been working jointly on library automation and information retrieval since 1968. One is BALLOTS (Bibliographic Automation of Large Library Operations on a

Time-sharing System), funded by the Office of Education and the other is SPIRES (Stanford Physics Information REtrieval System--informally known as the Stanford Public Information REtrieval System), funded by the National Science Foundation. The purpose of this collaboration is to create the common software required to support both the BALLOTS and the SPIRES applications. The joint effort is overseen by the SPIRES/BALLOTS Executive Committee chaired by Professor William F. Miller, Vice-President for Research.

The Stanford project structure and-system development philosophy reflect the common uses and individual needs of both BALLOTS and SPIRES. The concept of shared facilities refers to the system software and hardware designed to service both the BALLOTS applicaton and the SPIRES application. Examples are, an on-line text editor and a computer terminal handler. Both are shared software facilities which can service bibliographic input and specialized research files. Computer hardware such as a central processing unit or direct access devices (allowing shared files) are examples of shared hardware facilities. Combining resources in this system development effort reduces the cost of creating common facilities and provides a pool of skilled manpower resources for each area.

BALLOTS I and SPIRES I

In 1967 the Stanford University Libraries and the Institute for Communication Research began research projects with funds from the Office of Education (BALLOTS) and the National Science Foundation (SPIRES) respectively. In 1968 the shared perspective and close collaboration of these two projects was formal ized by placing them under the SPIRES/BALLOTS Executive Committee.

Stanford University was an appropriate setting to initiate research and development in bibliographic retrieval. Interest in automation was strong in all areas of the Stanford University Libraries and especially with its Director (then Associate Director), David C. Weber, and Assistant Director for Bibliographic Operations, Allen B. Veaner. The library had achieved during 1964-66 a remarkably successful computer produced book catalog for the J. Henry Meyer (Undergraduate) Library. Professor Edwin B. Parker and his colleagues at the Institute for Communication Research were already applying to computer systems the behavioral science analysis which had previously been applied to print, film and television media. The Stanford Campus Facility had an IBM 360 model 67 computer, a locally developed time sharing system and a first rate programming staff associated with one of the nation's leading Computer Science departments. A close working relationship between the University Libraries, the Computation Center, and the Institute for Communication Research was the firm foundation for research and development.

The project software development group applied itself to writing programs necessary for bibliographic retrieval. In the Library, an analysis and design group worked closely with the library staff in studying library processes and defining requirements. This joint effort created a prototype system which could be used in the main library and by Stanford faculty and -students, primarily high energy physicists.

In early 1969, two prototype applications were activated using the jointly developed systems software; an acquisition system was established in the Main Library (BALLOTS I) and a bibliographic

retrieval system (SPIRES I) was established for a group of High Energy Physicists.

Centralized management of library input was handled by two newly created departments, Data Preparation and Data Control; In the 1 i brary, several terminals were installed for on-line searching. An on-line In Process File was created consisting of 30% of the Roman alphabet acquisition material ordered by the library. On-line searching was conducted daily during regular library hours by a special ly trained staff. This prototype system operated during most , of 1969, demonstrated the technical feasibility of the combined proj ect goal s. It was studied and evaluated by the lirary systems and programming staffs. The human, economic, and technical requirements of a library bibliographic retrieval system were considered.

At the Stanford Linear Accelerator Center (SLAC) Library a file of preprints in high energy physics was created using SPIRES I. This file is still active. Records of new preprints are added week1 y, and a note is made of any preprint that is published. Input is via an IBM 2741 typewriter terminal in the SLAC Library. The preprint file contains approximately 6500 documents, including all the high energy physics preprints received in the SLAC Library for a period from March 1968 to the present. input and update is done by regular library staff at SLAC. Searching is possible by author, title, date and citation.- "Preprints in Particles and Fields" a weekly listing of preprints is produced from SPIRES I. It is now supported partially by subscriptions after an initial period of support by the Division of Particles and Fields of the American Physical Society.

BALLOTS II and SPIRES II: System Development

The result of operating the prototype applications (BALLOTS I and SPIRES I) was encouraging, particularly with respect to the advantages of utilizing common software. Feasibility and usefulness were clearly established and a wealth of knowledge was gained under actual operating conditions. The joining of library and retrieval application areas served by Shared Facilities (hardware and software) was shown to be a rewarding approach.

BALLOTS I and SPIRES I resulted from a development process in which user requirements were analyzed, programs written and tested, and protoyptes created and evaluated. Librarians, behavioral scientists, library systems specialists and computer specialists collaborated over an extended period of time. The development process which produced the successful prototype system was a major milestone. The outcome was the definition of a production bibliographic retrieval system with distinctive hardware and software requirements.

