Massachusetts Institute of Technology

Project MAC Progress Report V July 1967 - July 1968

Massachusetts Institute of Technology

Project MAC

545 Technology Square

Cambridge, Massachusetts

02139

Work reported herein was supported by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office Research Contract Number Nonr-4102(01). The primary support for some of this work came from the M.I.T. departments and laboratories participating in Project MAC, whose research programs are, in turn, sponsored by various government and private agencies. This support is acknowledged by specific mention of agency and contract number in the appropriate sections.

BLANK PAGE

ADMINISTRATION

ARTIFICIAL INTELLIGENCE

COMPUTATION STRUCTURES

COMPUTER SYSTEM RESEARCH

ELECTRONIC SYSTEMS LABORATORY

GRAPHICS RESEARCH

INTERACTIVE MANAGEMENT SYSTEMS

MAN-MACHINE COMMUNICATION

PROGRAMMING LINGUISTICS

RESEARCH LABORATORY OF ELECTRONICS

SCHOOL OF ENGINEERING

SCHOOL OF HUMANITIES AND SOCIAL SCIENCES

SCHOOL OF SCIENCE

TECHNICAL INFORMATION PROGRAM

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

PERSONNEL

FROM JULY 1967, TO JULY 1968

Administration

Prof. R. M. Fano
- Director

Prof. J. C. R. Licklider
- Associate Director

Prof. M. M. Jones
- Assistant Director

Committee On Computation

Dean G. S. Brown (Chairman)
Dean R. A. Alberty
Prof. R. M. Fano
Prof. R. I. Hulsizer, Jr.
Prof. W. B. Kehl
Prof. C. L. Miller
Prof. P. M. Morse
Prof. C. F. J. Overhage
Dean W. F. Pounds
Prof. L. D. Smullin

Academic Staff

C. A. Berg
G. D. Bernard
A. Bers
K. Biemann
W. L. Black
M. Blum
R. J. Briggs
S. C. Brown
D. C. Carroll
J. G. Charney
S. A. Coons
F. J. Corbató
J. B. Dennis
M. L. Dertouzos
J. J. Donovan
A. Evans, Jr.
R. R. Fenichel
J. W. Forrester
E. L. Glaser
G. A. Gorry
L. A. Gould
R. M. Graham
H. A. Haus
W. L. Henke
F. C. Hennie
R. A. Humphrey
I. d Pool
M. M. Jones
R. Kaplow
M. M. Kessler
Z. Kohavi
E. Kuh
F. F. Lee
J. D. C. Little
C. L. Liu
J. A. Mangano
W. A. Martin
G. H. Matthews
F. A. McClintock
H. S. Mickley
J. H. Milligram
M. Minsky

Academic Staff (continued)

J. Moses
J. Narud (Visiting)
D. N. Ness
C. F. J. Overhage
S. Papert
R. R. Parker
R. P. Parmelee
H. M. Paynter
P. L. Penfield, Jr.
D. Prerau
J. F. Reintjes
R. C. Rosenberg
J. H. Saltzer
M. M. Scott-Morton
T. B. Sheridan
L. D. Smullin
C. R. Sprague
K. N. Stevens
A. K. Susskind
R. L. Taggert
D. E. Troxel
J. Weizenbaum
J. M. Wozencraft

Non-Academic Research Staff

R. J. Abbott
M. C. Basile
G. W. Baylor
M. D. Beeler
W. Bennett
R. J. Bigelow
T. O. Binford
B. F. Boudreau
H. E. Brammer
M. F. Brescia
M. L. Cabral
R. H. Campbell
J. J. Cecil
T. B. Cheek
G. F. Clancy
R. W. Cornew
R. C. Daley
T. F. Dempsey
S. D. Dunten
C. Duren
M. Eisner
M. N. Fateman
C. G. Feldmann
R. J. Gardner
C. Garman
R. W. Gosper
R. Greenblatt
D. M. Griffel
H. F. Gronemann
J. A. Gunn
J. Harwitt
R. W. Henneman
J. H. Hewitt
F. B. Hills
P. Himot
J. T. Holloway
P. Johansen
E. R. Jensen
D. M. Jordan
R. L. Kusik
T. Lin
J. B. Lovins
C. Marceau
R. S. Marcus
P. Marmarelis
K. J. Martin
W. D. Mathews
S. D. McIntosh
E. W. Meyer
N. I. Morris
L. H. Morton
R. C. Nelson
W. Nissen
W. R. Nofsker
S. Ohayon
M. A. Padlipsky
R. B. Polansky

Non-Academic Research Staff (continued)

R. L. Rappaport
J. E. Rodriguez
J. S. Roe
S. L. Rosenbaum
D. T. Ross
J. R. Ross
K. D. Rude
P. Samson
P. M. Sheehan
T. P. Skinner
R. Sommer
W. H. Southworth
J. W. Spall
C. Spielman
M. J. Spier
R. L. Stetson
R. H. Stotz
W. D. Stratton
M. R. Thompson
D. E. Thornhill
O. J. Tretiak
M. C. Turnquist
T. H. Van Vleck
A. Vezza
G. Voyat
V. Voydock
D. B. Wagner
J. F. Walsh
M. E. Wantman
J. E. Ward
S. A. Ward
S. H. Webber
T. S. Weston
J. L. White
D. R. Widrig
B. L. Wolman
R. V. Zara

Instructors, Research Associates, Research Assistants and Others

J. W. Alsop
E. I. Ancona
R. Baecker
E. R. Banks
T. J. Barkalow
W. L. Bass
D. A. Belfer
J. R. Berdell
W. T. Beyer
A. K. Bhushan
J. W. Brackett
S. R. Brueck
R. H. Bryan
J. R. Carbonell
R. A. Carpenter
D. G. Chapman
E. Charniak
P. Choong
S. L. Chou
D. D. Clark
R. Crochiere
S. D. Crocker
J. A. Davis
H. Dietel
P. J. Denning
F. DeRemer
O. J. Dodson
R. S. Eanes
M. Edelberg
R. Feiertag
M. Fisher
D. L. Flamm
J. C. Free
N. D. Fulton
R. C. Gammill
S. Geffner
M. A. Geller
G. W. Goddard
L. I. Goodman
A. J. Gottlieb
H. L. Graham
J. E. Green

Instructors, Research Associates, Research Assistants and Others (continued)

R. S. Green
H. Greenbaum
R. Greenblatt
R. P. Greer
I. G. Greif
A. K. Griffith
J. M. Grochow
K. Grossen
T. K. Gustafson
E. G. Guttmann
A. Guzmān
J. A. Hamilton
K. Hatch
T. L. Hawk
P. Hebalker
D. A. Henderson
C. Hewitt
R. Hodges
N. Hoeg
W. Hutchison
W. M. Inglis
G. P. Jessel
T. L. Jones
M. E. Kaliski
W. R. Kampe
L. J. Krakauer
J. F. Kramer
B. R. Kusse
N. Leal-Cantu
M. J. Lennon
L. M. Lodish
F. L. Luconi
M. E. Manove
E. M. Mattison
P. D. McMorran
P. F. Meyfarth
J. R. Miller, III
G. H. Mitchell
J. H. Morris, Jr.
A. Moulton
J. M. Nead
R. R. Parker
S. S. Patil
D. N. Perkins, Jr.
K. P. Polzen
A. D. Pugh, III
H. A. Radizkowski, II
R. G. Rausch
C. L. Reeve
M. Richards
H. M. Schneider
M. Schroeder
L. Selsnick
L. Seligman
L. L. Selwyn
D. R. Slutz
C. R. Smith
T. L. Smith
S. W. Smoliar
M. Speciner
W. Stallings
J. Stinger
G. J. Sussman
J. R. Sussman
C. W. Therrien
R. H. Thomas
R. C. Thurber
C. C. Tillman, Jr.
H.-M. Toong
D. Vedder
D. H. Vanderbilt
R. J. Wald
R. N. Wallace
D. L. Waltz
C. Weissgerber
P. Wieselmann
D. A. Wright
C. Ying
I. T. Young

Technical Assistants

T. F. Callahan
D. E. Eastlake
R. Fidler
G. L. Rosen

Operating Staff

G. Andrews
C. Brandon
K. Carley
J. Cohan
R. Crowley
R. A. Degan
R. Hoeft
J. Kulick
D. Lang
J. McGillivary
R. McNamara
P. Monaco
R. Moore
D. Murphy
G. Noseworthy
M. Pagliarulo
D. Peaslee
E. Reardon
J. W. Waclawski

Technicians

R. Hill
R. G. Splaine

Machinist

N. F. Stone

Administrative and Supporting Staff

C. S. Alles
J. C. Anderson
M. E. Baker
C. J. Carter
L. Cavallaro
B. Chis
J. M. Constantine
K. F. Diamond
L. Frechette
C. D. Graceffa
D. L. Jones
F. Jones
E. Kampits
D. Kontrimus
L. L. Maddaus
B. L. McLean
E. T. Moore
S. C. Morr
B. Mutnick
J. E. Piñella
E. M. Roderick
D. C. Scanlon
L. S. Sloan
M. Stallings
S. L. Stone
K. M. Tomaselle
M. Webber
C. M. White

Resident Guests

N. Adleman	System Development Corporation
F. Bates	Union Carbide Corporation
D. J. Cameron	Ferranti Limited
J. T. Doherty	Raytheon Mfg. Company
K. Engelmann	MITRE Corporation
A. Forte	Yale University
R. B. Gluckstern	UNIVAC Div., Sperry Rand
H. J. Hebert	Shell Development Company
G. L. Lane	Sandia Corporation
S. Lorch	Massachusetts General Hospital
R. J. McDowell	Honeywell EDP
M. T. McGuire	Massachusetts General Hospital
C. Mercer	Informatics, for Rome Air Defense Command
R. A. Meyer	IBM Corporation
G. A. Miller	Harvard University
A. T. Nagai	The Boeing Company
G. C. Quarton	Massachusetts General Hospital
A. Sasaki	ElectroTechnical Laboratory of Japan
P. Schicker	Swiss Federal Institute of Technology
R. Silver	MITRE Corporation
P. J. Stone	Harvard University
W. R. Strickler	Shell Development Company
I. Wenger	Raytheon Mfg. Company
S. Zurnaciyan	Northrop Corporation

PREFACE

Project MAC was organized at the Massachusetts Institute of Technology in the spring of 1963 for the purpose of conducting a research and development program on Machine-Aided Cognition and Multiple-Access Computer systems. It operates under contract with the Office of Naval Research, acting on behalf of the Advanced Research Projects Agency of the Department of Defense.

The broad goal of Project MAC is the experimental investigation of new ways in which on-line use of computers can aid people in their individual intellectual work; whether research, engineering design, management, or education. One envisions an intimate collaboration between man and computer system in the form of a real-time dialogue where both parties contribute their best capabilities. Thus, an essential part of the research effort is the evolutionary development of a large, multiple-access computer system that is easily and independently accessible to a large number of people, and truly responsive to their individual needs. The MAC computer system is a first step in this direction and is the result of research initiated several years ago at the M.I.T. Computation Center.

Project MAC was organized in the form of an interdepartmental, interlaboratory "project" to encourage widespread participation from the M.I.T. community. Such widespread participation is essential to the broad, long-term project goals for three main reasons: exploring the usefulness of on-line use of computers in a variety of fields, providing a realistic community of users for evaluating the operation of the MAC computer system, and encouraging the development of new programming and other computer techniques in an effort to meet specific needs.

Faculty, research staff, and students from ten academic departments and two interdepartmental research laboratories are participating in Project MAC. For reporting purposes, they are divided into fourteen groups, whose names correspond in many cases to those of M.I.T. schools, departments and research laboratories. Some of the groups deal with research topics that fall under the heading of computer sciences; others with research topics which, while contributing in a substantive way to the goals of Project MAC, are primarily motivated by objectives outside the computer field.

The purpose of this Progress Report is to outline the broad spectrum of research being carried out as part of Project MAC. Internal memoranda of Project MAC are listed in Appendix A, and MAC-related theses are listed in Appendix B. Some of the research is cosponsored by other governmental and private agencies, and its results are described in journal articles and reports emanating from the various M.I.T. departments and laboratories participating in Project MAC. Such publications are listed in Appendix C of the report. Project MAC Technical Reports are listed in Appendix D.

Robert M. Fano

Cambridge, Massachusetts

ADMINISTRATION

Project MAC Administration

A. Personnel Changes
B. CTSS Operations
C. GE 645 Operations

Development of CTSS Resource Allocation Techniques

Academic Staff

R. M. Fano - Director

M. M. Jones - Assistant Director

J. C. R. Licklider - Associate Director

Administrative Staff

H. E. Brammer

M. L. Cabral

J. A. Gunn

T. H. Van Vleck

Research Assistant

L. L. Selwyn

Project MAC Administration - Malcolm M. Jones

A. PERSONNEL CHANGES

Several significant changes in Administrative personnel occurred during the past year. On 1 September 1967, Mr. Richard G. Mills, Assistant Director of Project MAC, was appointed Director of Information Processing Services for M.I.T., a new position created to coordinate all the education, research, and administrative computational needs of the M.I.T. community. Mr. Mills continues his association with Project MAC as Consultant to the Director.

Malcolm M. Jones, Assistant Professor of Management in the Alfred P. Sloan School of Management, was appointed Assistant Director of Project MAC to replace Mr. Mills. As a member of the Sloan School of Management research group, Professor Jones has been active in Project MAC since its inception. He continues to devote a portion of his time to teaching and research activities in the Sloan School.

On 1 January 1968, Mr. A. J. Saltalamacchia resigned as Administrative Assistant to the Director to become the Associate Publisher of Computer Design magazine. Mr. H. E. Brammer, formerly a member of the Intelligent Automata group within Project MAC, took over some of Mr. Saltalamacchia's duties. As part of the realignment resulting from the departures of Messrs. Mills and Saltalamacchia, Mr. M. L. Cabral was named Business Manager for Project MAC and put in charge of all day-to-day business activities of running Project MAC.

On February 15, 1968, Professor J. C. R. Licklider of the Electrical Engineering Department was appointed Associate Director of Project MAC. Professor Licklider's initial responsibility was directed toward formulating plans for strengthening and broadening the research program of Project MAC.

B. CTSS OPERATIONS

The most significant change in the administration of the Project MAC CTSS system during the past year was initiating system use charges on 2 January 1968. Formerly, there had been no charge for usage, but access to the system had been restricted to those who were closely associated with Project MAC.

The CTSS administrative utility programs were modified to carry additional information, and new programs were written to record requisitions authorizing expenditures, produce monthly bills, and terminate users who ran out of funds or reached their termination date. The charges for CTSS were the same as had been in effect on the duplicate CTSS system

run by the Information Processing Center and are listed in Table 1. These charges are computed daily and applied to each user's balance; if a user is out of funds he is not permitted to log in the following day. Memorandum MAC-A-259, dated 17 January 1968 describes the charging policy in detail.

During the first half of the reporting period — 1 July 1967 to 31 December 1967 — CTSS users required a total of 2,214 hours of CPU time and were logged in for a total of 55,384 hours. During the remaining time — 1 January 1968 to 30 June 1968 — CTSS users required 1,959 CPU hours and 43,146 console hours, and were charged $356,979 for the CPU time used, plus $206,409 in fees for console, access, disk storage, user-file-directory space, and disk editor usage, for a total charge of $563,388. During the latter period, the total operating expenses were $539,301. The number of users of CTSS showed a steady decline from 285 in January to 243 by June, due partly to the dollar charges, but more because of official policy not to allow any new users on the system, since the Project MAC CTSS system was to be terminated as soon as the need for it by the Multics group diminished.

No significant system software changes, other than those required to improve the billing system, occurred during this period and the hardware reliability remained at a very high level. Table 2 summarizes the overall time statistics for the system.

C. GE 645 OPERATIONS

During the past year, hardware logic bugs were discovered and corrected in the processor and GIOC subsystems. Also, several problems with the high-speed drum led to its replacement in September 1967. However, by December 1967, the 645 system was running reliably in a dual configuration. In December 1967, General Electric agreed to pay for half of the 645 configuration, enabling the system to be run as two separate 645 systems. It was agreed with GE that if any hardware units were down for maintenance, the two systems would be reconfigured, giving first priority to maintaining a usable Multics system, at the sacrifice of downing the GECOS system.

Development of CTSS Resource Allocation Techniques - Lee L. Selwyn

The objective of this project is to develop on-line interactive techniques for computer resource allocation, whereby a user may have direct communication with system administration in ordering the unique mix of available computer resources that best meets his requirements.

Starting with the design of an experimental system used for the Sloan School of Management Group (see MAC-PR-4, Appendix D) a new scheme

TABLE 1

CTSS PRICE STRUCTURE

Prime Shift 8 am - 6 pm	$300/hour
Evening Shift 6 pm - 12 pm	$200/hour
Midnight Shift 12 pm - 8 am	$200/hour
Weekend Shift 8 am Sat. - 8 am Mon.	$200/hour
FIB Shift (Foreground-Initiated Background)	$200/hour

Disk Editor	$200/hour
Disk Space	$.0067/record/day
User File Directory	$1/UFD record/day
Access Fee	$1/day
Console Hours	$4/hour for all console hours exceeding 40 times the total CPU hours on all shifts.

TABLE 2

CTSS SYSTEM UTILIZATION SUMMARY

Hours [Percent]*

Charged Time†	July '67	August '67	September '67	October '67	November '67	December '67
On-line Time	377 [50]	364 [49]	299 [41]	351 [47]	287 [40]	325 [44]
Background Time	42 [6]	46 [6]	35 [5]	33 [5]	31 [4]	24 [3]
Total Charged Time	419 [56]	410 [55]	334 [46]	384 [52]	318 [44]	349 [47]
Maintenance Time	76 [10]	80 [11]	57 [8]	61 [8]	59 [8]	65 [9]
Other Time (idle & Down)	249 [34]	254 [34]	329 [46]	299 [40]	343 [48]	330 [44]
Total Hours/Month	744 [100]	744 [100]	720 [100]	744 [100]	720 [100]	744 [100]

Charged Time†	January '68	February '68	March '68	April '68	May '68	June '68
On-line Time	298 [40]	317 [46]	262 [35]	281 [39]	365 [49]	283 [39]
Background Time	25 [3]	30 [4]	21 [3]	23 [3]	29 [4]	25 [3]
Total Charged Time	323 [43]	347 [50]	283 [38]	304 [42]	394 [53]	308 [42]
Maintenance Time	53 [7]	59 [8]	45 [8]	45 [6]	59 [8]	55 [8]
Other Time (Idle & Down)	368 [50]	290 [42]	416 [56]	357 [50]	297 [40]	357 [50]
Total Hours/Month	744 [100]	696 [100]	744 [100]	720 [100]	744 [100]	720 [100]

*Base = Total Number of Hours in each Month

†On-line time is the computer time used by people operating from remote terminals. Background time is the computer time used by conventional batch execution of programs.

was devised involving one new CTSS command (BUYTIM) and about a half dozen support programs. The BUYTIM package is available to user groups at Project MAC on an optional basis; only those groups desiring the facility need adopt it.

Besides providing an individual user with the capability of purchasing allocations of computer time and disk storage space with funds budgeted to him, the system also provides a convenient structure for management of user groups. Management functions may occur at any of several distinct levels. The highest level, wherein general budgetary decisions and the decision to add new groups to the CTSS system are made, is reserved for Project MAC Administration. The user group leader can use BUYTIM to: add and delete individual users within individual projects (problem numbers); allocate funds available to his group among individual problem numbers; and establish limits, for each problem number, on the aggregate amount of computer resources that may be obtained, even where funds may be adequate to exceed these limits. The user group leader can also assign some, or all, problem numbers in his group to a common pool of resources, or establish individual pools for each number. By this means, large projects, with several users, have been provided with the ability to manage their own use of computer resources, whereas the smaller problem numbers, consisting of, perhaps, a single user, may have this function performed for them by the group leader.

Where an individual problem number is to assume some role in its management functions, the system provides for several levels of control within the problem number. These include fairly powerful capabilities, from reassignment of funds and change of password, to total restriction against any user allocation changes. Individual users may be provided with the ability to change allocations of other users in the same problem number, and within this may be permitted or denied the ability to change funds allocations, passwords, time, or disk.

It is intended that the experience gained in the design and operation of this allocation system will be applied directly to the operation and management of the Multics system.

BLANK PAGE

ARTIFICIAL INTELLIGENCE

Research on Intelligent Automata

Computational Geometry

A. The SEE Program

B. The Theory of Perceptrons

C. Connectedness and Serial Computation

D. Designing a Stereo Vision System

E. Theorem-Proving Programs

Chess and Game Trees

Mathematical Laboratory

Interactive Computer-Mediated Animation

Fourier Transform Methods in Image Processing

Structure of Atonal Music

Academic Staff

M. Blum
M. Minsky
W. A. Martin
J. Moses
S. Papert

Non-Academic Research Staff

R. J. Abbott
G. W. Baylor
M. D. Beeler
W. Bennett
T. O. Binford
R. W. Gosper
R. Greenblatt
R. W. Henneman
P. Himot
J. T. Holloway
W. R. Noftsker
J. S. Roe
P. Samson
C. Spielman
G. Voyat
J. L. White

Instructors, Research Associates, Research Assistants and Others

R. Baecker
W. T. Beyer
E. Charniak
S. D. Crocker
S. Geffner
A. K. Griffith
A. Guzmán
C. Hewitt
T. L. Jones
L. J. Krakauer
M. J. Lennon
M. E. Manove
G. H. Mitchell
D. N. Perkins, Jr.
C. R. Smith
S. W. Smoliar
M. Speciner
G. J. Sussman
D. L. Waltz

Guests

A. Forte - Yale University
K. Engelmann - MITRE Corporation
R. Silver - MITRE Corporation

Research on Intelligent Automata - Marvin Minsky and Seymour Papert

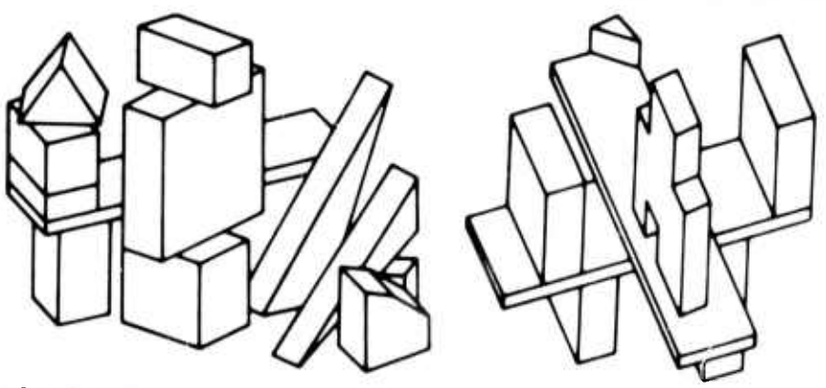
The largest sector of research in our group is still the study of methods for providing machines with greater visual and manipulative capabilities. Our general approach to this goal was described in last year's progress report. We expect, some time in 1969, to demonstrate some practical capabilities of automatic, visually guided manipulation, by showing the computer a structure made of children's blocks, and having it build a functional duplicate. Details of this work are described separately in the Status Reports of the Intelligent Automata Project, and in many of the Artifical Intelligence group memoranda available through our office.

Computational Geometry - Marvin Minsky and Seymour Papert

Preoccupation with theoretical aspects of machine vision has led us to crystallize a general concept we call "computational geometry". This is a new mathematical speciality concerned with the complexity of computations necessary to recognize various properties of geometric objects. The rapid evolution of discoveries in this area has given us hope that it will lead to new directions for "computer science" in general, by providing subject matter that combines intuitive clarity and practical importance, with close connections between classical parts of mathematics — geometry and algebra. We believe the widely recognized conceptual fragility of current "theories" about computation is due in large part to their attempt to build upon excessively abstract principles of automata theory and linguistic structure, without enough concern for thorough understanding of particular problem areas in relation to particular machine structures. The following sections show some examples.

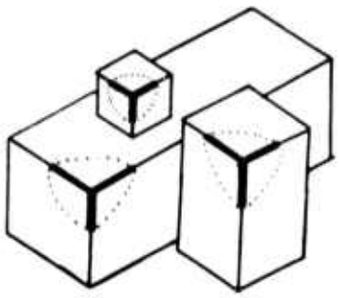
A. THE SEE PROGRAM

Computational requirements lead to geometric questions of an entirely new sort, such as surely never occurred to Euclid. Consider the need to analyze a scene in which some of the objects partially obscure others.



The methods described in last year's report presupposed a computer model or description of each kind of object — cubes, pyramids, etc. Since then it has become clear, in further work of Adolfo Guzmán, that this information is not wholly necessary. Indeed, computations are usually much simpler if based on a more abstract and general theory of the appearances

of objects. Earlier methods were based on partial recognition of the bodies such as the scene below:

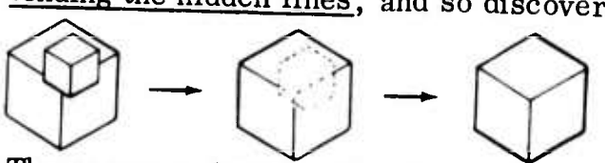


Here, where all objects are rectangular solids, and do not occlude one another badly, we can discover the objects by the extremely local process of locating all the "Y-joints." Each object contains at most one such distinctive feature. This could, of course, fail because of perspective, as in

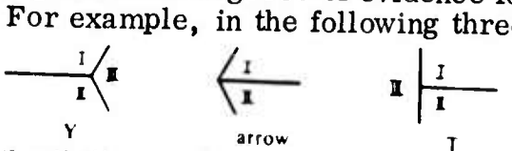
which could be a cube, or in

(for we require each of the three angles of a Y-joint to span less than 180 degrees). A more serious failure is in the case of occlusion, as in

where one of the Y-joints is completely hidden from view. But the great power of programs capable of hierarchical decisions is illustrated by the possibility of first recognizing the small cube, the removing it, then extending the hidden lines, and so discovering the large cube!

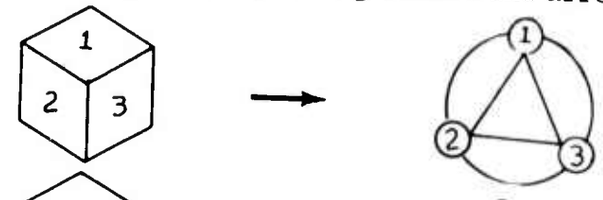


The program developed by Adolfo Guzmán proceeds in a rather different way; his idea is to treat different kinds of local configurations as providing different degrees of evidence for "linking" the faces that meet there. For example, in the following three types of vertex configurations

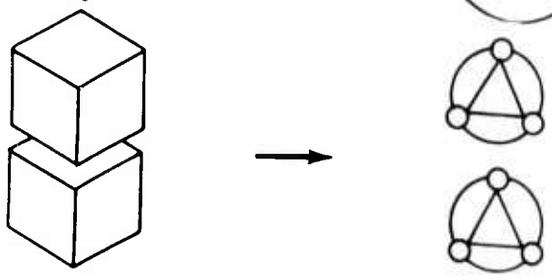


the "Y" provides evidence for linking I to II, II to III, and I to III. The "arrow" just links I to II. Because a "T" is ordinarily the result of one object occluding part of another, it is not regarded as evidence linking I to III, or II to III (and it is also neutral about I and II). Using just these

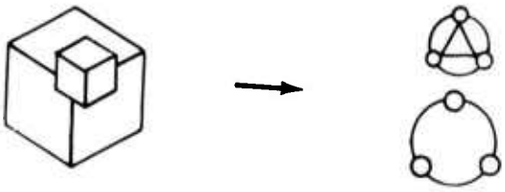
rules, we can convert pictures into associated groups of faces as follows: we represent Y links by straight lines and arrow links by curves.



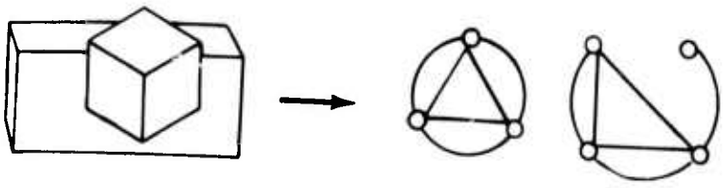
(1)



(2)

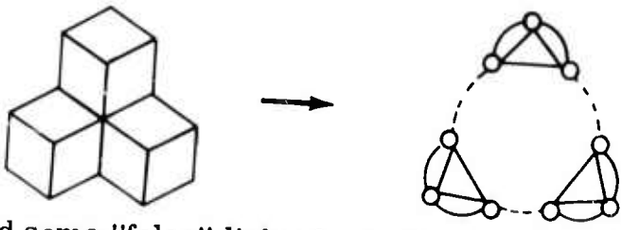


(3)



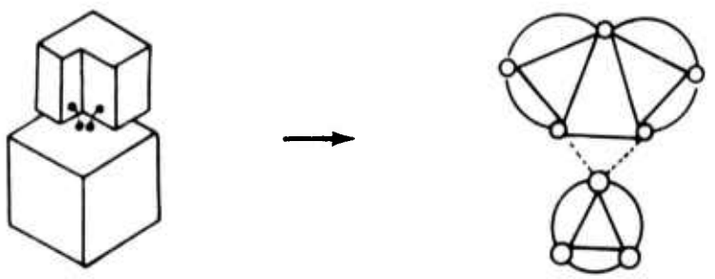
(4)

So far, there has been no difficulty in associating sets of linked faces with objects. The usefulness of the variety of kinds of evidence shows up only in more complicated cases. In the example 5



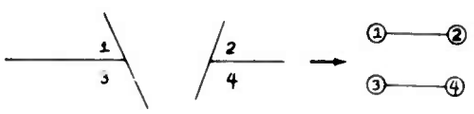
(5)

we find some "false" links due to the merging of vertices on different objects. To break such false connections, the program uses a hierarchical scheme that first finds subsets of faces that are very tightly linked (e.g. by two or more links). These "nuclei" then compete for more weakly linked faces. There is no competition in Examples 1-4, but in 5 the single false links between the cubes are broken by his procedure. In Example 6, if a very simple "competition" algorithm were not adequate here, one could also take into account the negative evidence the two T-joints provide <u>against</u> linking I-III and II-III.



(6)

We have described only the skeleton of his scheme; Guzman uses several other kinds of links, including evidence from collinear T-joint lines of the form



and the effects of some vertices are modified by their associations with others. This variety of resources enables the program to solve complex scenes like that illustrated at the beginning of this section.

Full details will appear in Guzmán's doctoral thesis to appear in the spring of 1969. The surprising power and elegance of this algorithm suggests that there will come a rich theory of the geometry of object clusters. Further evidence for this comes from the discovery of simple heuristics that seem to generate plausible hypotheses about missing lines in pictures of complex scenes.

B. THE THEORY OF PERCEPTRONS

For several years, we have been interested in finding a theoretical basis for assessing abilities and limitations of the Perceptron and similar machines. These are highly parallel computation schemes that attempt to recognize complicated inputs by 1) computing many properties of the input that are relatively easy to recognize, and then 2) basing a decision on some relatively simple combination of the results of the first stage, such as a comparison of weighted sums of evidence for each competing alternative possibility.

Our interest in such machines is not based on very much concern for their practical possibilities, for these are very limited. However, it is our conviction that these machines are nonetheless of critical importance as theoretical models, because if we cannot thoroughly understand such simple computers, we can have little hope of obtaining good theories of more powerful and general computers. (The "theory of computability" for

"universal" machines is totally unsatisfactory in the context of any real practical problems.) Fortunately, we have obtained a wide variety of theoretical conclusions about perceptrons, and these are given in detail in a new book (Perceptrons: an Introduction to Computational Geometry, M. Minsky and S. Papert, M.I.T. Press, 1969).

We will summarize some of the results here. First, let us define a perceptron of order K:

Let R be a part of a two-dimensional plane. Let X be an arbitrary black-and-white "picture" (i.e., a subset of R — any point in X is considered to be black, and the rest of R white). Let Φ be a set $\{\phi, \phi', \phi'', \ldots\}$ of predicates — functions whose values are 0 or 1 — each of which depends on no more than K points. Finally, choose for each ϕ a number α_ϕ and define

$$\psi(X) = 1 \text{ if } \sum_\phi \alpha_\phi \phi(X) \geq 0$$
$$= 0 \text{ if } \sum_\phi \alpha_\phi \phi(X) < 0$$

(This definition includes the concept of "threshold", as in

$$\sum \alpha_\phi \phi(X) \geq \theta$$

if we allow one of the ϕ functions to be a constant.)

Now we ask whether a perceptron can recognize a "pattern". For example is there a perceptron such that $\psi(X) = 1$ when X is a square (or convex, or connected) and $\psi(X) = 0$ when X is not a square (or not convex, or disconnected)? Our analytic methods depend mainly on replacing the geometric concept of a pattern by the algebraic properties of the transformations that preserve the features that concern us. We cannot give a full example of how this is done for geometric concepts, but the following sketch shows how the algebraic theory works in a fairly simple case. Chapter references point to corresponding sections in the book Perceptrons.

Theorem 3.1 (Chapter 3) Informal Version

Suppose the retina R has a finite number of points. Then there is no perceptron $\sum \alpha_\phi \phi(X) > \theta$ that can decide whether or not "the number of points in X is odd" unless at least one of the ϕ's depends on all the points of R.

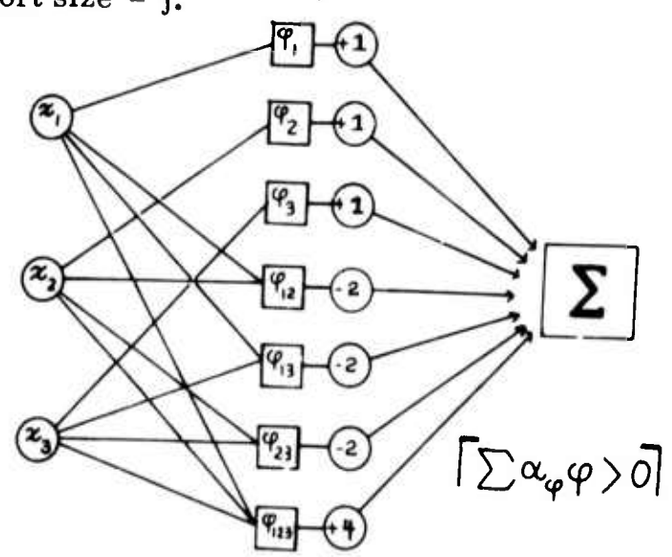
Thus no bound can be placed on the orders of perceptrons that compute this predicate for arbitrarily large retinas. To realize it, a perceptron must start with at least one ϕ that looks at the whole picture! The proof uses several steps.

Step 1: In §1.1- §1.4, we define "perceptron", "order", etc., more precisely, and show that certain details of the definitions can be changed without serious effects.

Step 2: In §1.3 we define the particularly simple ϕ functions called "masks". For each subset A of the retina, define the mask $\phi_A(X)$ to have value 1 if the figure X contains or "covers" all of A, value 0 otherwise. Then we prove the simple but important theorem (§1.5) that if a predicate has order $\geq K$ (see §1.3) in any set of ϕ functions, then there is an equivalent perceptron that uses only masks of size $\geq K$ (see §0.2).

Step 3: To get at the parity — the "odd-even" property — we ask: What rearrangements of the input space R leaves it unaffected? That is, we ask about the group of transformations of the figure that have no effect on the property. This might seem to be an exotic way to approach the problem, but since it seems necessary for the more difficult problems we attack later, it is good first to get used to it in this simple situation. In this case, the group is the whole permutation group on R — the set of all rearrangements of its points.

Step 4: In §2 we show how to use this group to reduce the perceptron to a simple form. The group-invariance theorem proved in §2.2 is used to show that, for the parity perceptron, all masks with the same support size — that is, all those that look at the same number of points — can be given identical coefficients. Let β_j be the weight assigned to all masks that have support size = j.



Group-invariant coefficients for $|R| = 3$ parity predicate.

Step 5: It is then shown (in §3.1) that the parity perceptron can be written in the form

$$\sum_0^k \beta_j \binom{|X|}{j} > 0,$$

where $|X|$ is the number of points in X, k is the largest support size, and $\binom{|X|}{j}$ is the number of subsets of X that have j elements.

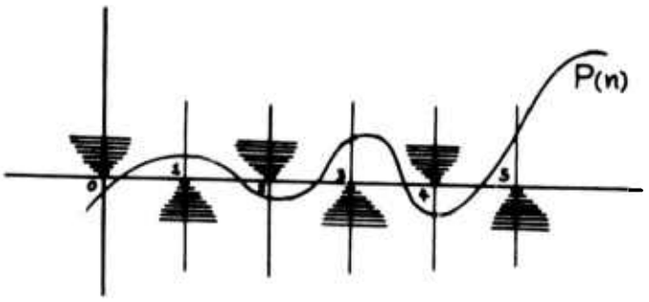
Step 6: Because

$$\binom{n}{j} = \frac{1}{j!}(n)(n-1)\cdots(n-j+1)$$

is a product of j linear terms, it is a polynomial of degree j in n. Therefore we can write our predicate in the form

$$P_k(|X|) > 0,$$

where P_k is a polynomial in $|X|$ of algebraic degree $\leq k$. Now if $|X|$ is an odd number, $P_k(|X|) > 0$, while if $|X|$ is even, $P_k(|X|) \quad 0$. Therefore, in the range $0 \leq |X| \leq |R|$, P_k must change its direction $|R| - 1$ times. But a polynomial must have degree $\geq |R|$ to do that, so we conclude that $k \geq |R|$. This completes the proof.



This shows how the algebra works into our theory. For some of the more difficult theorems we need somewhat more algebra and group theory.

Here are some of the positive results: that certain patterns have certain orders.

Patterns of Order 1

§0.8 "The center of gravity lies to the left of a certain given point on the X-axis."

§2.4 Other similarly defined properties of moments, in fixed coordinate systems. Includes "The area of the image is less than A."

§1.5 Linear threshold inequalities.

§1.4 "The image is exactly a certain one", or "differs from it by not more than a given area A."

Patterns of Order 2

§7.3 "The figure is symmetrical about a fixed point in the plane."

§7.9 "Two figures, on two given lines, are congruent under translation." (The coefficients diverge, however, as the retina size grows.)

§1.6 "The area of the image lies in a certain range."

§6.2 "The figure lies within an axis-parallel line."

§0.8 "The moment of inertia of the figure exceeds some threshold."

Patterns of Order 3

§7.10 "Two figures on two given lines are translation-equivalent with bounded coefficients w_i."

§6.3 "The image is a convex figure."

§6.3 "The image is an axis-parallel rectangle."

§7.7 "The image is a square." (Coefficients diverge)

§6.3 Any two instances of figures not translation-equivalent can be distinguished.

Patterns of Order 4

§7.7 "The image is a square parallel to the axis." (Bounded coefficients)

§6.4 "The image is a circle."

§7.5 "The image has a vertical axis of symmetry." (Coefficients diverge) (Believed to be unrecognizable in any order for bounded coefficients)

§7.4 Reflection symmetry on a line.

§5.8 "The Euler Number — that is, the number of Components minus the number of Holes — exceeds a given number."

Patterns of Order 5 (or possibly one less)

§8.3 "There is a certain number of convex objects." (No holes permitted)

§8.3 "The total curvature of all the boundaries lies in a certain interval."

§7.5 "Two plane figures are equivalent under translation." (Unbounded coefficients)

Patterns of Order 6 (probably)

§7.8 "Two plane figures are equivalent under translation and dilation."

The remarkable thing about the above results is that the order remains fixed, regardless of the size of the retina. But for other patterns, the order increases without bound as the retina is made larger, and it is fair to say that these are, in a practical sense, outside the conceptual ability of perceptron-like machines. These include:

Patterns of Unbounded Order

§3.1 "The number of occupied retinal cells is odd." (The order is known to be equal to the number of points in the retina.)

§5 "The image is a single connected whole." (The order is known to grow faster than $\sqrt{N}$, probably grows as (1/2)N.)

§5.8 All topological properties except simple functions of the Euler Number. For example: "one part of the image lies within a hole inside another part" cannot be recognized.

§4.0 Certain simple Boolean combinations of pairs of first-order patterns. This and §3.1 show strikingly how perceptrons differ from serial computers, for these patterns are very easy for serial machines.

§6.6 Very few of the finite-order properties mentioned above remain finite order in the context of other figures, or of noise.

C. CONNECTEDNESS AND SERIAL COMPUTATION

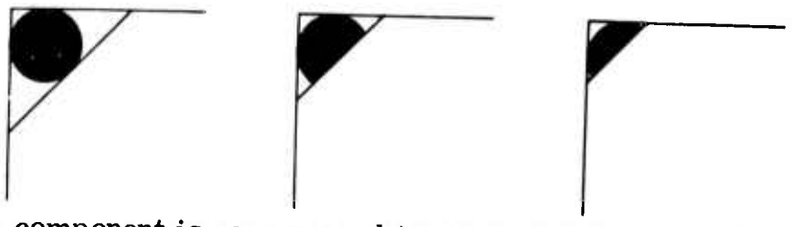
Our deepest results in the Perceptron theory are concerned with the geometric — or rather, topological — properties of connectedness. Because we felt that there was some inherently serial — or recursive — character to connectedness, we decided to investigate the computational geometry of this concept in the context of other mixtures of serial and

parallel machine structures, including Turing Machines and Iterative Arrays. Some of these are discussed in Chapter 9 of Perceptrons. We can give here an example of one such result.

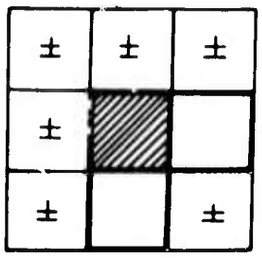
Terry Beyer has investigated the time necessary to compute $\psi_{\text{connected}}$ in a situation that provides a different and perhaps more natural model for parallel geometric procedures. Suppose that each square of a retina contains an automaton able to communicate only with its four neighbors. It can also tell the state (black or white) of its square. The final decision about whether the figure is connected or not is to be made by some fixed automaton, say the one in the top left-hand corner. On the assumption that the states change only at fixed intervals of time, we ask how many time units must pass before the decision can be made. It is obvious that on an n x n retina this will take at least 2n time units, for this is the time required for any information to pass from the bottom right corner to the top left. It is not difficult to design arrays of automata that will make the decision in the order of N time units, where N is the area of the retina. Beyer's remarkable result is that $(2 + \epsilon)\sqrt{N}$ is sufficient, where ϵ can be made as small as one likes by allowing the automaton to have sufficiently many states.

Thus the order of magnitude of time taken by the array is proportional to $\sqrt{|N|}$, which is intermediate between the times taken by the single serial machine (N) and the most general type of parallel computer which is known to take $\leq (\log N)^2$. (Perceptrons, Chapter 9)

The following gives an intuitive picture of Beyer's algorithmic process. The overall effect is that of enclosing a component in a triangle as shown below, and slowly sweeping it into the northwest corner by moving the hypotenuse inward.



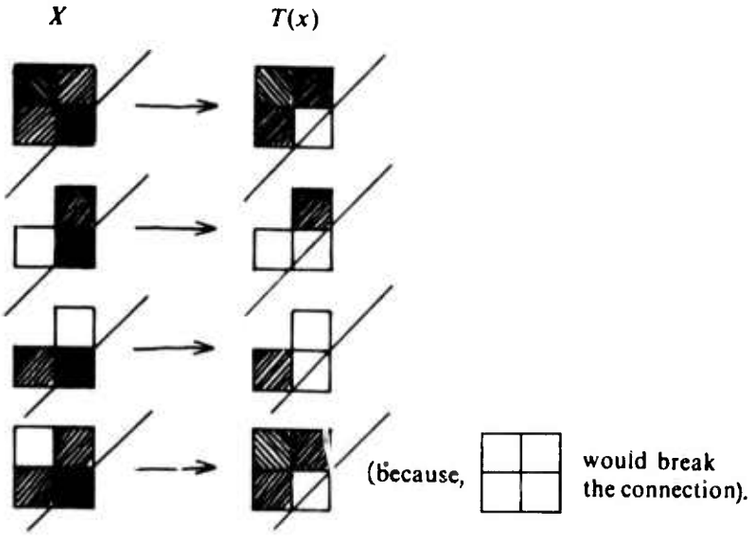
Each component is compressed to one isolated point before vanishing. Whenever this event takes place, it can be recognized locally and the information transmitted through the network to the corner. Thus the connectedness decision is made positively or negatively, depending on whether such an event happens once or more than once. More precisely, the compression process starts by finding the set of all "southeast corners" of the figure.



In this theory the refinal points are taken to be a square checkerboard-like array. Two black squares are connected if they share a common edge, or are in chain of squares so connected. (Diagonal corner-contact does not count as a connection.)

The center square is a southeast corner if the South and East are empty. All other squares shown may be empty or full.

In the compression operation, each SE corner is removed, while inserting a new square when necessary to preserve connectedness as shown below:



The diagonal lines show how repetition of this local process does squeeze the figure to the northwest.

Repeated applications of this operation eventually reduce each component to a single point. The next figure shows how it (narrowly but effectively) avoids merging two components.