The creation of a production system for library automation (BALLOTS II) and generalized information storage and retrieval (SPIRES II) requires the continuation of a comprehensive System Development Process. This process is a framework within which tasks are defined, assigned and coordinated. The System Development Process for the creation of BALLOTS II and SPIRES II has six phases:

Phase A: Preliminary Analysis

Phase B: Detailed Analysis

Phase C: General Design

Phase D: Detailed Design

Phase E: Implementation

Phase F: Installation

Phase F: Installation

Preliminary Analysis involves the definition of goals, description of the user environment, analysis of the existing system, selection of the system scope and establishment of gross technical feasibility of the selected first implementation scope. These factors are stated in detail in a System Scope Document which is the main output of the Preliminary Analysis Phase.

Detailed Analysis enumerates minutely the requirements to be met by the manual -automated system. (1) Performance requirements are stated quantitatively, including response time, hours of on-line accessibility, allowable mean failure time, maximum allowable recovery time and similar factors. (2) Record input/output is determined in terms of volume, growth, and fluctuations. Timing considerations for batch input/output are determined in order to plan for scheduling requirements. (3) All input/output document formats are determined on a character by character basis. (4) Rules transforming input data elements into output data elements are formulated and tabulated, and (5) the upper bounds of development and operating costs are established.

General Design encompasses both system externals (procedures, training, reorganization, etc.) and system internals (alternative hardware and software solutions to the stated requirements). An overall software-hardware configuration is selected and expressed in a General Design Document.

Detailed Design completes the internal and external design, creates implementation and testing plans, and provides programming specifications. These are incorporated in a Detailed Design Document.

In the installation Phase, training of all personnel is completed, fi les are converted and, after a time of parallel operation with the manual sys tern, a changeover is made to the automated system. Performance statistics are collected and a support plan and project history are written.

Each phase description has been necessarily abbreviated. Not all activities or outputs have been described. Some of the phase activities overlap and feed back to redefine previous activities. A "Wishbook" which has been maintained through all phases is put in final form in the Installation Phase. The "Wishbook" is very important since it represents the link to successive development iterations. It contains information on capabilities, services and operational characteristics the desirability of which became apparent during the development process but which could not be included because of time, cost or technical constraints. The Wishbook also contains information on internal (programming or hardware) and external (user or procedural) operational deficiencies determined after the system has been running for some period of time. This information will be considered in designing new portions and will aid in the overall improvement of the system.

This statement of the System Development Process guides SPIRES/BALLOTS II development from the definition of goals to the installation of a fully operational system.

BALLOTS I I and SPIRES I I : Goals

The project goals are presented as they relate to Library Automation (BALLOTS), Generalized Information Storage and Retrieval (SPIRES), and Shared Facilities. These goals are interrelated. The

goals of Shared Facilities (hardware and software) suport and serve the goals of BALLOTS and SPIRES.

BALLOTS

As the major information center of a large academic institution, the library must respond effectively and economical1 y to the university community. The library is a complex combination of people and machines providing the major bibliographic resources of the university to students and faculty. It reflects the needs and priori ti'es of a changing university environment. The university library is also part of a larger network of information sources which includes other research libraries, The Library of Congress and specialized information storage agencies.

The essential goals of BALLOTS are expressed in a library system (both the manual and automated protions) which is: USER RESPONSIVE. It adapts to the changing bibliographic requirements of diverse user groups within the university community. COST COMPETITIVE. It provides fast, efficient internal processing of increasing volumes of processing transactions. SYSTEM OPTIMIZED. It is not an attempt to automate protions of the existing manual system. It is based on the actual operating requirements of library processing and is not dependent on the exi sting procedura 1, organizational or physical setting. PERFORMANCE OR I ENTED. It provides the library and university administration with data which are useful for the measurement of internal processing performance and user satisfaction. FLEXIBILITY. It has the capability for expansion to embrace a broader range of services and a wider group of users. It will be able to link up and serve other information systems and effectively use national data sources.

These goals wi 11 he expressed in specific capabilities which will (among other things): minimize manual fi 1 ing, el iminate many clerical tasks now performed by professionals, and provide user suggestion mechanisms. The effect of these computer capabil i ties wi 11 be: to drastically reduce errors associated with manual sorting, typing and hand transcription; to speed the flow of material through library processing; to aid book selection by providing fast access to central machine files; and to enable librarians to advise a patron of the exact status of a work about which he inquiries. In summary, responsiveness to 1 i brary users, efficiency of operation, optimization, performance monitoring and flexibility for future improvement, are the essential goals of library automation.