It is easy to see that a component within a hole will vanish (and be counted) just in time to allow the surrounding component to collapse down. We do not know any equivalent process for three dimensions. (Consider knots!)

We believe that further investigations of this sort should lead to a deeper understanding of parallel computation in general and of the potential role of such processes in practical artificial and biological visual systems.

D. DESIGNING A STEREO VISION SYSTEM

Another example of computational geometry is the reconstruction of a three-dimensional scene from stereo views. In very simple cases — for example where the scene consists of a single point or line segment — the problem belongs to prospective geometry. But scenes such as those shown in discussing SEE raise different problems. One possible approach is to make an analysis using SEE on each stereo view and then match identified objects. Another "pure" approach is to use the stereo match as a means of object identification. In practice, a real vision system would try to use a mixture of both. In this case a new kind of difficulty arises: that of manipulating data of varied kinds and varied degrees of reliability.

David Perkins has been developing for his Ph.D. thesis a programming system for handling complex net-like data structures that could allow information of diverse sorts to be built up in the course of exploring a scene. The stereo problem is being used as the subject-matter for the system.

E. THEOREM-PROVING PROGRAMS

We are concerned about adapting automatic heuristic deductive programs to problems with large and diverse data bases. Gerald Sussman has programmed a classical theorem-prover based on the resolution principle and has begun to gain experience with the introduction of heuristics of a more specific sort than are usually employed. Carl Hewitt, at the other extreme, has developed a system and language called PLANNER to permit the most diverse possible operations that might be used in proofs. Neither project is yet sufficiently advanced to warrant detailed analysis of results.

Chess and Game-Trees - Richard Greenblatt

During the year, substantial progress was made on Mechanical Chess — a domain that has long been of great interest to workers in Artificial Intelligence. The program, known as MACHACK VI, has achieved the level of ability of an average amateur player and has been

accepted as an honorary member of the American Chess Federation. The clear superiority of this program over all previous ones is interesting both theoretically and as a visible demonstration of programming techniques and systems. (See Greenblatt, Appendix C.)

For a long time we have hoped to see a clarification of the theory of non-terminal Evaluation functions in game-playing, tree-search programs. One's first inclination is to try to interpret the E-function as an approximation to an estimate of probability of success, improved by the look-ahead procedure. No one has been particularly successful at making this interpretation workable. Some promising results on probabilistic strategy theory are given in a recent thesis by David Johnson.

Mathematical Laboratory - William A. Martin and Joel Moses

It has been our long-term goal to develop a man-machine system for amplifying the effectiveness of a mathematician working on either applied or abstract theoretical problems. This project is proceeding in several directions, to bring together such systems as the previously-reported Integration system of J. Moses, the Symbolic Mathematical Laboratory of W. Martin, and the Mathlab of C. Engelman. A two or three-year project is planned, leading to an on-line system to aid in carrying out complex and tedious symbolic calculations, and to build up an active store of knowledge about mathematical algorithms and heuristic methods. We are preparing a monograph on symbolic algebraic manipulations. Martin is pursuing the theory of parsing context-free expressions, both as pure theory and in connection with an on-line parser for hand-written mathematical formulae. Moses is developing a simplification system of advanced power that will, for example, be able to convert

$$\frac{1 + 2 \log (\sin^2 2y + x + \cos^2 2y) + \log^2(1+x)}{\log (1+x) + 1}$$

to

$$1 + \log(1+x).$$

With this, it should be feasible to implement a powerful decision procedure for integration due to Risch; the system should be able to integrate

$$\frac{\log z - 2}{(\log^2 z + z)^2} + \frac{\frac{2}{z} \log z + \frac{1}{z} + 1}{\log^2 z + z}$$

to

$$\frac{-\log z}{\log^2 z + z} + \log (\log^2 z + z)$$

A program, called SARGE, which drills students in freshman calculus integration problems has also been written by Moses. Students are supposed to type step-by-step solutions to an integration problem. The computer checks the correctness of the justification (e.g., substitution of variables) given for each step. Sometimes, though rarely, the computer can give a reasonable description of the type of mistake the student made in an erroneous step.

Interactive Computer-Mediated Animation - Ronald M. Baecker

Standard techniques of man-machine graphical communication have been supplemented with the capability for immediate viewing of a synthesized animation sequence. These were constructed using three special-purpose animation systems which have been implemented with the interactive graphics capability of the Lincoln Laboratory TX-2 computer.

The concept of a table-driven algorithmic synthesis of an animated display has been developed as an approach to the specification of picture dynamics. The tables, called movement descriptions, abstract aspects of behavior that recur over extended intervals of time in a particular animated display. Rhythm descriptions express patterns of the triggering, pacing, coordination, and synchronizing of picture change.

Techniques of control that particularly exploit the interactive environment have been developed. The animator is coupled to the film under construction, both through the sketching and graphical editing of static pictures of dynamic behavior (expressed by movement descriptions), and through the dynamic mimicking of desired behavior. The animator's dynamics are expressed through stylus, push-buttons, and other devices; the language may be used to construct animated visual displays, to build system tools that aid the construction process, and to implement special-purpose animation systems.

Computer time and support for this research have also been provided by the M.I.T. Lincoln Laboratory, under a contract from the Advanced Research Projects Agency.

Fourier Transform Methods in Image Processing - Berthold K. P. Horn

The two-dimensional Fourier transform has been used widely in optics for the evaluation of image-forming systems, yet has been used infrequently in pattern recognition and perception studies. It was thus of interest to find areas of image processing to which the Fourier transform could profitably be applied. Although the Fourier transform has many useful properties that indicate its use in image processing (such as translational invariance) many of these are shared by other functions.

The Fourier transform was found to be useful either in spectral analysis of small image areas, or as a global image-transformer in:

1) focusing an optical device,
2) restoring a degraded picture,
3) resulution beyond the usual limits,
4) edge detection in a preprocessing pass,
5) confirmation of edges in a given position,
6) least-squares filtering the the presence of noise,
7) dynamic range reduction.

No conclusions have been reached about the usefulness of Fourier methods in texture description or curvature detection. A computer program was developed for the PDP-6 to allow experimentation on images read in through the on-line computer "eye" and the Fast Fourier Transform method was used in this program.

Structure of Atonal Music - Allen Forte

Some aspects of this project, which is concerned with a general structural description of so-called atonal music, have been presented in Progress Report III and, more extensively, in MAC-TR-39. (See Appendix D.) During the past year the score-reading program, which parses an input string representing standard music notation, was improved in the direction of greater generality and efficiency, and more effective tools were developed to deal with certain basic problems of musical analysis. In particular, the notion of time-point states has led to a solution to the problem of segmentation (the determination of structural units) and has suggested some useful ways in which relations between such units might be displayed. The task of constructing an appropriate model for this music remains difficult. Work done thus far has suggested several possibilities which are currently being programmed.

BLANK PAGE

COMPUTATION STRUCTURES

Resource Allocation in Multiprocess Computer Systems

Asynchronous Computational Structures

Flow Graph Schemata

Theory of Program Graphs

Asychronous Cooperative Multiprocessing within Multics

A Radical Computer Organization

Phase Structure Grammar for Planar Patterns

Structure Theory of Finite-State Machines

Table-Driven Compiler System

Academic Staff

J. B. Dennis

F. C. Hennie

Z. Kohavi

C. L. Liu

Instructors, Research Associates, Research Assistants and Others

R. A. Carpenter

P. J. Denning

M. Edelberg

J. A. Hamilton

P. Helbalker

D. A. Henderson

F. L. Luconi

S. S. Patil

L. Seligman

D. R. Slutz

D. H. Vanderbilt

C. Ying

Resource Allocation in Multiprocess Computer Systems - Peter J. Denning

The dynamic allocation of limited processor and main memory resources in a user community has been investigaged, as a supply-and-demand problem, in four phases. (See MAC-TR-50, Appendix D.)

First constructed was a model for program behavior; such a model is needed, because one's ability to analyze an entire computer system, and indeed one's philosophy of resource allocation, depends on understanding program behavior. The model constructed — the working set model — is based on locality, the concept being that, during any interval of execution, a program favors a subset of its information; a working set being a dynamic measure of a computation's set of favored information. A working set storage management policy allocates processors to a computation if and only there is enough uncommitted space in main memory to contain its working set. Under such a policy, a computation acquires and releases storage as needed, independently of other computations. Because computations are thus made statistically independent, it has been possible to derive many detailed properties of such policies, both in shared and unshared situations. A problem plaguing many contemporary computer systems is thrashing, i.e., the collapse of system performance due to excessive paging. The working set model provides an explanation for this phenomenon and reveals preventive methods.

In the second phase, the properties of system demand were defined and studied. A computation is regarded as the basic demand-making entity, placing demands jointly on processor and main memory resources. Its system demand is a pair (processor demand, memory demand), where its processor demand represents its immediate processor requirement (intensity and duration), and its memory demand represents its immediate main memory requirement (its working set size).

In the third phase, the properties of dynamic system balance were defined and studied. Computations that demanded resources were segregated into two classes: the standby set, to which the use of system resources is temporarily denied, and the balance set, to which the use of system resources is granted. A system is balanced when the total system demand of the balance set matches the system capacity. A balance policy is one of resource allocation which regulates membership in the balance set so that balance is maintained. Balance policies are formulated as mathematical programming problems whose solutions are found dynamically by the scheduler. Bounds on the system capacity required to implement fair balance policies were found. We have shown that, in computer systems which page on demand from a drum, the scheduler must give top priority to maintaining memory balance. In this relatively special case, the mathematical programming has a simple solution, which is exhibited as an algorithm for the scheduler.

In the fourth phase, these ideas were applied to design and administration of multiprocess systems. An important equation relating hardware configuration (i.e., relative processor and memory capacities) to program behavior and secondary memory access time has been derived. An extension of the working set model provides criteria for managing multilevel memory systems. The need for pooling processor hardware at a fine level of detail was demonstrated as a prerequisite for obtaining the processor capacity that will be needed in future systems. Finally, appropriate measures of system performance were investigated.

In general terms, this work is intended to fulfill three goals. First, it represents a new approach to modeling the behavior of computational processes in complex environs. Second, it provides a general, unified philosophy about resource allocation and sharing. Most importantly, the work is intended to enkindle new thinking about the design, the analysis, and the administration of multiprocess computer systems.

Asynchronous Computational Structures - Fred L. Luconi

Two trends have begun to appear in the design of modern digital equipment: one toward the increased use of modular systems, the other toward concurrent or paprallel processing. The "computational schema" model was developed as a tool for dealing with these trends.

(See MAC-TR-49, Appendix D.) This mathematically formulated schema provides a descriptive discipline in which to represent computational structures.

The theory of systems from which the schema model evolved is not concerned with state behavior, but with activity described by chains of consequences. Each basic operation in a schema is assumed to react independently only to that information in the system to which it has direct access. The result is that computational schemata — unlike conventional programming languages — can be used to represent systems in which several asynchronously communicating processes may proceed concurrently. Also, one does not have to deal directly with total system states that, in practical systems, are unmanageably large in number.

In addition to representational attributes, computational schemata, have interesting mathematical properties. We developed a theory of asynchronous intermodule communication which implies important and useful properties of modular systems. In particular, output-deterministic system behavior was related to conditions on subsystem communication.

The theory of computational schemata has several interesting applications. It provides a basis for a digital system design language. It's a tool in developing methodologies of modular hardware and software design. And most important, it injoins other attempts to create new mathematical concepts in the field of information processing.

Flow Graph Schemata - Donald R. Slutz

A model called a "flow graph schema" — an extension of work by Karp and Miller* — has been formulated and studied extensively. Model components are a set of memory cells, a set of operations, and a control. The memory cells store values, and the operations are used to effect transformations over these values. Each operation is able to read values in certain cells and write values into certain (not necessarily distinct) cells. Sequencing of various reads and writes is governed by a control. The model operates asynchronously and allows for parallel computation. A flow graph schema is an uninterpreted model for a computation in the sense that no specific meaning is associated with its operations or the quantities manipulated. The research covers three principal topics.

The first topic concerns determinacy. A model is determinate if results of a computation depend only on initial values and not on any model timing properties. Necessary and sufficient conditions for determinacy have been worked out, and a decision procedure was devised for testing these conditions in a large class of flow graph schemata.

*Karp and Miller, Parallel Program Schemata, IBM Research Report RC 2053, IBM Research Division, April 1968

The second topic is equivalence of schemata. We have shown that equivalence is generally undecidable, but for a large class of determinate glow graph schemata, equivalence is decidable. The decision procedure involves flow graph schemata that are in a maximum parallel form, which can always be found for flow graph schemata in this class.

The third topic concerns equivalence preserving transformations on the control structure of a flow graph schemata. Sufficient conditions for equivalence were formulated that depend only on the portion of the structure to be transformed. A procedure was devised to test these conditions, and the problem of obtaining a maximum parallel form through a sequence of local transformations was considered.

A brief evaluation of current and future computational systems is made, in terms of the results obtained for flow graph schemata, and a number of interesting extensions of this work are suggested, in a Doctoral thesis to be published as MAC-TR-53.

Theory of Program Graphs - Jack B. Dennis, Fred L. Luconi, and Suhas S. Patil

The studies of Luconi (see this section) and Rodriguez (see Rodriguez, Appendix B) have provided the basis for developing an exposition of computation schemata and program graphs for undergraduate instruction*. This effort has produced a simplified version of the theory that starts from a limited model suitable for the description of digital hardware, and extends this model by stages to encompass features, such as recursion and information structures, essential to the descriptions of algorithms in general. Thus the research carried on by the Computation Structures Group is strongly interacting with the current effort to formulate a quality undergraduate curriculum in Computer Science.

Asynchronous Cooperative Multiprocessing within Multics - Prakash K. Hebalkar

Existing systems do not allow for cooperative multi-processing, i.e., computations consisting of several concurrently operating processes which actively use shared data-bases. If all data references were fully synchronized to avoid conflict, such computation would be possible. Currently, however, such synchronization is nearly impossible. When asynchronous operation of processes is permitted, output-functionality — i.e., the property that results of a computation are reproducible — becomes the chief consideration; but it is difficult to obtain in practical systems.

*J. B. Dennis, et al, notes for M.I.T. Course 6.232 "Computation Structures", Department of Electrical Engineering, 1968

The model for output-functional cooperative multiprocessing developed by Van Horn (see MAC-TR-34, Appendix D) has been used as the basis for a subsystem of Multics that would ensure the output-functionality of multiprocess computations (See Hebalkar, Appendix B.)

Operation of processes on segments is controlled, using the Multics access checking mechanism, so that conflicting data references are constrained to produce unique sequences of values. Procedures for use in this control of access, and for the creation and deletion of processes in a convenient manner, have been defined in detail.

The proposed subsystem ensures that all computations performed within that environment will have reproducible results with minimal restrictions. Multiprocessing in the subsystem requires fairly substantial units of computation by each process to attain efficient performance, as access control is implemented by software means.

A Radical Computer Organization - Jack B. Dennis

In a paper presented at the IFIP Congress 1968*, two trends are cited as being most important in determining the architecture of future large-scale computers. One is the increasing need for computer designers to obtain greater performance through the use of parallelism in computer operations. The other is the increasing dominance of programming costs relative to the cost of computer hardware. The paper argues that these factors will eventually necessitate computer architecture radically different from the forms familiar to us during the first two decades of stored-program machines. In particular, the ability to couple independently written programs together to form larger programs — a system property called "programming generality" — requires that systems implement location-independent means of referencing information. The attainment of efficient use of memory and processing equipment in such systems requires exploitation of parallelism within programs.

*J. B. Dennis, "Programming Generality, Parallelism, and Computer Architecture", Proceedings of IFIP Congress 1968 (Also as Computation Structures Group Memo No. 32, Project MAC, M.I.T., August 1968

The outline of a computer design which could meet these requirements is given in the paper. The design is based on the use of program graphs (which exhibit parallelism in a computation) as the representation of programs, and a general model for hierarchical information structures.

Phase Structure Grammars for Planar Patterns - D. Austin Henderson

The problem of providing a precise specification for classes of two-dimensional symbolic patterns, such as all mathematical expressions like

$$x + y + \frac{\left(\frac{a+b}{c} \quad /ef\right)\left(\int_{z=0}^{1} f(z) \quad dz\right)}{2.04}$$

has been studied. One approach is to define special symbols indicating spatial position, to be inserted between symbol strings to as to transform the problem to one of string syntax, which may be handled by known methods. A similar approach has been suggested by Anderson*, in which the special symbols are replaced by complex positional predicates. In studying character recognition, Narasimhan† used head-tail patterns to define sets of letters, placing them together, according to algorithmic pictorial rules, to generate larger patterns.

A review of these and similar techniques for pattern specification has led to formulation of an approach based on generalizing the notion of phrase-structure grammars to planar or higher-order arrangements of symbols. The syntactic types are now thought of as being n-dimensional aggregates of symbols (two-dimensional patterns in the case of mathematical notation). Positional relations between such aggregates are explicity indicated, in contrast to the implicit left and right juxtaposition relationships in phrase-structure grammars.

*R. H. Anderson, "Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics", presented at ACM Symposium on Interactive Systems for Experimental Applied Mathematics, August 1967

†R. Narasimhan, "Syntax Directed Interpretation of Classes of Pictures", presented at ACM Workship on Programming Languages, August 1965

The problem of pattern specification is recognized as being separable into two related but distinct problems, that of lexical analysis (determining which terminal shapes exist and what their relationships to each other are) and syntactic analysis (collecting terminal symbols and aggregates into higher-level aggregates and identifying their syntactic type). The fundamental form of grammatical rule used is a pattern replacement, analogous to the rule of a context-sensitive grammer. The notation closely parallels the AMBIT/G programming language. Restricted forms and notation were developed for the context-free case, providing a specification language which may be considered the Backus-Nauer Form for two-dimensions. Some notations for specifying highly context-sensitive pattern languages were examined, and a two-dimensional, regular-expression-like notation was developed.

Structure Theory of Finite-State Machines - Chung L. Liu

The structure theory of information transducers was studied. A certain class of information transducers, including all finite-state machines, is characterized by lattice functions describing the transformation of input information to output information. These transducers can be further classified by their characterizing lattice functions being isotone functions, join homomorphisms, meet homomorphisms, and lattice homomorphisms. Corresponding to these lattice functions, algebraic systems known as pair algebras can be defined. The mathematical properties of pair algebras, and their applications to the structure theory of finite-state machines, were investigated.

A problem concerning the structure of stochastic finite-state machines was also studied. A stochastic finite-State machine is said to be definite if its state probability distribution is independent of its initial state probability distribution after a finite number of transition steps. An algorithmic procedure for testing the definiteness of stochastic finite-state machines was found.

Table-Driven Compiler System - James A. Hamilton, Chung L. Liu

Previous work on compiler systems was extended. A generalized translator system was designed and implemented, with efficiency and flexibility as primary objectives. The salient features of the system are: (1 a highly machine-like, table-driven compiler which allows the designer to specify his own translation algorithm in terms of primitive operations which are well matched to the problem; and 2) a table generator which provides a low-level, assembly-type language in which to express these primitive operations. Details of the translator system, together with a working compiler designed as an illustrative example, were described in a Master's thesis by J. A. Hamilton. (See Hamilton, Appendix B.)

COMPUTER SYSTEM RESEARCH

CTSS and Multics System Development

A. Introduction

B. System Integration Benchmarks

C. Performance Improvement

D. Other Efforts

E. Hardware

F. Future Plans

Academic Staff

F. J. Corbató
J. J. Donovan
A. Evans, Jr.
E. L. Glaser
R. M. Graham
J. H. Saltzer

Non-Academic Research Staff

R. H. Campbell
J. H. Cecil
G. F. Clancy
R. C. Daley
S. D. Dunten
M. N. Fateman
R. J. Gardner
G. Garman
C. Marceau
K. J. Martin
E. W. Meyer
N. I. Morris
M. A. Padlipsky
S. Ohayon
R. L. Rappaport
S. L. Rosenbaum
T. P. Skinner
W. H. Southworth
J. W. Spall
M. J. Spier
M. R. Thompson
M. C. Turnquist
T. H. Van Vleck
V. Voydock
D. B. Wagner
M. E. Wantman
S. H. Webber
D. R. Widrig

Instructors, Research Associates, Research Assistants and Others

E. I. Ancona
D. D. Clark
H. Deitel
R. Feiertag
H. Greenbaum
J. M. Grochow
M. Schroeder
R. H. Thomas

Guests

N. Adleman - System Development Corporation
H. J. Hebert - Shell Development Company
C. Mercer - Informatics, for Rome Air Defense Command
A. Sasaki - ElectroTechnical Laboratory of the Japanese Government
P. Schicker - Swiss Federal Institute of Technology
W. R. Strickler - Shell Development Company

CTSS and Multics System Development - Fernando J. Corbató

A. INTRODUCTION

From July 1967 through June 1968 , the Computer System Research Group devoted its attention almost entirely to Multics system development. The Multics (Multiplexed Information and Computing Service) system is a collaborative venture of Project MAC, the General Electric Corporation, and Bell Telephone Laboratories, on the GE 645 computer. As personnel from each group are involved with most of the efforts on the system as a whole, no attempt will be made in this report to separate out one group's contribution to the overall progress of Multics. The only activity of the Computer System Research Group not directly related to Multics — and one which is gratefully acknowledged — was the consultation and supervision of maintenance and accounting activities for CTSS (the Compatible Time-Sharing System) provided by Thomas Van Vleck, in addition to his work on the Multics Project.

At the beginning of the reporting period, the initial Multics system planning and design was essentially complete. Therefore, the primary effort expended during this period was directed toward implementation issues. As discussed below, certain redesign work was performed; but, on the whole, the production and integration of individual modules was the primary concern. "Integration" of the system — that is, causing separately coded modules to perform properly in the system environment — is a major task in any system development. Despite a high degree of planned, functional modularity in the Multics design, integration was still found to be a major effort, since Multics contains a large number of modules, with complex interdependencies.

B. SYSTEM INTEGRATION BENCHMARKS

As related in last year's report, various "benchmarks" have been defined to gauge development progress. The first of these, called "Phase .5", was actually achieved in June of 1967; we recapitulate here, as the Phase .5 system offers the context for the remaining benchmarks. The system's fractional designation was due to the fact that it was a pre-Multics system. That is, the environment in which the system operated was not a true Multics "process" one, but a "pseudo-process" environment which used an adaptation of the GECOS monitor. Interaction was provided through the operator's typewriter on the GE 645. The Phase .5 system comprised the Basic File System of Multics. As this subsystem is considered to be fundamental to Multics, it received the earliest attention in both design and

implementation. The Phase .5 system demonstrated the workability of the Multics notions of segmentation and paging, as well as the workability of the hierarchical file system structure.

By December of 1967, the "Phase 1" system had been demonstrated. Self-initializing from a reel of tape, this system not only operated it, but itself established, a Multics "process" environment. The Phase 1 system comprised one process, with interaction furnished through a standard Multics console. In addition to the Basic File System, early versions of the Input/Output System and the Command System were present. However, the Traffic Controller was not as yet present; hence, the Phase 1 system possessed no multi-programming or multi-processing abilities. The command repertoire was limited to simple commands, primarily dealing with file system functions. Phase 1 was, though, the first "real" Multics system.

An early version of the Traffic Controller was integrated with the Phase 1 system in March of 1968, thus demonstrating the ability to perform multi-programming and multi-processing, and also allow new processes to be created by the working Multics system. Commands were executed by several users working from several consoles. The final major module to be integrated during the reporting period was Access Control, in May of 1968. This module furnishes protection of segments in two distinct senses. First, supervisor segments are partitioned from user segments by a series of "protection rings". Access to supervisor segments is limited to defined points that are "gates" in the protection "walls". As the segments which constitute the Multics supervisor are themselves part of each process, this first form of protection may be thought of as intra-process protection. The second sort of protection which Access Control provides is that of "inter-process" protection; that is, the ability to mark individual segments, whether or not they be members of the supervisor, as being protected from users other than their creators.

By the end of the reporting period, the Demonstrable Initial Multics benchmark was near completion. All of the modules necessary to fulfill the definition of the system (essentially equivalent to the "Initial Multics" benchmark described in last year's report) were available, but further debugging and integration tasks remained to be performed. The performance requirement for this system is the ability to support about eight users doing useful work for a reasonable time (on the order of a few hours) before encountering a system difficulty which must be debugged off-line. Shortly after the Demonstrable Initial Multics benchmark is achieved, a further benchmark, called Limited Initial Multics, is expected to be reached. At this point, final

integration and development work will be performed, essentially under just the Multics system. That is, there will be no further dependency upon CTSS for source file creation and editing. (Under Limited Initial Multics, there will for a time probably be some need for the GECOS environment, for a certain amount of compilation and system tape generation, until the system compiler and assembler are performing efficiently under Multics.)

A version of the system is expected to be available to members of the Project MAC user community, beyond the Multics development workers, early in 1969.

C. PERFORMANCE IMPROVEMENT

In parallel with the integration tasks discussed above, a major effort has been directed at improving the performance of each version of the system as it evolved. (A performance improvement is either a means of furnishing more rapid response to a given user or a means of allowing more users to employ the system simultaneously.) There are two broad categories of performance improvements: redesign of selected modules, and the application of certain optimizing strategies.

The optimizing strategies, in turn, are of two sorts: The first and more obvious is that of improving the compiler for the language in which the bulk of the system is encoded. During the year, the EPL (Early PL/I) compiler was subjected to close scrutiny, and its code generation improved. Further, increased understanding of the compiler on the part of the programming staff allowed for recoding, especially of commands, in order to employ optimal tactics. With the latest version of EPL, an enlightened programmer could reduce program bulk by some 50 percent or more. The importance of improvements to the compiler cannot be overstated: in at least one case, recoding and recompiling led to a reduction to one fifth the previous size. At the same time, a full, optimizing PL/I compiler is under development. When available, this compiler should "automatically" afford considerable further code tightening. The second, less obvious strategy is "binding". This technique all the combining of separately compiled segments into single segments. Binding improves system efficiency in several ways: linkage faults are reduced in number, missing segment faults are reduced in number, and the bookkeeping necessary for maintaining segments in a process is minimized. By the end of the reporting period, a large number of the segments comprising the current version of the system were bound.

The redesign of various system areas proved to be especially fruitful in improving system performance. This is particularly true because initial encoding of a given system portion is often of "first draft" nature; that is, after completion of a system area, the consequencies of a functional specification are fully evident. In the case of a simple module, the normal procedure is for a programmer to iterate the design in his head, trying out alternatives and eventually settling on the choice which works out most effectively in the situation. In the case of a large functional area, however, this iteration is usually beyond the capacity of even gifted programmers. The only recourse then seems to be to try out the design, evaluate the performance, isolate the difficulties and, in a deep sense, understand better the process that is being programmed. In this way the next iteration of design is possible and recoding can commence. Historically, this effect is not unfamiliar: for instance, the initial versions of the Fortran compiler for various machines were quite large and cumbersome, and required rather large-scale development efforts to produce; contemporary Fortran compilers, benefitting from some ten years of iteration, can be surprisingly compact, and are produceable by one- or two-man teams. The point is that the initial implementation of a (functionally) large program tends to be primarily concerned with causing the desired functions to be performed at all; causing them to be performed in an efficient, "tight" fashion is often relegated to the status of a second, discrete step.

The following areas underwent major redesign and/or extensive recoding during the reporting period:

1. The logic required for processing a missing page fault was reduced, from a time consumption of some 90 milliseconds to a time consumption on the order of 15 milliseconds. (Note that the 90 milliseconds figure itself represents a reduction, through recodings of the earlier logic, from a level of some 200 milliseconds.)

2. Several iterations of the typewriter I/O system reduced it from over 40K instructions in length to fewer than 10K instructions.

3. The Shell — the command language interpreter and primary module of the Command System — was reimplemented, reducing its bulk from over 20K instructions to fewer than 3K.

4. A new version of the standard context editor command was produced, of fewer than 3K instructions, a reduction from over 50K instructions.

5. The amount of "wired-down core" — that fraction of memory which must be reserved for the system-wide information and code necessary to process a page fault non-recursively — was reduced by virtue of various redesign efforts from more than 130K to less than 80K. (Also, additional functions are performed at this reduced level which were not performed in the previous version.)

6. The Core Control module — responsible for finding and freeing blocks of memory into which new pages are brought — was redesigned, reducing the number of instructions required from some 24K to about 1K. (Recoding and recompilation of the earlier version had resulted in a 5K version, again demonstrating the gains experienced via EPL improvement.) The Core Control data base was reduced from about 12K words to about 2K words by the redesign. A further redesign is projected which will eliminate Core Control as a separate entity, subsuming its functions under other parts of the File System.

7. The "GIM" — the interface module between the I/O system and the Generalized Input/Output Controller (GIOC) — was reduced from some 65K instructions and data to some 8K instructions and data after one large redesign.

8. The "Drum DIM" — the interface module between the File System and the high-speed drum — went from about 65K words of instruction and data to about 15K as a result of recoding and recompilation, then to some 2K through a redesign. A further redesign is projected which will reduce the total number of instruction and data words to approximately 1K.

9. A redesigned version of the Inter-Process Communications facility requires some 10K, down from over 40K. Further simplifications are expected to result in an IPC of about 4K.

10. The Segment Management Module (SMM) — responsible for locating segments in the file hierarchy by symbolic name, and maintaining the relationship information in regard to

segment names and numbers — underwent drastic rethinking. Originally an independent module of over 30K instructions, with an extensive dedicated data base, the SMM became a portion of the Basic File System comprising some 5K instructions, while the data base was eliminated completely, its functions being included in an existing data base.

11. A global change in system logic was the combining of linkage sections for many segments into single linkage segments. This "combined linkage segment" approach radically reduces the number of segments (and consequent bookkeeping) in a process. An indirect measure of the results of this change was that system initialization time decreased by one half when it was installed.

12. Finally, the amount of time necessary for the system to initialize itself was reduced, by virtue of extensive rethinking and recoding, to approximately 5 minutes from over 45 minutes.

With the exception of the page fault path and system initialization, direct timing figures are unavailable. However, one of the consequences of a paged environment is that space reductions automatically lead to time reductions because of the smaller number of page faults incurred by smaller modules. (Fewer instructions also require less time to execute, of course.) A further advantage which accrues is that with each user process made smaller, the number of multi-programmable processes in a given amount of core increases. Thus, the reductions in bulk cited serve both of the performance improvement goals mentioned earlier: each process expends less time, and more processes can be accommodated.

The implications of the performance improvement efforts undertaken merit further discussion, in the general context of principles of system design and development. Although in practice they are not necessarily separate, let us consider first recoding, then redesign. There are two points to be made about the coding problems. First, despite the fact that the use of a high-order language continues to be very valuable, as discussed in previous reports, it does leave the project somewhat vulnerable to the chosen compiler's inefficiencies. Having a compiler which generates good code early in the history of a project is particularly desirable because it facilitates resolution of the ambiguity as to whether performance is being hampered by the design or by the implementation. Second, even with a good compiler, coding

"style" is a factor which is often not given enough importance. There are frequently alternate ways of encoding the same algorithm, and superior and inferior ways must be identified. The point at issue here, which may be obvious, is that certain encodings invoke more elaborate mechanisms and code constructs from the compiler than do other encodings.

The implications of the redesign work are less obvious. However, it may be observed that even the extremely competent people working on the project experienced difficulties in seeing in advance the full consequences of a given design of a given functional area. Many times, an initial design of a functional area was found to rely too heavily on structural modularity, and the improved design turned out to be a single unit (segment) in place of a number of units (segments). That is, the consequences of a compact design seem to be far easier to grasp. It is possible that the solution to this problem lies in expending far more effort on reviewing and mentally "stepping through" system designs than is typically done. The difficulty with this approach is that it is very hard to evaluate a design without actually using it. Therefore, what seems to be needed is a judicious balance of pre- and post-implementation design iteration.

D. OTHER EFFORTS

In addition to the integration and performance improvement tasks, various other efforts were performed during the year. Foremost among these was continued expansion of the command repertoire, beyond the immediate needs of the various benchmark systems. Development work took place upon numerous compilers, including FORTRAN, FL (a Function Language developed by the General Electric staff), the full PL/I mentioned earlier, BCPL, and SNOBOL. Also, planning was begun for implementing the AED language under Multics.

Several additions were made to EPLBSA, the system assembly language, during the year: multiple location counters were added; relocation bits for binding were implemented; the ability to produce a symbol segment was added; and commented linkage sections are now produced.

The following development tools were implemented: A "tape daemon", which allows magnetic tape input and output under the Demonstrable Initial Multics System; a new version of the "merge editor", which is used for creating input tapes to GECOS for compilation and assembly, for use under Multics; and a new version of the Multics System Tape Generator, also for use under Multics. The

latter two features are especially important to the Limited Initial Multics system, which is to operate independently of CTSS. Also, a Multics Bootload Operating System (MBOS) was produced; this system allows the operations staff to perform consecutive system tests and take dumps when necessary, as well as to record the state of a running system for subsequent restoration.

In the area of documentation, it may be noted that the Multics System Programmers' Manual grew from 373 sections at the beginning of the year to 586 sections by the end of the year, with total pages going from 2,258 to 3,470. In addition, Professor Elliot Organick's Primer for Multics Sub-System Designers has continued, with three chapters already published and three others in advanced draft states. Finally, detailed planning and some initial writing of a Multics Users' Manual began during the year; this manual to be available at the advent of the Operational Initial Multics System as the primary public source of usage information about the system.

Figure 1 oifers a bar chart depicting the growth of the system during the reporting period. The units are pages of source code. By the end of the reporting period, the checked-out code in the system — exclusive of translators — totalled some 350,000 instructions.

E. HARDWARE

As to be expected with prototype hardware, there were some transient instabilities during the year. The only major difficulty encountered was with the magnetic surface of the drum, which had flaked, requiring a replacement unit.

The two-processor configuration at Cambridge was partitioned into a "GECOS machine" and a "Multics machine". The former, which has 128K of core at its disposal, is employed for compilation and assembly runs, as well as for module check-out in the pseudo-process environment. The latter, which has 256K at its disposal, is used for "bootloads" — that is, for testing in the Multics environment by initializing the system from tapes, beginning with a figurative or literal push of the bootload button — and for console sessions in which the current version of the system is used for productive work.

In the fall of 1968, it is anticipated that "Phase B" production model hardware will be brought in.

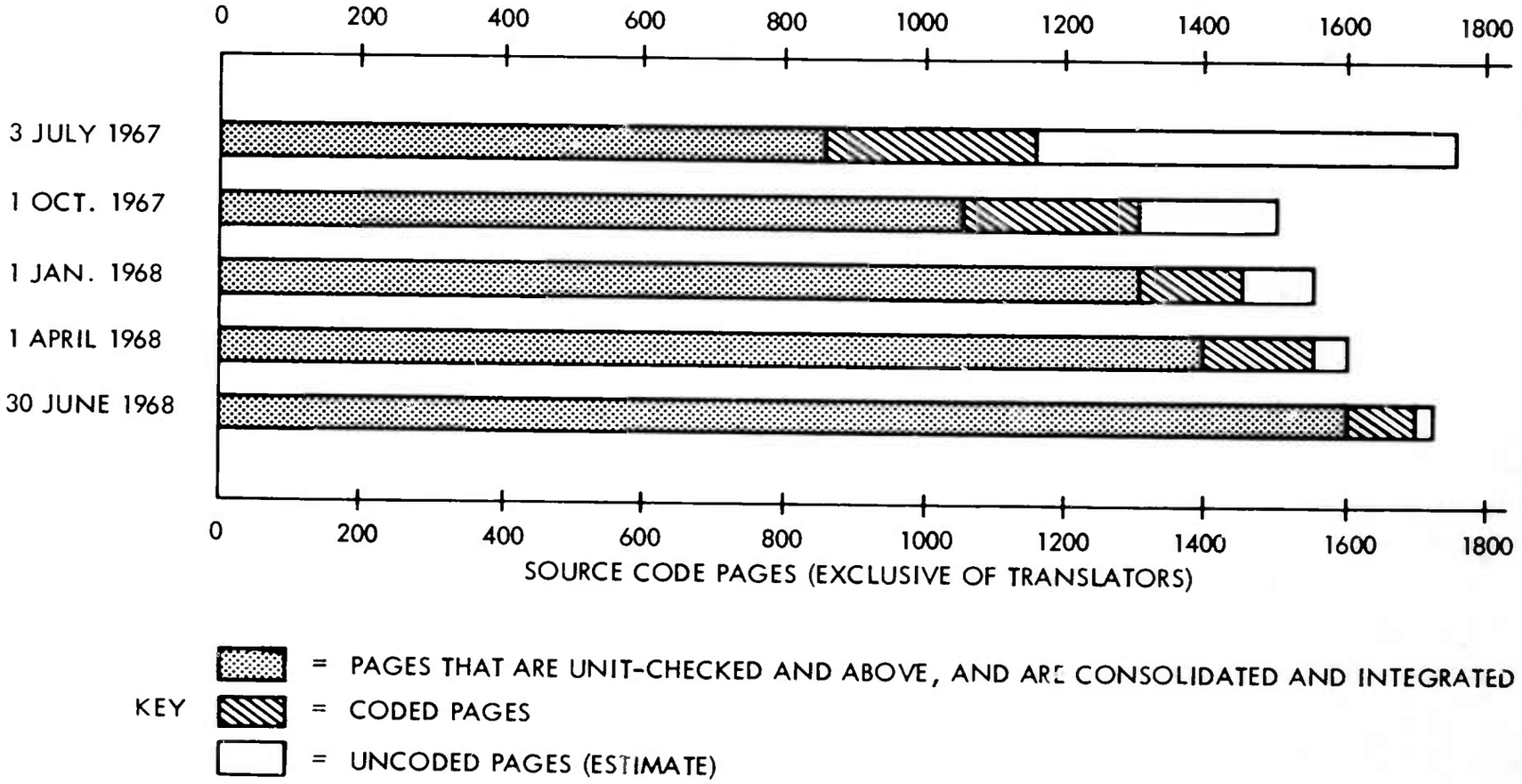


Figure 1. Phase I Multics and Initial Multics Size

F. FUTURE PLANS

It would perhaps be of interest to summarize the future plans discussed previously. Improvement of system performance will continue to be a major goal; important modules of the supervisor for which redesigns are already under development are Page Control and Traffic Control. A second area of performance improvement is the PL/I compiler now being implemented; when available, this compiler will be employed to re-compile the existing system, producing optimized code which in turn will decrease execution time of the system in general. Among the benchmarks, the most imminent is Demonstrable Initial Multics, a version of the system which will permit system programmers to perform productive work (as distinguished from mere system testing) under Multics itself. Next will come Limited Initial Multics, at which point development work becomes independent of CTSS. After that, the remaining dependency on GECOS (for additional compilation and assembly ability) will be removed, and Multics will become available for use by members of the Project MAC user community; at which time it is anticipated that the system's command repertoire will, as CTSS experience dictates, expand rapidly. Both the command repertoire and the supervisor itself will, of course, still be the concern of the system programmers as well.

ELECTRONIC SYSTEMS LABORATORY

Introduction

Computer-Aided Design Project

A. The AED Bootstrapping Process
B. CADET
C. Display Interface System
D. AED Cooperative Program

On-Line Simulation of Networks and Systems

A. CIRCAL-II
B. A Recursive Approach to the Computer Analysis of Nonlinear Networks
C. Tearing Techniques
D. On-Line Simulation of Block-Diagram Systems

Project Intrex

A. The Augmented Catalog
B. Text Access

Display Systems Research

(This work is reported in the next section - Graphics Research.)

Academic Staff

M. L. Dertouzos
L. A. Gould
J. Narud (Visiting)
C. F. J. Overhage
J. F. Reintjes
A. K. Susskind

Non-Academic Research Staff

R. J. Bigelow
M. F. Brescia
T. B. Cheek
R. W. Cornew
C. G. Feldmann
F. B. Hills
P. Johansen
R. L. Kusick
P. Marmarelis
R. C. Nelson
R. B. Polansky
J. E. Rodriguez
D. T. Ross
J. R. Ross
R. H. Stotz
W. D. Stratton
D. E. Thornhill
A. Vezza
J. F. Walsh
J. E. Ward
T. S. Weston
B. L. Wolman
R. V. Zara

Instructors, Research Associates, Research Assistants and Others

A. K. Bhushan
R. H. Bryan
D. G. Chapman
F. DeRemer
R. S. Eanes
M. Edelberg
N. D. Fulton
H. L. Graham
Miss I. G. Greif
K. Hatch
W. Hutchison
W. M. Inglis
G. P. Jessel
M. E. Kaliski
K. P. Polzen
R. G. Rausch
C. L. Reeve
T. L. Smith
J. Stinger
J. R. Sussman
C. W. Therrien
D. Vedder

Guests

F. Bates - Union Carbide Corp
D. J. Cameron - Ferranti Limited
J. T. Doherty - Raytheon Mfg. Co.
R. B. Gluckstern - UNIVAC Div., Sperry Rand
G. L. Lane - Sandia Corp.
R. J. McDowell - Honeywell EDP
R. A. Meyer - IBM Corp.
A. T. Nagai - The Boeing Co.
I. Wenger - Raytheon Mfg. Co.
S. Zurnaciyan - Northrop Corp.

Introduction - John E. Ward

The Project MAC time-sharing system continues to stimulate the research activities of a substantial number of faculty, staff, visiting staff, and graduate students of the Electronic Systems Laboratory. In addition, a number of other graduate and undergraduate students have found opportunities for thesis research in connection with MAC/ESL activities.

During the past year, MAC-related activities in ESL included research in display system technology, programming systems and languages for computer-aided design, computer-aided electrical network design, and library information retrieval (Project Intrex). Display System Research is described beginning on page 69, and the other topics are discussed in this following section.

Part of the display research in the Electronic Systems Laboratory is directly supported by ARPA through Project MAC — other MAC-related research in the Electronic Systems Laboratory is supported by a number of other agencies, including: Air Force Materials Laboratory, WPAFB; the National Aeronautics and Space Administration; the National Science Foundation; the Council on Library Resources, Inc.; the Carnegie Foundation; and the American Newspaper Publishers Association.

Computer-Aided Design Project - Douglas T. Ross

The M.I.T. Computer-Aided Design Project is engaged in a program of research into the application of the concepts and techniques of modern data processing to the design of mechanical parts, as an extension of automatic programming (APT) systems for numerically controlled machine tools. Whereas part programming is a relatively bounded domain which permitted a single, standard APT program and language, the problem of designing large systems such as aircraft is so complex that no one design program or language can be constructed that will serve all the varied needs. In fact, it is clear that a very large number of design languages and programs will be required, each tailored to a specific aspect of the overall design process. Since the time and effort needed to construct each such language and program — and make it available on computers of various types — could equal that of the entire APT development is traditional methods were used, the major effort of the project for the past several years has gone into developing techniques for automating as much of the processes of constructing specialized languages and programs, and moving programs from one computer to another, as possible. The result is the AED (Automated Engineering Design) family of programming systems, including: the AED-1 System, whose domain is general programming,

compiling, and operating of programs on essentially any large-scale computer; the RWORD System, which builds a lexical processor; the AEDJR System, which builds a parsing processor; and the CADET (Computer-Aided Design Experimental Translator) System, aimed at a generalized approach to computer-aided design applications.

The major emphasis during this year has been on the subject of machine independence and the process required to convert the AED System programs to new and basically different computers through an almost completely automatic computer technique called "bootstrapping". Because of the importance of this technique, which has already resulted in availability of the AED System on the IBM 360 and Univac 1108 computers, a brief exposition of the bootstrapping process is presented.

In cooperation with the Graphics Research Group, a first-level program to permit the PDP-7 to act as a buffer between the ESL Display Console and the 7094 time-sharing system was completed and checked out, and research continued into new and better ways to use the two computers interactively.

The AED Cooperative Program, which began in 1964, is aimed at transferral of research results to government and industry. Eight visiting staff members from industry participated in Project activities during the past year, making a total of 32 visitors from 22 companies since 1964. Sixteen outside organizations received tapes and operating information for the initial release of AED for use on the 360 and 1108 computers.

A. THE AED BOOTSTRAPPING PROCESS

Given the AED System operating on one computer, called the Host Computer, the programs which constitute AED are compiled into a new form which will operate on another computer, called the Target Computer. This process is outlined in the following sections.

1. AED Compiler Pieces

The AED Compiler is composed almost entirely of programs written in the AED-O Language. The remaining portion is a minimum number of additional, machine-language programs which handle a few operations that are totally machine dependent and serve to interface the AED Compiler with the machine environment.

Most of the AED-O Language source programs are completely machine-independent; i.e., they deal with operations that must be performed on any computer. Machine-dependent data structures (called "state beads")

describing the Target machine are all defined in a single program segment that is inserted in the machine-independent programs by an INSERT statement. A smaller number of the compiler's AED-O Language source programs deal with machine-dependent aspects of the compilation process. The number of these programs depends upon the special machine characteristics and how "smart" the compiler is in the use of these characteristics.

In addition to the body of the compiler programs, there is a series of packages of procedures which we call the "Support Package". Each portion of the Support Package handles one aspect of the system-building process (dynamic storage allocation, string manipulation, etc.), and the compiler's source programs make frequent use of these packages. In the same way as the compiler source programs, the Support Package is divided into AED-O Language source programs, both machine independent and machine dependent, plus a few machine-language programs.