SPIRES

. The SPIRES generalized information storage and retrieval system wi 11 support the research and teaching activities of the library, faculty, students, and staff. Each user wi 11 have the capabi 1 ity of defining his requirements in a way which automatically tailors the system response to his individual needs. The creation of such a system is a major activity involving the study of users, source data, record structure, -file organization-and considerable experimentation with facilities. The SPIRES system will be characterized by flexibility, generality and ease of use. The goals of SPIRES in specific areas are as follows: DATA SOURCE AND CONTENT. A general ized information storage and retrieval capability will store bibliographic, scientific, administrative and other types of records in machine readable form. Collections will range from large public files converted from centrally produced machine-readable data to

medium-small files created from user generated input (faculty, student files). SEARCH FACILITIES. It will provide the capability for searching files: interactively (on-line) via a computer terminal, on a batch basis by grouping requests and submitting on a regular schedule or on a standing request basis in which a search query is routinely passed against certain files at specified intervals. FEEDBACK. Reports on the use frequency of various system elements wi 11 be provided. This will include statistical analyses of user difficulties and system errors. RECORD MODIFICATION. Update and edit capability will be provided on a batch basis or on-line; and options for update wi 11 be at the level of record, data element and character string within data element. COSTS AND CUSTOMERS. The cost of these services should be sufficiently low for a wide range of customers to cost justify their use of the system. The variety of services should be sufficiently great to encourage a growing body of users. Costs and services must be related at various levels to permit users to select the type of service which meets their needs within the limits of their economic resources.

BALLOTS and SPIRES Shared Facilities

Shared facilities are software and hardware designed to provide concurrent service to BALLOTS and SPIRES applications. Since the sharing of such resources represents a substantial savings to all appl ications served, maximum attention will be given to the sharing concept. Whenever possible, advantage will be taken of economies gained by providing major facilities for multiple applications. HARDWARE. The hardware environment will provde reliable, economical, and flexible support to those applications residing within it.

SOFTWARE. The software, which will consist of an operating system, an on-line executive program, a terminal handler, a text editor, and many other facilities, will be jointly used by various applications.

GENERALITY/EXPANDABILITY. The shared facilities will be designed to allow growth of the current applications as well as to allow the addition of new applications to Shared Facilities without modification to previous applications.

TOWARD AN INFORMATION FACILITY

The operational environment for SPIRES/BALLOTS has implications beyond bibliographic retrieval and library automation applications. There is a growing need for computer and other information retrieval services in support of socially significant research. Such research is being conducted, for example, in the developing fields of ecology and- urban studies.

Several capabilities and services are required. Data Banks of bibliographic and other information are needed for studies which draw upon several disciplines. Strong disciplinary information systems (e.g., psychology) and centralized national systems (e.g., ERIC) produce large amounts of data on magnetic tapes. In addition, data is generated by local research, at Stanford and at nearby centers. An Information Facility with large scale storage equipment and sophisticated program capabilities can create and maintain data banks derived from several input sources.

Data selected from large machine-readable files can be subjected to further computer processing. Programs that perform mathematical or statistical analysis can be used to produce evidence for a problem solution which may not have been considered when the data was first

gathered. Fresh insights can often be obtained without the necessity of generating large amounts of new data. Similarly, new data can be added to an existing file and up-to-date analyses performed to confirm or extend the conclusions of previous studies.

In addition to the large amounts of information in machine-readable form, even larger amounts are now available in microforms. This includes, microfiche, microcards, and microfilm. Computer generated on-line indexes to massive microform files are a form of information retrieval that an Information Facility can provide.

In social research there is an increasing premium placed on providing information fast and at the site of research, often beyond the university campus. A computer Information Facility can meet these time and distance requirements. Remote terminals in a nearby city can access a Stanford computer. Direct access storage devices operating with time sharing software can provide immediate interactive response. Stanford has several years experience in operating a multi-user interactive system.

An Information Facility based on SPIRES and BALLOTS combines production operating characteristics and sophisticated search and retrieval- capabilities. Reliability, security, fast recovery, and cost acceptability are required to support library and administrative operations. Ease of use by people with non-technical backgrounds (faculty or students) and extensive search request capabilities are needed by the SPIRES users. Through a combined facility, librarians will be able to use one or more search programs and a researcher will

be able to access library files. Both of these activities can be carried out simultaneously from different locations. Common software such as a terminal handler serves all user groups.

In a sense, a comprehensive Information Facility is an 'extended library." The boundaries of this library are not physical walls but the telecommunication limits of the facility terminals. A researcher will not need to go to the library catalog to search for material, the "catalog" will be as close as a terminal.

Such an Information Facility will be used by several major user groups. These groups include not only librarians and university administrative personnel but also researchers at Stanford and in nearby locations. In the past, and even now, no one of these user groups could afford to have a computer facility devoted to its own information needs. By creating an Information Facility designed to serve the daily operational needs of the university and the special information needs of various research groups, computerized information services can be offered at a favorable cost-benefit ratio.

BIBLIOGRAPHY
Publications and Reports

AUTOMATION NEWSLETTER. PROJECT BALLOTS., published by the Automation Division, Stanford University Libraries. Stanford, California.

No. 1	June 1969
No. 2	Sept. 1969
Mo. 3-4	March 1970 combined issue

Parker, Edwin B. "Behaviorial Research in the Development of a Computer-based Information System." Delivered at the Conference on Communication among Scientists and Technologists: The Study of the Production, Dissemination and Use of Information by Scientists and Technologists. The Johns Hopkins University, Baltimore, Maryland, October 28-30, 1969. To be published in the Conference Proceedings.

Parker, Edwin B. "Democracy and Information Processing." EDUCOM (Bulletin of the Interuniversity Communications Council), 1970, 5: 4, pp. 2-6.

Parker, Edwin B. "Developing a Campus Based Information Retrieval System," in PROC. OF STANFORD CONFERENCE ON COLLABORATIVE LIBRARY SYSTEMS DEVELOPMENT. (Stanford University, Stanford, California, Oct. 4-5, 1968.) Stanford University Libraries, Stanford California, pp. 213-230, 1969. (ERIC document number ED 031 281)

Parker, Edwin B. "Information Utilities and Mass Communication." In Nie, N. and Sackman, H. (ed.) AN INFORMATION UTILITY AND SOCIAL RESEARCH. Montvale, New Jersey: AFIPS Press, 1970.

Parker, Edwin B. "A System Theory Analysis of Feedback Mechanizms for Information Systems." Proceedings of FID International Congress of Documentation. September 21-24, 1970. Buenos Aires, Argentina.

RECON Working Task Force (Allen B. Veaner, Member) CONVERSION OF RETROSPECTIVE CATALOG RECORDS TO MACHINE-READABLE FORM. Library of Cbngress, Washington D.C., 1969.

RECON Working Task Force (Allen B. Veaner, Member) "Levels of Machine-Readable Records," JOURNAL OF LIBRARY AUTOMATION, Vol. 3, No. 2 PP. 122-127 line, 1970.

SPI RES REFERENCE MANUAL. Stanford University, Stanford, California. January 1969. Revised July 1969.

SPIRES/BALLOTS Report. A 15-minute, color 16mm film giving an overview of the library automation and information retrieval problem in general and Stanford's approach. Written and directed by D.B. Jones. Produced by the Department of Communications, Stanford University, 1969.

SYSTEM SCOPE FOR LIBRARY AUTOMATTON AND GENERALIZED INFORMATION STORAGE AND RETRIEVAL AT STANFORD UNIVERSITY. Stanford University, Stanford, California. February 1970. Second printing, April 1970. p. 157 (ERIC document number ED 038 153)

Veaner, Allen B. "The Application of Computers to Library Technical Processing." COLLEGE AND RESEARCH LIBRARIES, *Jol.* 31, No. 2, pp. 36-42, January 1970.

Veaner, Allen B. "Major Decision Points in Library Automation." Delivered at the 75th Annual meeting of the Association of Research Libraries, Chicago, Illinois, January 17-18, 1970. COLLEGE AND RESEARCH LIBRARIES, Vol. 31, No. 5, pp. 299-312. The full text is in the published meeting minutes, pp 2-33.

Veaner, Allen B. "Stanford University Libraries, Project BALLOTS, A Summary," in PROC. OF STANFORD CONFERENCE ON COLLABORATIVE LIBRARY SYSTEMS DEVELOPMENT. (Stanford University, Stanford, California, Oct. 4-5, 1968.) Stanford University Libraries, Stanford, California, pp. 42-49, 1969. (ERIC document number ED 031 281)

Weber, David C. "Personnel Aspects of Library Automation" Delivered at the Information Science and Automation Division Program Meeting of The American Library Association, Detroit, Michigan, July, 1970. Submitted to the JOURNAL OF LIBRARY AUTOMATION.