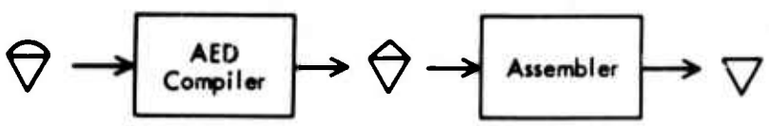
The bootstrapping process is most easily described in terms of a pictorial representation which contains all of the relevant information in a compact form. The following "triangle" symbols are used:


AED-0 Source Program


Mochine Longuage Source Progrom

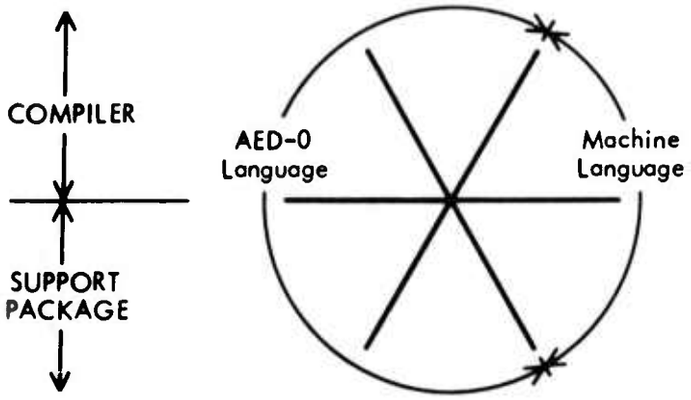

Binary Object Program (result of compiling ond/or assembling one of the other forms)

The current AED Compiler generates character-string output for a format accepted by the existing machine-language Assembler supplied with the Target computer. In this way, detailed concern about binary bit patterns is avoided for the initial bootstrap. Later, similar techniques may be used to eliminate the Assembler step by producing bit strings directly instead of character strings. At present, however, the compilation of an AED-O Language source program into a binary object program is:

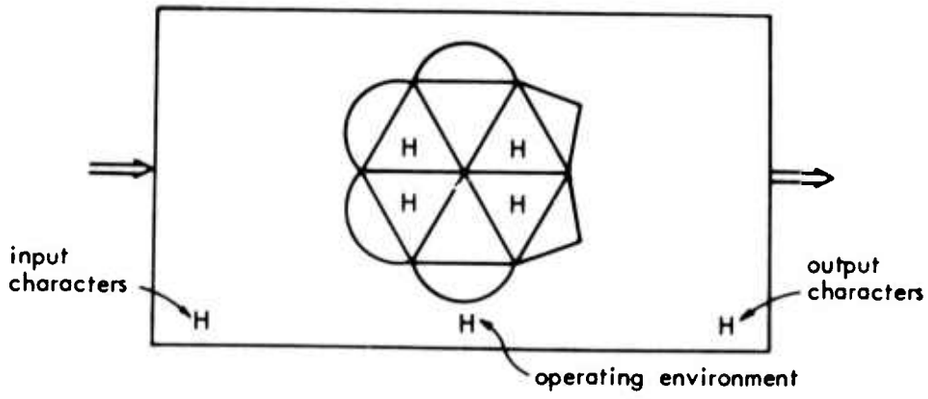


If a program contains machine-dependent statements, we write the letter H or T inside the triangle to indicate Host or Target dependence.

Triangle representation of the six categories of programs of an AED Compiler creates a hexagon, where the orientation of the triangles within the hexagon depicts each program's category. The Support Package occupies the bottom half and the Compiler the top half. AED-0 source programs are shown in the left-side (machine-dependent) and middle (machine independent) sectors and machine-language programs occupy the right-side sectors. The sectors for the graphical notation thus becomes:



By adding the H and T symbols to show machine dependence, and showing the environment via a box around the hexagon, the representation for a Host Compiler is:



Using these conventions, the total process of moving from the Host to the Target Computer is shown in Figure 2. Space does not permit an explanation of this total process, but it is instructive to select one triangle and follow its course through the diagram. Figure 3 shows the path of the machine-dependent AED-O portion of the compiler from the original host version to the final target version.

We see that steps 1 through 4 in Figure 3 prepare a compiler which accepts AED-0 programs in the Host environment and produces output for the Target machine (called the "H-T" compiler). Processing AED-0 programs through this H-T compiler then produces pieces of the desired Target Compiler.

2. Relative Program Sizes

Up to now, nothing specific has been said about relative sizes of the six hexagon pieces, and all six have been drawn the same size. To give a better idea of the extent of true machine independence which has been achieved in the AED Compiler — and of the reprogramming job involved in the process — data for the May 1968 bootstrap to the IBM 360 Computer are shown in Figure 4. The number of 32-bit binary machine words for each of the six sectors of the Target Compiler is shown by a proportionally-drawn arrow through the sector.

Figure 5 shows a breakdown based on programs and tables. This distinction is of interest since tables are set up with little new thinking, whereas programs require creative decisions on the part of the programmer. The machine-independent AED-0 portion requires no reprogramming except for a possible redesign of the state beads. Most of the "tables" are automatically generated by the RWORD and AEDJR Systems on the Host computer in the form of assembler macro calls. Therefore, the only step required to convert the tables for a new machine is to reprogram the macro definitions with Target machine information and process the macro calls through the existing Assembler of the Target computer.

The size of the machine-dependent AED-0 program portion varies considerably depending upon the computer. The 360 is by far the most complex yet used in this regard, so the 7378 program instructions required for the 360 bootstrap should be taken as a maximum.

The portion labeled "Machine Language Programs" in Figure 5 will shortly be reduced by the introduction of the AED version of the ASEMBL output package (presently 1041 words of machine language). Also, some of the number-conversion routines and other of the Support machine language programs are most likely already available in the existing software for any

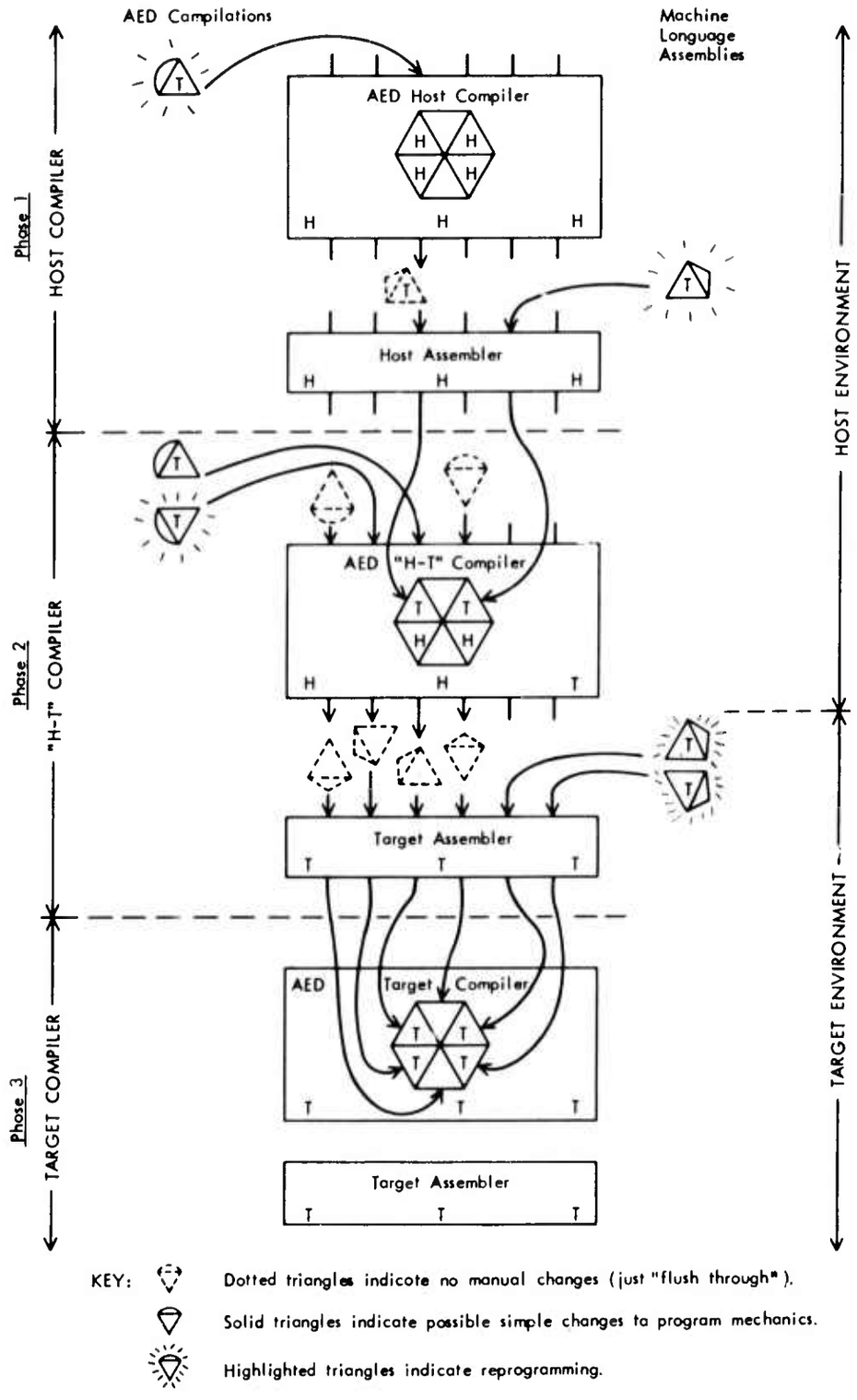
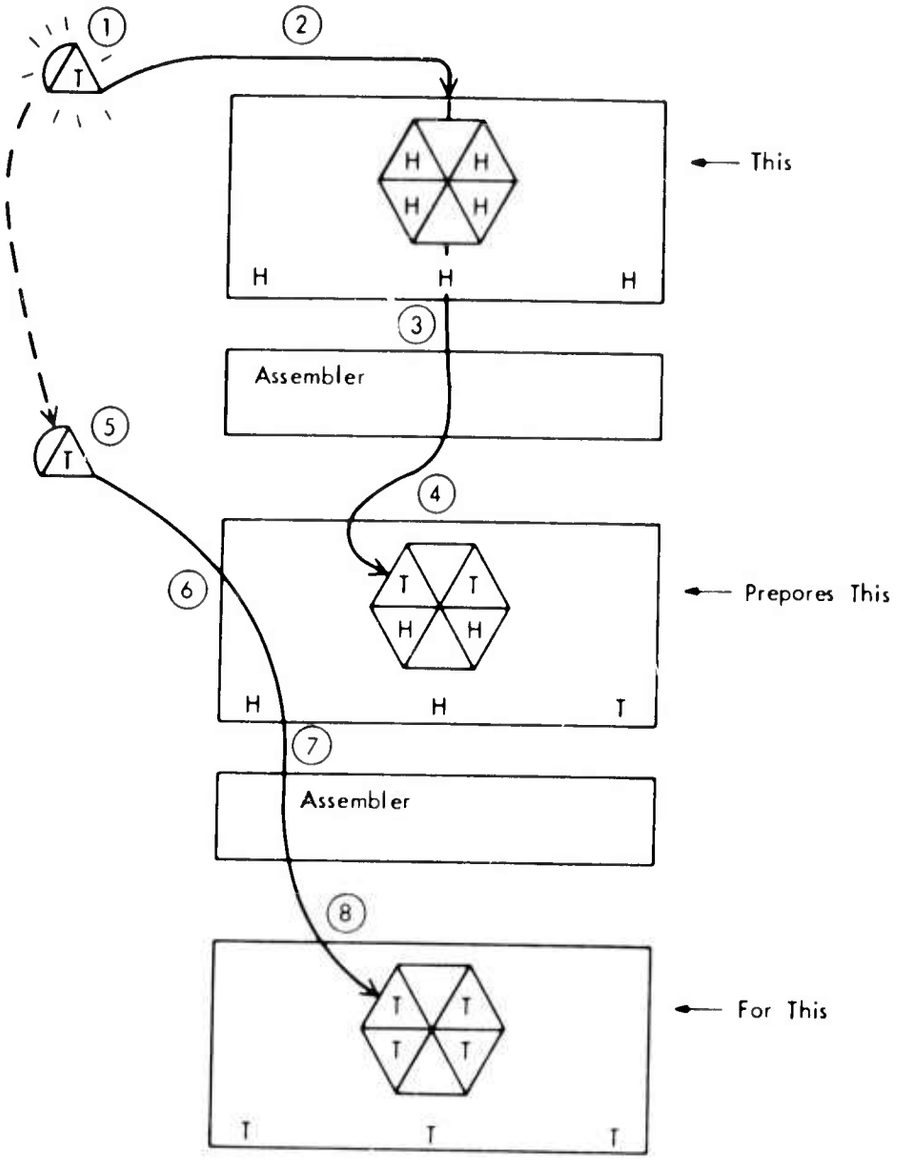


Figure 2. The AED Bootstrap Process



STEP 1: Reprogrom with target information.

STEP 2: Compile resulting programs on Host machine.

STEP 3: Assemble resulting programs on Host machine.

STEP 4: Lood new Host-resident compiler with reprogrommed pieces.

STEP 5: Rework program mechonics for target mochine.

STEP 6: Compile reworked progroms on Host mochine with the compiler generated in STEP 4.

STEP 7: Assemble resulting programs on Torget mochine.

STEP 8: Lood target compiler with reworked programs.

Figure 3. An Example of Conversion Flow

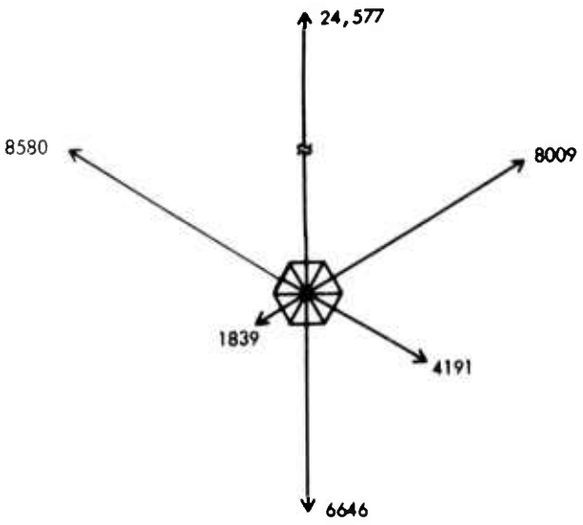


Figure 4. Relative Sizes of 360 Hexagon Pieces
(In 32-bit words)

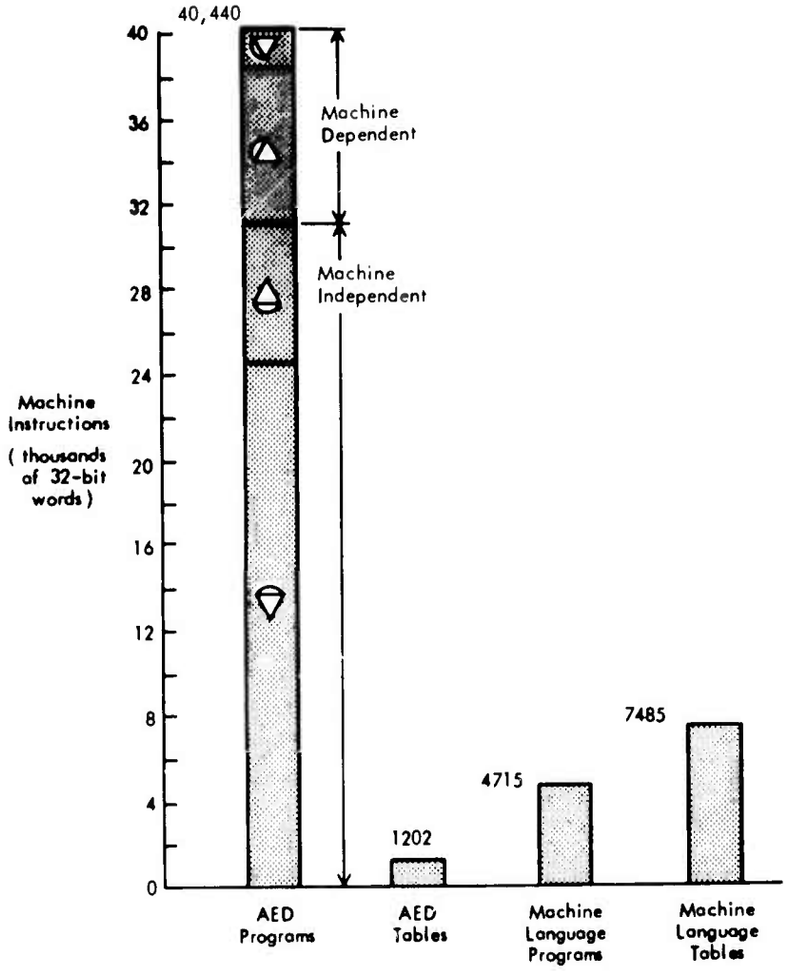


Figure 5. Sizes of 360 Programs and Tables

given Target machine. Thus a considerable part of this section can usually be "stolen" with little reprogramming effort.

3. Special 360 Conversion Improvements

During the process of reworking the old AED-0 Compiler logic to serve as the framework of the machine-independent AED-1 Processor used in the bootstrapping described above, a number of additional benefits were derived. The AED-1 Processor is an essentially new compiler in several important ways which will be of use when new AED-1 Language features are incorporated in the future. In addition, the techniques of using the AED-0 Language and AED Compiler features to express machine-independent programs have been expanded greatly. Most of these techniques may be used with any AED programs to achieve a higher level of true machine-independence than has been achieved before.

The AED-0 Language itself underwent a few minor alterations to provide the basis for more rigorous computer environments in which more control information is needed than was required for the original 7094 version. Such features as POINTER as a distinct data type, and the ability to have procedure components called on the left of an assignment statement, strengthen AED-0 as a general software language.

While the 360-oriented bootstrap was underway at M.I.T., a bootstrap to the Univac 1108 computer was completed by United Aircraft Corporation, starting from source programs supplied by M.I.T. Since these systems differ in various ways, it is planned to re-bootstrap both the 360 and 1108 systems so that a single Compiler is available on both machines. Also, remaining parts of the AED family, such as the RWORD System, the macro processor, etc., will be bootstrapped. A separate effort in cooperation with Project MAC and other groups at M.I.T. will be to bootstrap AED to the GE 645 (Multics time-sharing system). At the same time, the bootstrapping process will be documented carefully so that computer manufacturers or other interested groups may perform other conversions with a minimum involvement on the part of the M.I.T. Project.

B. CADET

The CADET System development continued as a parallel effort to the machine-conversion project. As a first substantial step toward a CADET-1 System, we are attempting to build upon the Polyface Package, first demonstrated at the Second AED Technical Meeting in early 1967.

Although the first version of Polyface used graphically displayed lines to demonstrate manipulation of graphics structures, the system is

intended to provide generalized "modeling" of the structure of arbitrary problems. Building upon the experiences gained from the initial Polyface Package, the following specifications for the second version of Polyface were formulated:

1. The model is fully "common sub-expressioned"; i.e., each distinctly entity occurs only once in the system no matter how often it may be used.

2. The system only keeps track of incremental changes by means of "variation beads", so that there is no redundancy.

3. The complete history of generation of a model is recoverable from the model.

4. Not only the structure of the total model (which may represent several alternate designs) is available, but also any substructure is uniquely isolatable at any time.

5. A generalized mouse algorithm has been devised which leaves no "tracks" in the data structure of the model. That is, the complete state of the mouse is contained within itself so that any number of mice may be running simultaneously over the same model without interference.

6. The encoding of variations so that a mouse knows which variations to obey is done in a very compact optimum binary code which uniquely identifies the precise location of a bead in the structure of the entire model.

7. The beads of the model are successively generated in the natural construction sequence and are unchanged from the time of creation; i.e., there is no necessity to read back old information and make modifications.

8. The structure of the model naturally matches the concepts of zone structure, and the unique mouse path will provide the necessary timing for shuttling large structures in and out of bulk store, when we get to that mode of operation.

Programming of the new Polyface package is well underway, and checkout should begin shortly.

C. DISPLAY INTERFACE SYSTEM

The general approach to the display interface problem is intended to yield a machine-independent, display-independent, problem-independent, and operating-system-independent approach to the coupling of graphic displays to man-machine, problem-solving systems in time-sharing. During the previous reporting period, the basic system for operating the ESL Display Console through the PDP-7 computer (attached to the data channel of the time-shared 7094) was made operational. The system structure uses a minimal executive residing in the PDP-7, augmented by additional PDP-7 programs to suit the user's needs. As a first step in meeting these needs, the PDP-7 was programmed to provide an exact duplicate of the features of the present 7094 module which drives the ESL Console, so that existing display programs will run in buffered mode without change. These programs were debugged during the reporting period and have been in operation for some time.

As the next step, work on the full-scale version of the Display Interface System was launched. Work began at the lowest and highest levels of the system simultaneously; i.e., the Communications Package for interfacing with the hardware was programmed, and work on devising a display language was begun. The Communications Package is now debugged. The package uses a display/program queue to perform two "simultaneous" operations on the same display data, one by the real-time programs and another by the display unit itself. The formulation of a hardware-independent display language has also progressed rapidly. The AEDJR "first-pass structure" appears to fit the problem extremely well, and work on the remainder of the system continues.

This work has been done jointly with the Graphics Research Group, and the reader is referred to page 78 for other details of the display interface work.

D. AED COOPERATIVE PROGRAM

The AED Cooperative Program encompasses those aspects of the overall M.I.T. Computer-Aided Design Project effort which are sufficiently developed to merit industry participation. A major feature has been the sponsorship by industry of visiting staff members who work with the M.I.T. staff learning the capabilities of AED while contributing to its improvement. Past and present participants in the program number 32 people from 22 organizations. During most of the current reporting period, eight visitors were in residence at M.I.T.

Eight copies of the initial 360 version of the AED System (compiler plus AEDJR and subroutine packages) were sent to various companies, even though the initial version of the AED-1 Compiler is incomplete. A more complete version will be ready for distribution shortly, and an even greater demand is anticipated. Several copies of the 1108 AED System were also distributed by Univac. The 360 version was also put on the Cambridge Scientific Center's CP/CMS (360 Time-Sharing System) now available for general distribution to users of the IBM 360 Model 67 computer. AED-0 is now available at M.I.T. on both the Project MAC and IPC (Information Processing Center) 7094 time-sharing systems and on the IBM 360 Model 65 (batch processing) at IPC. AED is used by many research projects and thesis studies and was used in several academic courses in the spring term.

On-Line Simulation of Networks and Systems - Michael L. Dertouzos

The main objective of this research is effective use of present and projected on-line computer utilities in designing electrical networks and systems. This includes studies in the mathematical foundations of computer-oriented network and system analysis, and the interactive features essential for the design of networks and systems. In the case of networks where relationships are primarily implicit, emphasis is placed on developing an integrated on-line circuit-design system, CIRCAL-II, which has evolved from the earlier CIRCAL-I. In the so-called block-diagram systems, where relationships are explicit, an "equivalent" on-line simulator LOTUS-1 is under development, in which analog or digital systems are simulated as compositions of primitive functions and functionals.

A. CIRCAL-II

The first version of CIRCAL-II has been implemented. It is now possible to: create a circuit, consisting of standard elements and nested structures; analyze it, and have the results either plotted or printed; and from this data make changes in either the network topology or some parameter and then repeat analysis.

As shown in Figure 6, the operations of CIRCAL-II may be collected into three groups: 1) basic file system operations, 2) setup of a data structure and subsequent analysis, and 3) output of requested data and modification of the circuit under investigation. An important feature in CIRCAL-II is the standardized data structure, developed in a Master's thesis by James R. Stinger, which acts as a "plug" into which any number of analysis routines may connect. Similarly, the resul's from an analysis are stored in a standard output array, upon which various types of output

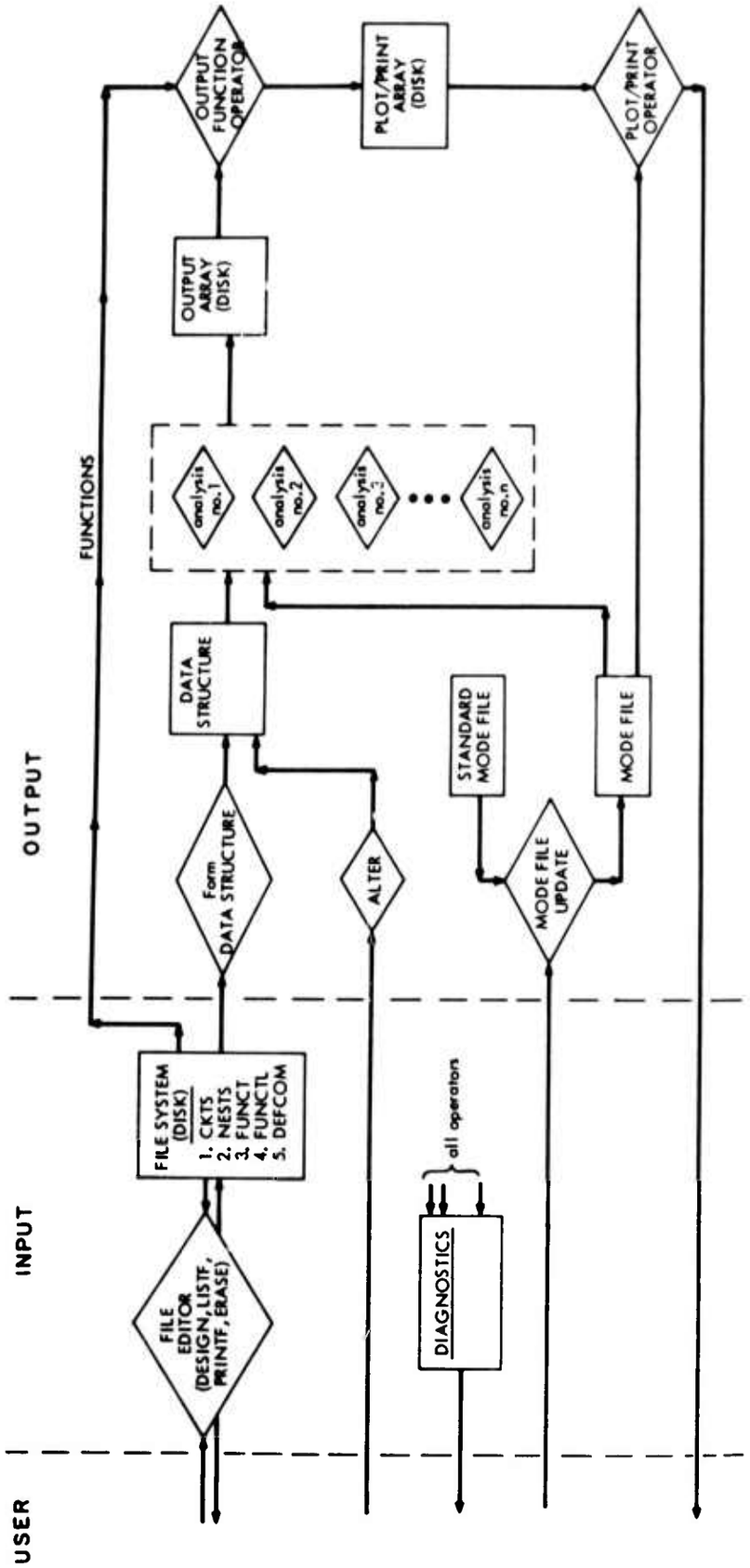


Figure 6. Basic Structure of CIRCAL-II

programs (print, plot, display, etc.) can operate to present results to the user. In addition, all parameters relevant to the output phase of operation are stored in a "mode file". This permits a user to simply indicate incrementally any desired changes and then perform re-analysis, rather than having to each time specify, the entire set of necessary parameters. A global diagnostic procedure is also provided which indicates to the CIRCAL-II user the type of error encountered (out of about 86 types), and whenever possible makes appropriate corrections.

The entire file system of CIRCAL-II is now in operation. Five types of files have been specified: circuits, nested structures, functions, functionals, and defined commands. These files may be created or modified by use of the DESIGN command, which treats all information as a string of characters without regard to its ultimate usage. Several other general housekeeping commands are available. These are: LISTF, which allows a user to list all or a subset of his file directory; PRINTF, which prints out the contents of a file; and ERASE, which deletes one or more of the users' files. The command structure of CIRCAL-II is shown in Figure 7.

CIRCAL-II may employ any one of several analysis subprograms which are stored in CIRCAL's files. The first such program, a basic frequency analysis routine, was written to fully determine the generality of the data structure and the overall modularity of the system. It is significant that only three man-weeks were required to develop this program, as opposed to several man years which were spent on CIRCAL-I. This confirms one of the basic objectives for the design of CIRCAL-II, i.e., the realization of savings by a standard implementation of the large number of common "overhead" programs present in on-line circuit design. Networks of up to 100 elements and 30 nodes, with several levels of nesting, have already been analyzed.

Future work will entail the following steps. First, a number of different analysis routines will be implemented to broaden the scope of CIRCAL-II. These will include linear-time-domain, nonlinear-transient and statistical-sensitivity analysis. Second, CIRCAL-II will be used as a forum for implementing two techniques which are nearing completion of their theoretical stage, as detailed in the sections that follow. These techniques involve the solution of nonlinear networks by tearing and by a new recursive-structure method of analysis. Finally, an area of interest centering on definitional commands will be implemented. These (DEFCOM) features will act as a pseudo-user and will allow operations such as network optimization to be done automatically.

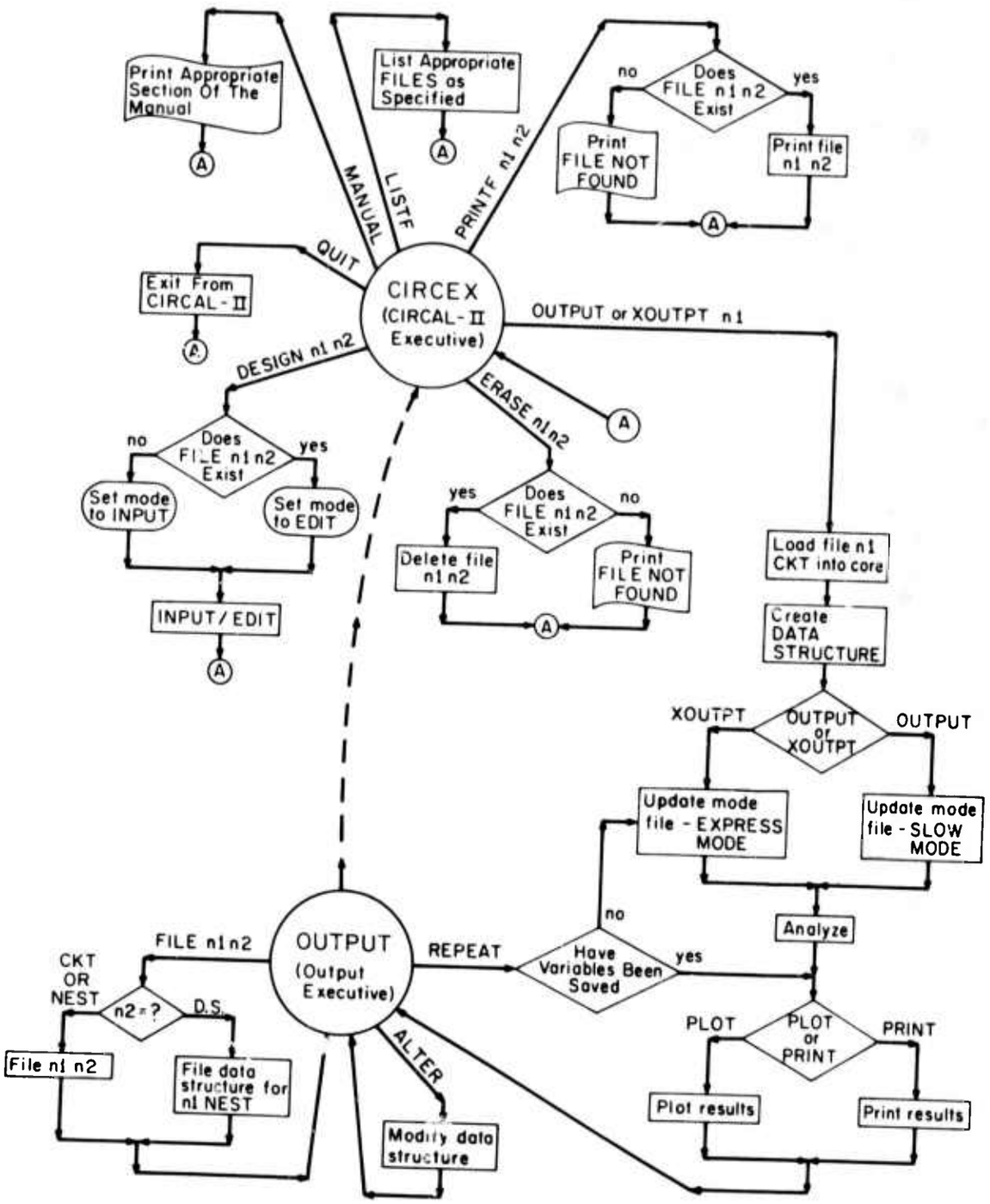


Figure 7. The CIRCAL-II Command Structure

B. RECURSIVE APPROACH FOR COMPUTER ANALYSIS OF NON-LINEAR NETWORKS

The object of this research is to develop methods for inversion of direct functions in a direct non-iterative manner. With such methods, an initial overhead would be expended for construction of the inverse function, but the response to any excitation vector could then be easily evaluated. Previous reports have described the development of an algebra of functions by which the inverse function of a network can be expressed in terms of the basic network-element functions. Function representations obtained in this manner generally contain operations on functions of n-1 variables where n is the number of network nodes. Since the computational effort expended depends directly on the order (number of variables) of the functions being manipulated, practical considerations dictate the need for representation of the lowest possible order. For this reason, the concept of network order has been developed and investigated.

Network order is based on extension to higher dimensions of the familiar class of series-parallel networks, which have an elementary recursive structure providing a direct means of constructing the inverse function. Expressions for the order of specific types of networks have been determined. A result of particular importance shows that the order of a complete graph on n-nodes is the integer q, such that $2/3\,(n-1) \leq q \leq (2/3\,n)$. This, in turn, provides an upper bound for the order of any network on n-nodes. Future research is primarily concerned with improving order-determining techniques, and implementing these theoretical concepts into practical analysis procedures in the CIRCAL-II program.

C. TEARING TECHNIQUES

Tearing is a method for solving networks by splitting them in pieces, solving each of the pieces, and then constructing the solution of the entire network from the solutions of the pieces. A question important to the design of on-line circuit analysis program is, "How should networks be torn so that they may be solved with a relatively small computational effort?" A formal theory of tearing including a tearing model and related algorithms is under development to treat this general question. Part of this work is a Doctoral thesis investigation by Charles W. Therrien.

Tearing of a network is accomplished by separating the network "at some nodes" into a number p of disjoint pieces. The tearing model postulates the existence of certain "computation functions" that provide a measure of such quantities as the number of computer operations required to invert the nodal admittance matrix of a linear network, the

number of iterations required to compute the solution of a nonlinear network by a relaxation algorithm, etc. The computation required to solve an N-node network by the method of tearing can then be expressed in terms of the computation functions. The ratio of this computation to f(N), the computation required to solve the network without tearing, is called the computation ratio C.

Investigation of the computation ratio C has yielded several general properties of 2-tears of N-node networks; for example, a lower bound on the reduction of computation achievable by tearing. However, it does not indicate the best way to tear a network. Several algorithms for locating 2-tears have been tried with reasonable results. A new algorithm now under study has the advantage that there are no degenerate networks for which it fails to find a good 2-tear (unless the network has none). Present work is directed toward speeding up this algorithm and extending its application.

D. ON-LINE SIMULATION OF BLOCK-DIAGRAM SYSTEMS

This work involves the development of a general approach for on-line simulation of a variety of block-diagram systems. These systems may be analog, digital, or hybrid; with memory or memoryless; explicit or implicit (i.e., possessing loops of elements); vector or scalar; or of a more specialized nature, such as dynamic systems with integrators as memory elements; and simulation may be conducted in one or more dimensions, such as time and/or space. An on-line simulation program, LOTUS, has been developed to realize the syntactical and organizational aspects of simulating this wide class of systems.

The present version of LOTUS simulates explicit systems with inputs and outputs defined on the real numbers. A user has the ability to define these systems through an appropriate sequence of on-line commands which interconnect primitive elements and/or nested structures into any configuration in which no two outputs are connected. The program has been written modularly to permit future growth. Work is currently in progress to make the current, limited, version of LOTUS more efficient, and use the fundamentals of system representation to extend the power of the program to handle all systems in the class specified above.

Project Intrex - J. Francis Reintjes

Project Intrex (Information Transfer Experiments) continued its activities during the 1967-1968 academic year under the general direction

of Professor Carl F. J. Overhage, School of Engineering. It is the dual purpose of the project to perform research and experimentation directed toward the design of new library services that might become available in the 1970 decade and to develop competence in the emerging field of information transfer engineering.

Responsibility for the research activities of Intrex resides in the Electronic Systems Laboratory, and the Intrex Group has channeled its efforts towards preparing an experimental augmented catalog system and an experimental text access system with which a selected community of users will interact. Details of the research are contained in the Annual Report of the Electronic Systems Laboratory. A summary is presented below.

A. THE AUGMENTED CATALOG

The purpose of the augmented catalog experiments is to determine — from actual library user response — the types of bibliographic data that should be included in a computer-stored, remotely accessible catalog. Preparation of this catalog is going forward in three groups within ESL — the Catalog Input Group, the Computer Programming Group, and the Console Group.

Catalog entries are prepared off-line on punched-paper-tape equipment, and are then read into the MAC CTSS through the high-speed paper tape reader of the PDP-7 display-buffer computer. As of 30 June 1968, the Catalog Input Group had completed cataloging 5,300 documents in selected areas of materials science and engineering. The initial experimental data base will ultimately contain 10,000 documents.

The Computer Programming Group is engaged in developing storage and retrieval programs for the augmented catalog. The programs are being developed in three phases. Phase I is for the Intrex staff to test and evaluate various techniques of storage and retrieval. Phases II and III are more advanced retrieval programs which will permit broader usage of the data base by an interested sector of the M.I.T. community. The Phase I programming has been completed and is being used to test how well certain of our file organization, storage, and retrieval techniques work. The Phase II system is in the final stages of debugging and should soon be available for our community of users.

The Console Group has designed an experimental augmented-catalog console, and is presently completing the fabrication of console hardware. The console is based on a drum-refreshed alphanumeric display with a screen capacity of 2000 characters at a refresh rate of 60 frames per

second. The character generator consists of a CRT flying-spot scanner which produces a 10-line vertical scan of any of 256 characters on a film mask. This scan-type generation was chosen to produce high-quality characters, and to permit the large character repertoire and special symbols needed for library work. For smaller character sets, a monoscope symbol tube can be substituted for the flying-spot scanner.

A Varian 620I computer serves as a buffer/controller to drive up to 10 consoles from the drum, perform local display interaction based on light-pen and switch inputs, and tie the console system to the main time-sharing system over a dataphone connection. Initially, the connection to CTSS will be at 1200 bits per second, but the system design permits communication rates up to 50,000 bits per second. The display electronics and the digital logic for the console are constructed and are currently undergoing tests, while most of the 620I programs have been written and debugged by simulation.

B. TEXT ACCESS

The research program to create and test a text-access system has been sponsored by the Council on Library Resources, Inc. for the past year. The purpose of the text-access experiments is to evaluate schemes for giving a library user guaranteed rapid access to the full text of documents whether he is at the library or at a location remote from the library.

The first experimental scheme envisions a central store of full text on microfiche. The microfiche will be stored and retrieved using a modified Houston-Fearless CARD retriever with a capacity of 750 microfiche (45,000 pages). The retriever is controlled by the 620I computer, and upon user demand a selected microfiche frame is located and electronically scanned by the flying-spot scanner. A 2000-line scan is performed in one-half second, resulting in a 4.5 MHz video signal transmitted over a coaxial cable to the requester's terminal. Each frame is sent on a single-scan basis, preceded by a digital address code to select the proper one of a number of terminals which share the same transmission line. It is up to the addressed terminal to store the data in electrical or image form for display or permanent record.

Two terminals are presently being prepared for experimentation by the Text Access Group. One is a 35-mm film station employing a high-resolution cathode-ray tube and a camera/processor unit. The video signal received over the coaxial cable will be transformed to a short-duration image on the face of the cathode-ray tube for recording by a 35-mm camera.

The film will be automatically developed in a rapid-processor specially modified for use with the Intrex terminal. The finished film can then be examined by the user on a conventional microfilm reader.

The second terminal will employ a Tektronix Type 611 direct-view storage tube. In this unit the received signal will be converted to a visible image that remains on the face of the storage tube until it is electrically erased by the user. Currently, the quality of this image is marginal, but we expect that it can be improved to yield a display that will be adequate for brief scanning. Thus the user will be able to check the relevancy of a document before requesting a permanent copy.

GRAPHICS RESEARCH

Display Systems Research

A. ESL Display Console

B. ARDS Low-Cost Display

C. Computer-to-Computer Communication

D. Graphic Software

E. Related Display Technology

[Editor's Note: The personnel of this section are a subset of participants from the Electronics Systems Laboratory, and therefore are included in that section's List of Personnel.]

Display Systems Research - John E. Ward

The ESL Display Group has been performing research and development in the field of computer-driven CRT displays and related equipment for several years in support of the ESL Computer-Aided Design Project and of Project MAC at M.I.T. During this time, the group has developed a number of hardware devices, including: the ESL Display Console, which is now connected to the Project MAC 7094 time-sharing system via a PDP-7 buffer computer; a scan-conversion system, to convert computer displays to TV format; and a low-cost remote graphic terminal for time-shared computer systems. The following paragraphs describe progress during the past year in improving these systems, developing graphics system software, developing techniques for terminal and intercomputer communications, developing hard-copy techniques, and planning future display applications for the Project MAC Multics (GE 645) system. (Also, see Kaplow, this volume.)

A. ESL DISPLAY CONSOLE

Previous annual progress reports have discussed the design and construction of a special hardware interface that would permit a PDP-7 computer to be used as a buffer for the ESL Display Console, which formerly was operated directly from a direct data channel of the Project MAC 7094 time-sharing system. To expedite installation of the buffer system, and maintain software compatibility for existing users of the ESL Display Console, the interface was designed to splice the PDP-7 into the data channel connection in such a way that 7094 would act as if it was still "talking" to the display, and the display act as if it was still "talking" to the 7094. The first phase of the system software for this configuration, described in a later section, consisted of moving the display buffer memory and the associated real-time control programs from the 7094 supervisor to the PDP-7, retaining the same software interface for the user programs in the 7094.

The above system became fully operational in September 1967. The immediate benefits were a greatly reduced demand on CTSS time for display operations, a doubling of the amount of memory space available for user display lists, and elimination of the display "blackouts" which had previously occurred during 7094 drum swaps. The PDP-7 has also provided a valuable paper-tape input facility for Project TIP and Project Intrex, whose extensive data bases are primarily prepared on off-line paper-tape equipment. In addition, the PDP-7 tape reader was modified to permit reading of TTS (Teletypesetter) tapes for the newspaper information-processing project.

During the previous year, a second ESL Display Console was acquired by the M.I.T. Computation Center (now the IPC) and operated briefly on a channel basis from their 7094 CTSS. During the present year, it was decided to move this console to the Electronic Systems Laboratory in Building 35, buffer it with a PDP-9 computer acquired by the ESL Computer-Aided Design Project, and tie the PDP-9 to the Project MAC CTSS via a high-speed (50 or 230.4 kbps) telephone link. Because the 7094 possessed no communications adapters for such data speeds, it was decided to enter the Project MAC 7094 through the PDP-7, which already had channel-to-channel communication with the 7094. The main purposes of this tie-in, shown in Figure 8, are to experiment with communication procedures for remote buffered displays, and to provide an interim display capability in the Electronic Systems Laboratory. However, the test configuration was planned with the future in mind. Since the PDP-7 and the PDP-9 are now equipped for telephone-line communication, both displays may later be operated directly from Multics or the IPC 360-67 when these central time-shared facilities become equipped with suitable high-speed communication adapters.

Present status of the second ESL Console is that the PDP-9 console interface hardware, designed by D. Vedder, has been constructed and checked out, and the Type 637 communications adapters for the PDP-7 and PDP-9 have been installed by the Digital Equipment Corporation. The Type 303 Data Sets and connecting telephone line were ordered from the New England Telephone Company, but have been held up by the recent telephone strike. It is hoped that the communications link can be installed in the near future so that tests may begin.

B. ARDS LOW-COST DISPLAY

The computer-buffered, refreshed displays discussed in the preceding section provide a dynamic graphics capability needed for certain types of problems, but the expense of such equipment precludes widespread installation. To provide the average time-sharing system user with a graphics capability requires a low-cost display device which can directly replace the existing teletype terminals. During the past three years, the Display Group has been developing a new type of display terminal, called the Advanced Remote Display Station (ARDS) console, to provide high-speed alphanumerics and full graphical (picture-drawing) capability over an ordinary voice-grade telephone line.

Previous annual reports have described the design and construction of prototype electronics for use with a direct-view storage tube. The advantage of such a storage tube for this application is that no local refresh memory is required, the character and line generators need operate only fast enough to keep up with the incoming data on the

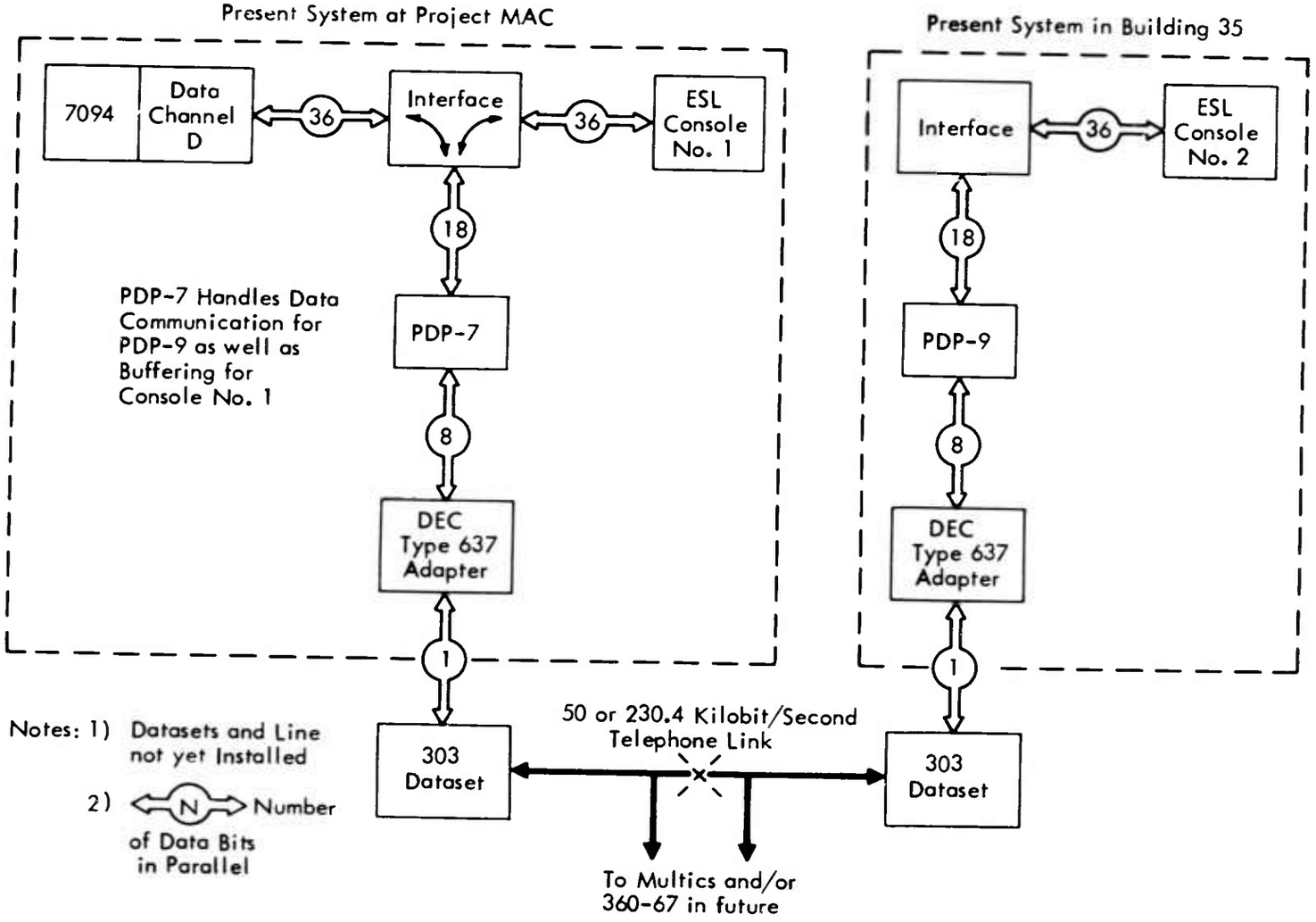


Figure 8. Data Link for Remote PDP-9/ESL Console System

telephone line, and more information can be displayed than on refreshed displays which are limited (by flicker considerations) in the amount they can display in one refresh cycle. The major remaining problem at the start of the present reporting year was that available storage tubes (five-inch) lacked adequate screen size and picture resolution. In August 1967, a new storage-tube monitor became available — the Tektronix Type 611. This new tube has a screen size of 6-1/2 x 8-1/2 inches and has a stored spot size of 0.008 inches. Incorporation of this new tube with the prototype electronics immediately resulted in an outstanding alphanumeric and graphical display capability, as shown in Figure 9.

The ARDS currently operates from CTSS over a switched telephone network at 1200 bit/sec (Type 202 Dataphone) which permits a character rate of 120 per second and a line-drawing rate of 30 to 60 per second. Over 4,000 characters may be displayed simultaneously on the screen, and pictures have no upper limit on the number of line components. For example, the global projection in Figure 9 (programmed by R. Gammill of the Meteorology Department) contains about 2,400 line components (drawing time 40 seconds), and serves as the base for weather displays in which isobars containing several thousand additional line components are added. (See Gammill, this volume.)

In March, 1968, R.H. Stotz and T.B. Cheek, principals in developing the ARDS, left the Display Group and formed a company to place the units on the market. Five units were immediately ordered by Project MAC and other M.I.T. groups for connection to CTSS. The first of these units was received in late June and is shown in Figure 10. The upper part of the table-top ARDS unit is a standard Tektronix Type 611 storage display monitor; the display electronics, Dataphone interface, and power supply are housed in the base section. In the first group of five ARDS, the keyboard is a separate self-contained unit manufactured by the INVAC Corporation. The small hemispherical device shown to the right of the display is a version of the Stanford Research Institute "mouse", a graphical input device which steers an electronically generated, non-storing cursor over the face of the screen. Buttons on the mouse permit transmission of x, y coordinates (or relative line coordinates) to the computer. Several graphical input programs have already been written. One of these permits the input of electrical circuits using the mouse to select elements from a "pick list" display on the screen.

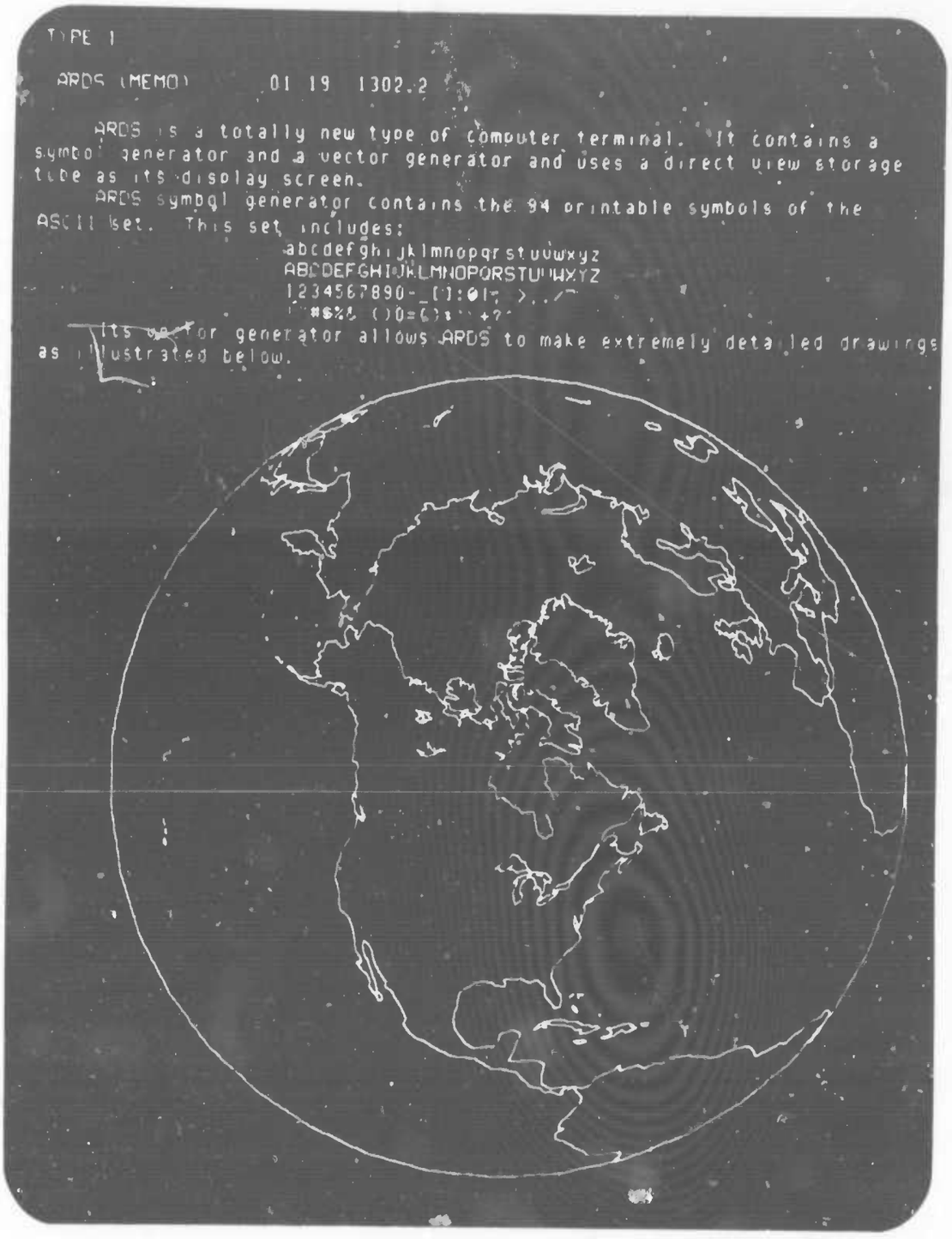


Figure 9. Sample ARDS Display

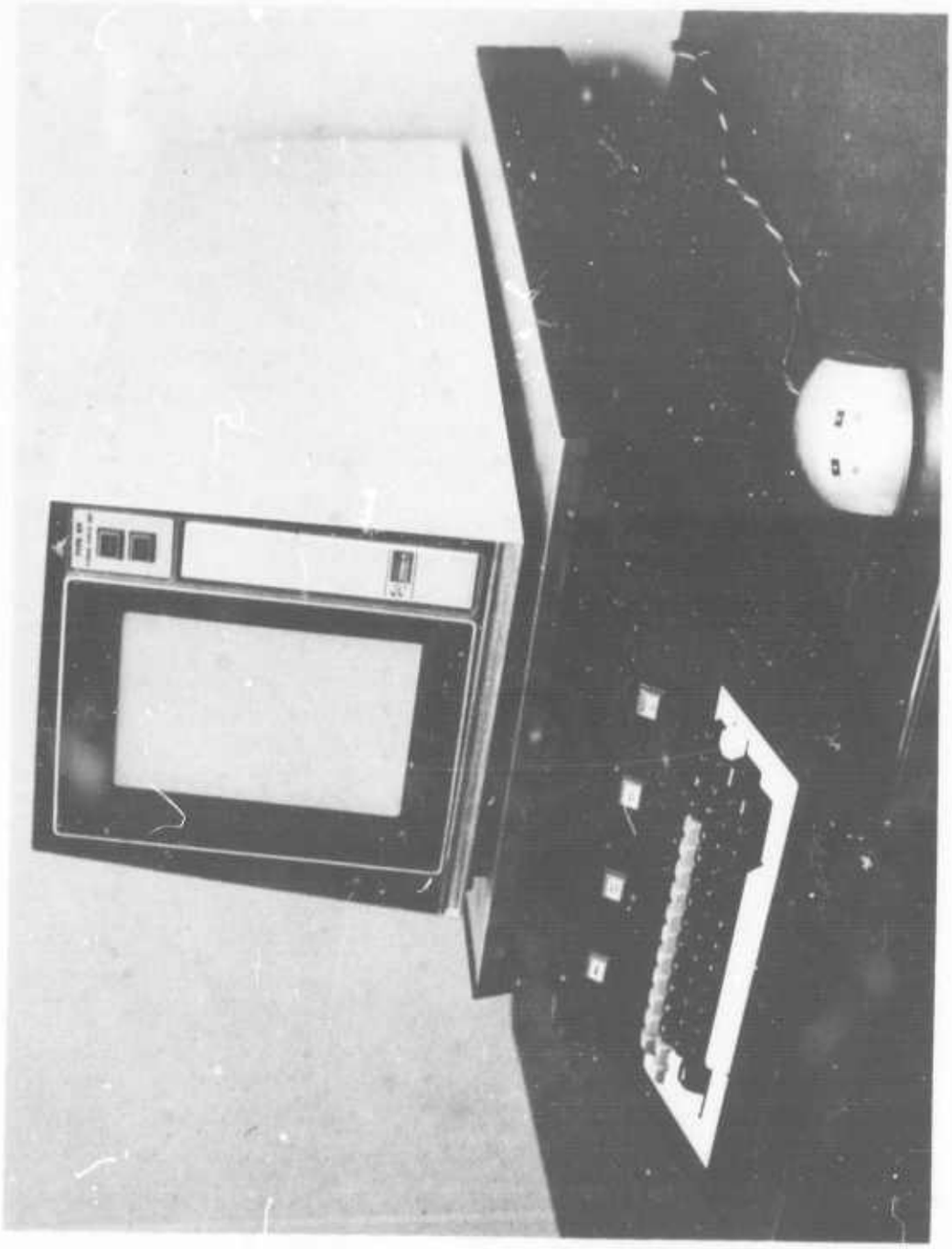


Figure 10. ARDS Display Terminal

At the close of the reporting period, preparations were being made for operation of the initial ARDS unit from West Berlin, Germany, in conjunction with the joint M.I.T.-Technical University of Berlin, Joint Summer Conference "Computers in the University", July 22 through August 3. A 1,200-bit link was being set up via a dial-up connection from West Berlin to Frankfurt, Germany; ITT Datel Service from Frankfurt to New York; and a dedicated AT&T line from New York to the Project MAC computer. As part of this effort, D. Vedder designed and constructed a hardware interface unit to permit ARDS to operate in a half-duplex mode when connected to the 7750 communications controller of the IBM 7094. This has now been established as a standard mode of operations for ARDS. [ARDS was successfully operated for an hour or more each day of the conference.]

M.I.T. Groups which have ordered or are actively considering ordering ARDS units include: Project MAC, the Materials Science and Engineering Center, the Civil Engineering Department, the M.I.T. Library (Project TIP), the Department of City and Regional Planning, the Center for International Studies, and the Sloan School of Management.

C. COMPUTER-TO-COMPUTER COMMUNICATION

With the tie-in of the remote computer-buffered ESL Console discussed in the first section, it became clear that attention should be given to the general problem of computer-to-computer communication. In particular, the establishment of standard character codes, message formats, and communications protocol would do much to improve the present situation, wherein the majority of existing inter-computer communication links have been tailored to meet the idiosyncrasies of I/O devices on individual machines, with no compatibility from one link to another.

A proposed set of standards has been developed, which was described in a MAC memorandum (see MAC-M-351, Appendix A) and in a paper by A.K. Bhushan and R.H. Stotz, presented at the 1968 Spring Joint Computer Conference. These standards, based on the USA Standard Code for Information Interchange (USASCII), permit transmission of binary data as well as text and incorporate the concept of a message header which can contain an address, message identification, and message description. Provision for message acknowledgement and error recovery are included. A.K. Bhushan became a representative on USASI Subcommittee X33 on Data Communications, which is studying these same problems.

The above standards relate to the form and content of messages on communications-type facilities, but say nothing about the facilities themselves. Consideration was also given to the types of links needed within the M.I.T. environment, and it was concluded that if all links (private wire or common carrier) are set up using common signaling standards, which are compatible with the existing common carrier communication facilities, then considerable flexibility can be achieved in communicating between different computers within M.I.T. and between these computers and other computers outside the M.I.T. environment. A set of standards and a "data switching center" was proposed in a Project MAC memorandum, "Recommendations for an Inter-Computer Communication Network for M.I.T." These standards are being used in the PDP-7/PDP-9 link. (See MAC-M-355, Appendix A.)

D. GRAPHIC SOFTWARE

1. Present Display Interface System

The initial programming of the Display Interface System for the buffered ESL Display Console was completed in September 1967. The PDP-7 buffer computer now stores the display file, maintains a picture on the console, performs real-time computations associated with control of the console hardware functions (rotation, translation, etc.), and processes display interrupts (light pen, push button, etc.) for users at two stations; functions which were previously performed by the IBM 7094 CTSS supervisor. This has freed approximately 2500 registers of core storage in the supervisor, and reduced the load on CTSS for driving the display from the previous 3 to 20 percent to a negligible level. Accounting routines have been written for the PDP-7 which enable us to measure the portions of the PDP-7 time employed in maintaining the picture. It was found that only about 3 to 30 percent of the PDP-7's time is used for this purpose, depending on how heavy the real-time action is.

The program packages and data flow between a user program in the 7094 and the display are shown in Figure 11. At the GRAPHSYS 1 (KLULIB) and A-core SDCOPE interfaces, the system looks to the user exactly like the former unbuffered system. The facilities available to the user in the PDP-7 console routines are the same as in the former 7094 routines and are fixed, since the user cannot change them. The main change in the 7094 is the new channel control program.

Also shown in Figure 11 is the data flow for the ARDS displays. The CTSS supervisor program WRFLX has been modified and a new program WSCOPE added to make the ARDS operate in the system very much like a teletype. Various hardware functions available on teletypes, but missing on the ARDS, are simulated by software. Initially, when the ARDS breadboard model had no keyboard or local echo capability, all input

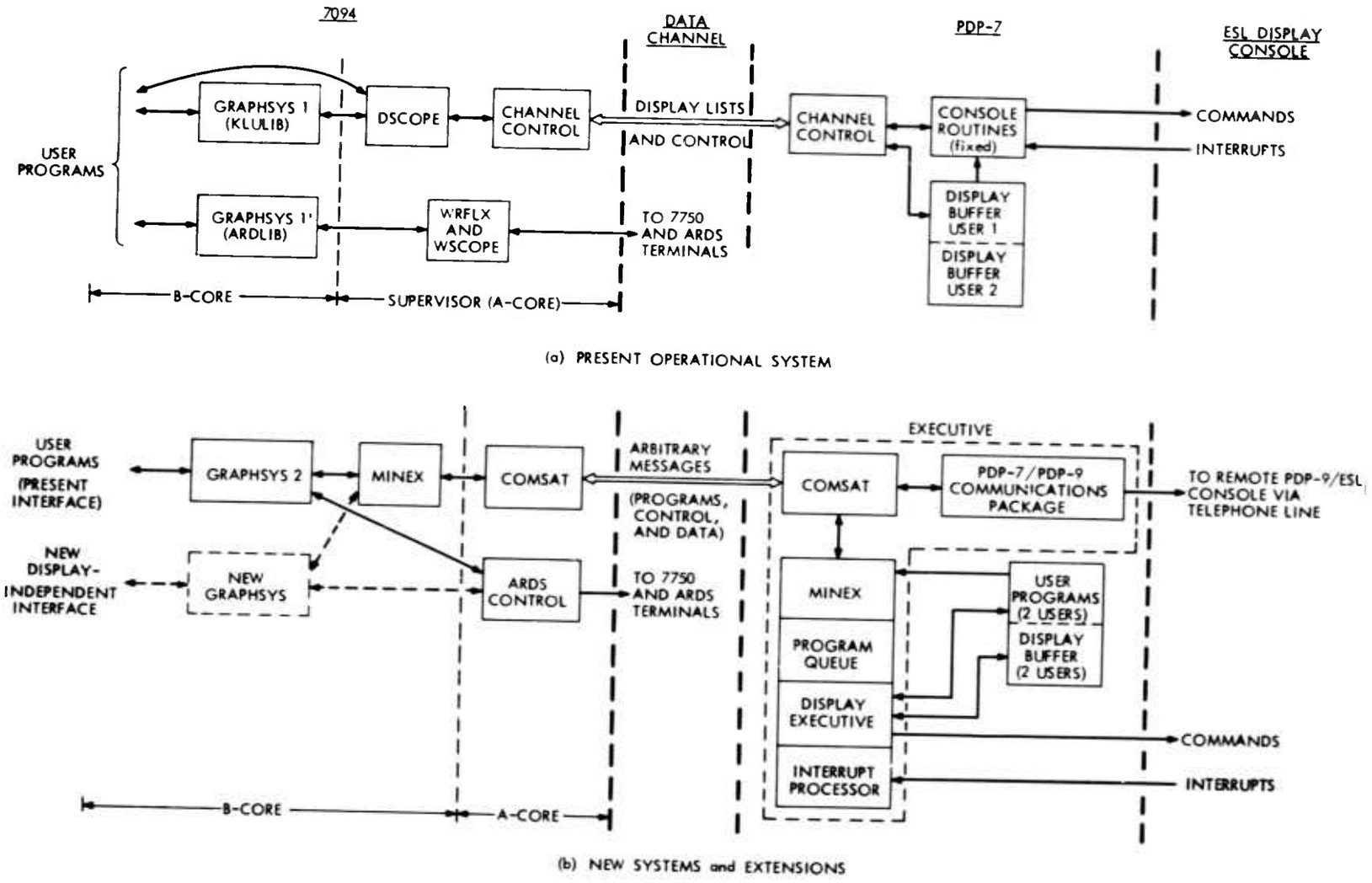


Figure 11. Present and Planned Display Interface Systems

characters were echoed by the CTSS supervisor. As the keyboard, hardware carriage return, and other features were added to the ARDS, the supervisor programs were modified accordingly. additional system programming was provided to handle "bottom-of-page wait" when a user's output reached the bottom of the screen and to allow the user to employ the full graphical drawing capability of the ARDS. When the graphical input devices were added to ARDS, supervisor programs were written to accept this new form of input.

The KLULIB programming system for the ESL Display Console was first modified about a year ago to provide a compatible programming interface to the ARDS, called ARDLIB. This package has been revised and extended during the past year by C. Reeve, in a Master's thesis research program, to add routines which allow the "mouse" graphical input device to be used like a light pen, and routines for defining "light buttons". Other software extensions include a display simulator package and provisions for storing ARDS pictures as CTSS disk files. Changes and additions are now being considered for GRAPHSYS 1, so that the programming interfaces for ARDS and the ESL Console can be made as nearly identical as possible.

2. New Display Interface System

A major effort over the past year has been in the design of the next version of the Display Interface System, shown in Figure 11. The intent here is to provide a single interface for the ARDS and the ESL Console (and perhaps other graphic devices); reduce the supervisor portion of the software to the bare minimum communications functions; and, in the case of the PDP-7/ESL Console, permit users to write their own real-time display routines. To allow users maximum flexibility in allocating PDP-7 resources, the executive program in the PDP-7 consists of a minimal executive (MINEX) which interprets and handles all messages, handles storage allocation, and serves as the interface for user programs in the PDP-7. MINEX may be augmented at user request by standard programs for program queueing, display operation, interrupt processing, etc. The remainder of the memory space allocated to a user may be divided between user programs and display data in any way that the user wishes.

Design of the PDP-7 minimal executive, including a communications package for messages between the 7094 and the PDP-7 and an interrupt handler, has been completed by D.E. Thornhill and H.D. Levin. The augmented executive, including the program queue and general input/output handling routines, is being designed — as is in the display/program queue which will handle the interlock of display and real-time computations

on the display file. All of these programs permit two users at the two console stations to work independently. A first version of the COMSAT package for the 7094 has been written, and work is proceeding toward moving the present DSCOPE supervisor functions into B-core to form GRAPHSYS 2. The general goal in this work, carried out in cooperation with the Computer-Aided Design Project, is to make the whole graphics software system as display independent and machine independent as possible. Toward this end, ideas are being collected for a new user interface, and attention is being given to the problems of moving the whole system to new computers, such as Multics and the S/360-67.

E. RELATED DISPLAY TECHNOLOGY

During the past year, a number of hardware investigations have been concerned with extensions of the present ARDS configuration.

In a Master's thesis research, J.R. Sussman designed and constructed a controller to incorporate a 1200 bit/second incremental magnetic tape recorder into the ARDS breadboard. All characters appearing on the telephone line, whether generated by the computer or the ARDS, are recorded. The controller supplies "page" numbering which is displayed on the ARDS screen, and can search on these numbers by means of commands typed on the ARDS keyboard. Thus the user may request redisplay of recorded pages on an off-line basis, providing a "look-back" capability similar to pulling back the paper on a teletypewriter to examine previously typed input/output. The recording system is also useful for off-line ARDS demonstrations. Further investigations of its use for off-line computer input are planned.

One of the difficulties with data-set-coupled terminal devices is that a delay of up to several weeks is usually incurred in getting the data set installed or moved to a new location. Acoustic and/or inductive couplers for ordinary telephone handsets are widely used for teletypewriter terminals (up to 300 bps), but no handset couplers for 1200 bps are commercially available. D. Chapman, in a Master's thesis research, designed a set of couplers which provide a full-duplex channel at 1200 bps in one direction and 150 bps in the other. Input at each end is acoustic, and pickup at each end is inductive. The ARDS breadboard was successfully operated for a two-week period using these couplers to establish a data circuit between two M.I.T. telephone extensions. Further refinement to fit the new ARDS half-duplex operating conventions is planned.

As discussed earlier, the ARDS terminal design was predicated on telephone-line coupling at data rates of 1200-2400 bits per second. At these data rates, the character and line generators (which operate at an incremental rate of 10 microseconds per point) have considerable dead time waiting for new instructions. In a Bachelor of Science thesis, R.G. Rausch investigated the improvement in display speed obtainable by removing the data-rate limitation and speeding up certain internal logic operations. By driving the ARDS data from the I/O bus of the PDP-7, he demonstrated a 4:1 improvement in drawing speed for characters and up to 8:1 for lines. Further logic changes to obtain an additional factor of 2:1 were worked out but not implemented. These results indicate additional uses for the ARDS as a direct-driven display, as well as growth potential as higher-speed data sets become available.

One of the problems with display-type terminals is that no hard-copy record is produced. A. Vezza is investigating a hard-copy technique in which a full-size photographic paper copy is made directly from the image on the face of the storage tube. Using special wide-angle lenses and new dry-process (heat-developed) dry-silver emulsions which have recently come on the market, it is possible to obtain high-quality finished prints in 20 seconds — 10 seconds for exposure and 10 seconds for development. A prototype hard-copy station for the ARDS breadboard is now being designed and should be ready for trail use by late fall. Since such stations cost little more than an ARDS and operate over a telephone line in exactly the same way, they could be placed at various remote locations convenient to ARDS users.

INTERACTIVE MANAGEMENT SYSTEMS

SIMPLE Project

Marketing Models

Behavior of Complex Systems

Computer-Aided Diagnosis

A Branch and Bound Algorithm for Optimizing with a Simulation Model

Minimization of Machine-Dependent Routines

On-Line Data Analysis

Academic Staff

D. C. Carroll

J. J. Donovan

J. W. Forrester

G. A. Gorry

M. M. Jones

J. D. C. Little

D. N. Ness

M. M. Scott-Morton

C. R. Sprague

Instructors, Research Associates, Research Assistants and Others

J. W. Alsop

D. A. Belfer

J. R. Berdell

J. R. Carbonell

M. Fisher

R. S. Green

N. Hoeg

L. M. Lodish

J. R. Miller, III

A. L. Pugh, III

R. C. Thurber

H.-M. Toong

SIMPLE Project - Malcolm M. Jones, John J. Donovan, Joseph W. Alsop, Hoo-Min Toong, and Richard C. Thurber

The SIMPLE Project was formed in June 1967, for the purpose of designing and implementing a new simulation language. This new language was to differ from existing simulation languages in that 1) it would be designed specifically for use in an on-line environment, and 2) it would incorporate the use of a cathode-ray-tube graphical display device. (See Jones, Donovan, et al., Appendix C, for a discussion of the desirability of incorporating both of these features into a simulation language.) The existing languages were considered unsuitable for these applications. (See MAC-TR-48, Appendix D.) The new language was named SIMPLE

The first phase of the project, which took place during the summer of 1967, was the study of existing simulation languages. It was necessary to become familiar with these languages from the point of view of both the user and the designer, and hopefully be in a position to incorporate just their desirable features into SIMPLE. Thus the summer was spent programming simulations in GPSS, SIMSCRIPT, SIMULA, SOL, etc., and studying the internal workings of the languages.

By September, there was general agreement concerning the characteristics of SIMPLE. Like GPSS, SIMPLE should be easy to learn, so that a user could begin programming with the language after only an hour or two of studying the manual; should offer the computational power of a general algebraic language like FORTRAN or ALGOL; should have extensive tracing and other debugging facilities; and should contain extensive, easy to use statistical and graphical output facilities.

In particular, we decided to adopt the process concept and program structure of SIMULA. It was undesirable to adopt SIMULA directly, however, because it is based on ALGOL, which has never been popular among programmers in the United States. Thus, for a majority of programmers, the new language would be incompatible with any other languages they would normally use. However, PL/I, does contain, among other things, block structures and the other desirable features of ALGOL, and will probably be as well-known as FORTRAN within a few years. Thus we decided to use PL/I as a base language and add SIMULA-like simulation features to it.

We also decided to design as general a language as possible. SIMPLE was to be a super-set of PL/I; suitable for implementation either as an interactive simulation language on a time-shared computer, or as a general simulation language on computers without on-line or graphical

display facilities. In September 1967 it was necessary to select a computer for the initial implementation of SIMPLE. The original plan had been to implement the language on Multics or the M.I.T. Computation Center's version of IBM's TSS, but at this stage it was obvious that neither system would be operational soon enough. Therefore, we decided to put SIMPLE on the IBM 1130 computer, as it was the only computer readily available. (The M.I.T. Computation Center had offered us almost unlimited free time on their 1130 machines.) There are also currently over 2,000 IBM 1130's in the United States — many in universities — thus providing a large potential users' market for SIMPLE.

Because the normal 1130 operating system provided by IBM is strictly for batch-processing, we could not implement the on-line features of SIMPLE. Thus we decided to re-write the operating system so a user could run his programs on-line from a console while batch-processing was going on in the background.

By January 1968, the design of the SIMPLE language and a preliminary design for the 1130 operating system had been completed. The language specifications are documented informally in the "Preliminary SIMPLE Manual".* This manual was prepared as a supplement to a series of lectures given on SIMPLE to a Sloan School class studying simulation.

At this stage, we decided that a preliminary SIMPLE compiler should be given priority over the 1130 time-sharing system. An appropriate subset of SIMPLE for the 1130 was defined and the compiler design initiated. At the same time, Malcolm Jones, John Donovan, and Joseph Alsop prepared a paper entitled "A Graphical Facility for an Interactive Simulation System" for presentation to the 1968 IFIP Congress in Edinburgh, Scotland, 5-10 August 1968.

By June 1968, the SIMPLE compiler was designed† and plans for the summer consisted of beginning work on implementing the compiler.

---

*Thurber, Richard C., "A Preliminary SIMPLE Manual", Internal Paper, Project MAC, 1968

†Alsop, Joseph W., "A Design for the SIMPLE Compiler", Internal Paper, Project MAC, 1968

Marketing Models - John D. C. Little and Leonard M. Lodish

The purpose of this research is to develop marketing models to be used by marketing management in decision making, and to increase knowledge that will aid others to do the same. Of central interest is the exploitation of interactive computational capability. This can be a great help in model construction, but of even greater interest is the design of interactive models to facilitate use by a manager.

A previously reported and published account of an interactive model for a retail store location problem has stimulated at least one firm to work on a similar model for its own purposes. (See Little and Lodish, Appendix C.)

An interactive media planning model, also reported previously, has gone through an extensive redevelopment. The model addresses the problem: given a set of media alternatives, an advertising budget, and various data about the media and the audience to be reached, which alternatives should be used, and when should they be scheduled to maximize a chosen measure of performance? Experience with the model has demonstrated that the interactive capability is important to frequent and effective use. The speed of obtaining answers, the ease of use by non-computer personnel, and the ability to redirect the analysis while it is in progress appear to be critical to the acceptance which the model is rapidly achieving.

Current work is directed to developing market response models which the user parameterizes a priori and then adaptively updates as new data become available. The user is also to have the ability to test out his own and competitive strategies on the model.

Some of this work has been supported in part by the Marketing Science Institute.

Behavior of Complex Systems - Jay W. Forrester

The Project MAC time-sharing system was used during the past year to study the dynamic behavior of complex systems. Complex systems are high-order, multiple-loop, non-linear, and contain both positive and negative feedback. They are to be found in all social systems and are of great importance in understanding the behavior of corporations and larger social groups.

Such systems cannot be solved mathematically. The only approach is experimental investigation of system behavior followed by generalization about the observed modes shown by complex systems.

A number of interesting observations have been made which still require much additional study:

1) Complex systems are counter-intuitive. Their behavior is usually different from that predicted by judgment and intuition which has been developed in the context of simple systems. This explains much of the frustration and futility encountered in the management of complex social systems.

2) Complex systems are remarkable insensitive to changes in most parameter values. A multiple-loop system internally rebalances and compensates for parameter changes which may sometimes be severalfold.

3) Conversely, complex systems have a few influence points through which system behavior can be altered. These are apt to lie at points least likely to be indicated by intuitive inspection.

4) Policy changes within a complex system are apt to lead to short-term responses which are opposite in direction from long-term responses. This is highly misleading, because a program producing improvement in the near future may lead to degradation in the more distant future.

Computer-Aided Diagnosis - G. Anthony Gorry

Research on computer-aided diagnosis has been continued. An extensive investigation of the use of the diagnostic system to detect congenital heart disease has been completed. (See MAC-TR-44, Appendix D, and Gorry and Barnett, Appendix C.) The results were very encouraging. The system performed at a level comparable to that attained by experienced cardiologists. A natural-language version of the heart disease program was implemented using parts of the ELIZA system developed by Professor Joseph Weizenbaum. An experiment to compare protocols of expert cardiologists with those generated by the program is being conducted with Dr. G. Octo Barnett of the Massachusetts General Hospital.

New work in this area is being undertaken with Drs. William B. Schwartz and Jerome P. Kassirer of the New England Medical Center, and Professor Joseph Weizenbaum, to develop combination diagnostic-teaching programs. The intention is to put these programs into actual use in a regional medical information system.

In a different area, an adaptive group-theoretic algorithm for integer programming problems has been devised with Professor Jeremy Shapiro of the Sloan School. (See Gorry and Shapiro, Appendix C.) This algorithm is currently being implemented as a programming system.

A Branch and Bound Algorithm for Optimizing with a Simulation Model - Marshall Fisher

The research reported here is concerned with the problem of optimizing a particular system, by using a simulation model to determine system performance with various settings of system parameters.

The particular case of optimizing the capacity of a job shop is considered. The objective function for the problem is

$$\text{Min } U(N) = C \cdot N + \alpha T(N)$$

where N is a vector whose components are the free parameters of the system and represent the capacity at each work station of the job shop, C is a vector of unit capacity cost for each work station, $C \cdot N$ is the capacity cost of a given solution, α is the cost of tardiness, and T(N) is the average order tardiness associated with a particular value of N. T(N) is a particularly unorthodox function; its value for a given value of N is found by one run of a job-shop simulation model.

A branch and bound algorithm has been developed which finds the particular value of N to minimize this objective function. The algorithm develops a lower bound on the cost of a particular solution using the facts that the machine cost of a solution is known prior to simulation and that T(N) is a monotone non-increasing function of N. The algorithm has been coded in FAP (Fortran Assembly Program) for the 7094 and dubbed SET (Simulation Enumerating Technique).

To provide a reference point for computational tests, a heuristic technique known as the Decentralized Gradient Approach was adopted to this problem and also coded in FAP.

About 20 varied problems have been solved using both approaches. The results indicate that for problems of 5 work stations or less, SET finds optimal solutions which are from 0 to 12 percent lower than the DGA heuristics solution; about 75 percent additional computation time is required for SET. The computing time for SET grows approximately exponentially with problem size, and for problems of 8 stations or more the algorithm becomes prohibitively expensive. Possibilities for combining SET with a heuristic approach to handle larger problems are being discussed.

Minimization of Machine-Dependent Routines - Alexander L. Pugh, III

During the past year, DYMANO II has been modified to conform to the AED-I language and to minimize the number of machine-dependent constructions. These modifications will greatly simplify converting DYNAMO to other machines, such as the GE 645 and the IBM S/360.

The actual conversion to the S/360 is nearly complete, and DYNAMO II users have been invited to try out the system on their machines. The GE 645 version of the system cannot be completed until the AED-I language is available on the 645.

The AED language is a good one in which to write routines that are nearly machine independent. By careful choice of declarations and synonyms, 75 percent of DYNAMO can be maintained as machine-independent AED routines (with separate declaration files for each machine). The 25 percent of DYNAMO that is machine dependent actually generates machine code, and could never be made entirely machine independent.

On-Line Data Analysis - James R. Miller, III

The purpose of this research has been to produce an on-line language, named DATANAL, for data generation and analysis.* It has been

*In addition to Project MAC, several other organizations have provided direct support to the development of DATANAL since 1963. These include the National Aeronautics and Space Administration under Contract number NSG-235 (1963 - 1966); the MITRE Corporation under Contract number AF19(628)5165(1966-1967; the Stanford Graduate School of Business (1968; and the Stanford University Computation Center (1968). Indirect support has also been provided by the Sloan School of Management (1966-1967) and The General Electric Company (1967-1968).

designed primarily to facilitate analysis of externally generated empirical data. However, DATANAL may also be used as a simulation language to generate and then analyze its own data internally. Specifically, DATANAL has been designed to:

1. facilitate analysis of any kind of numerically coded empirical data, generated and collected externally in any research context;

2. alternatively, generate its own data via simulation, and then selectively analyze the results of a simulation run;

3. operate on-line and conversationally between a user and selected portions of whatever data he has collected and/or generated;

4. converse in English or something very close to English; and

5. permit immediate usability by individuals relatively naive with respect to computers and their idiosyncrasies.

DATANAL is essentially a command language which interacts intimately with the CTSS supervisor. It is difficult to classify DATANAL as either an interpretive or a compiler language exclusively, since it performs either or both types of activities, depending upon the nature and extent of whatever computations are requested. When machine instructions are compiled, DATANAL also performs the functions of a loader; e.g., memory allocation, instruction relocation, etc. For these reasons, it is best to think of DATANAL as a self-contained subsystem under loose control of the CTSS supervisor.

Development of DATANAL was initiated late in 1966, the first version being written in MADTRAN and MAD. A second version was completed in June 1967. A third and final version was completed during this reporting period. The final version has been coded entirely in assembly language (FAP) and includes over seventy separate commands for generating and analyzing data. In addition, a user is free to invent his own private commands and to append these to "his" version of DATANAL.

Due to the imminent departure of CTSS from Project MAC, no further development of DATANAL is planned. However, a working version of the entire system has been transferred to the Computation Center for continuing use. (See Miller, Appendix C, and MAC-TR-40, Appendix D, for a detailed description of DATANAL. This latter document also serves as a user's manual.)

BLANK PAGE

MAN-MACHINE COMMUNICATION

The TEACH System

A. Background

B. System Overview

C. Experience with Students

D. Pedagogic Problems

E. Economic Problems

F. Evaluation

Academic Staff

R. R. Fenichel

J. Weizenbaum

Non-Academic Research Staff

S. A. Ward

Instructors, Research Assistants, and Other Students

J. M. Nead

Guests

S. Lorch	-	Massachusetts General Hospital
M. T. McGuire	-	Massachusetts General Hospital
G. A. Miller	-	Harvard University
G. C. Quarton	-	Massachusetts General Hospital
P. J. Stone	-	Harvard University

The TEACH System - Joseph Weizenbaum and Robert R. Fenichel

A. BACKGROUND

One of the major educational problems confronting the nation's colleges and universities is that of providing instruction in elementary computer programming. The problem is important, because an ever-increasing number of academic disciplines are pervaded by computer-based techniques, and Computer Science itself is an increasingly popular study. The world of practical affairs outside the university is also making more and more use of the computer. The overall demand for people who know how to program digital computers is therefore growing very rapidly. Already, programmers are a scarce national resource.

The problem is difficult, because programming cannot be effectively learned without considerable practice. This means that the student must be given access to a computer. Even more important, the student's work must be coordinated with the lessons to which he is exposed and must somehow be supervised. These requirements, when met in the form of ordinary classroom instruction, generate a large drain on institutional staff resources. Precisely because talented computer people are in extremely short supply, most educational institutions, even those with large financial resources, find it either impossible or nearly impossible to meet even their self-determined goals.

M.I.T.'s large course in elementary computer programming, for example, is the largest course at the Institute. In the 1967-68 academic year, when there were 950 freshmen, there were 959 students in this large elementary programming course. In addition, several departments offered small elementary courses of their own.

In that same academic year, the large basic programming course required at least a part-time commitment from each of 14 faculty and staff members. The salaries of these persons, pro-rated by the fraction of their time which was devoted to the course, amounted to $53,000. This figure, however, takes no account of the fact that faculty time, at M.I.T. as at other universities, is the scarcest of resources. The course also accounted for $12,000 of rent on punched card equipment. This equipment, whose utilization is almost entirely chargeable to the course, occupies approximately 1000 square feet of uncharged floor space. Finally, the course was charged $8,000 for its use of a large-scale computer. Yet, for all this expense, the course is not effective: more significantly, it is batch-oriented; while M.I.T. has made routine use of time-sharing for five years.

The course makes use of standard batch languages, and in the 1967-68 academic year, the language in use was FORTRAN IV. Much of FORTRAN is, in a sense, unteachable. That is, while the language can surely be learned by rote, much of it can not be logically motivated and explained. Moreover, many important programming concepts are missing in it and other common batch languages. Surely one measure of the success of an elementary programming course must be the ease with which students move on to more theoretical courses or to other languages. But such transitions are difficult when the course has never given experience with recursion, block structure, functions as objects, or manipulation of strings. Finally, it does not seem possible to teach any of these languages, none of which was designed with teaching in mind, in any pedagogically sound sequence. One cannot teach the transfer-of-control statement before one has taught the use of labels; one cannot teach anything until one has taught the nature of the compile-load-run process; and so on. Students must take much on faith, and this is particularly hard on those students who wish to understand only that core of material which is necessary for their work.

B. SYSTEM OVERVIEW

The existence of the versatile CTSS computer time-sharing system at M.I.T. provided us with an opportunity to attack both the problem of reducing the Institute's staffing burden, with respect to instruction in elementary programming, and the problem of devising a rational teaching strategy for that subject.

It seemed clear to us that, whatever the state of computer-aided instruction (CAI) with respect to other subjects, the computer is the ideal instrument for teaching its own use. If a computer is to be used to teach any laboratory subject, then it will have to be made to simulate that laboratory during some stages of the instruction. This imposes severe difficulties in most cases; but a computer can be made to simulate a computer quite easily. Also, a large part of any laboratory course consists of a student's actual experience with the laboratory's hardware. In the computer-based teaching system which we have developed, the student is manipulating the laboratory hardware, so to speak, from the very beginning and throughout his instruction.

The major resource we had available to us in the summer of 1967 was the Compatible Time-Sharing System (CTSS). This includes a large number of teletypewriter consoles for communicating with an IBM 7094 computer, as well as a generous set of software facilities. Of these software facilities, the most important to us were the CTSS file system, a good algebraic language (MAD), and a list-processing

language (SLIP). The problem of teletypewriter-computer communication had also been completely solved for us by the time as we undertook our task.

We began by setting ourselves certain goals and objectives. Chief among these was that we intended to teach programming – not some particular programming language. We realized, of course, that some programming language was required as a vehicle to convey the ideas we intended to communicate. Our task of designing such a language was governed by the following criteria:

1) A student, having learned our language, should be in a position to learn any standard algebraic language very quickly on his own, simply by studying the manual for that language.

2) Our language should accordingly contain all the fundamental ideas of current programming practice. These should be clearly identified and appropriately named – we should not introduce new jargon. Examples of such ideas are: identifiers and variables, control of iteration, recursion, the subroutine, and user-defined functions.

3) Every important idea should be presented only after a need for it has been clearly established. To the greatest extent possible, the student should be brought to the threshold of inventing the idea himself just before it is presented to him. For example, he should have had to write a particular program segment many times within a single program before the introduction of the idea of the subroutine. While this criterion is more pedagogic than one of language design, it translates into the latter. In particular, it dictates the dependency relationships among various modules which, when finally joined, constitute the whole language. It determines, for example, that facilities necessary for construction of large program segments must be made available before mechanisms allowing true subroutining. Hence, for example, loop control mechanisms must be independent of subroutine mechanisms.

4) When, in the design phase of the language, a particular facility may be provided in a number of different ways, the design that is capable of being explained most simply and clearly should be chosen. In general, we believe that a hard or awkward-to-explain implementation is probably wrong. The faithful adherence to this design criterion has led us to

deep and searching discussions about questions which appeared initially to be simple and even superficial. Our own understanding of language design has deepened and, we believe, the language we have produced is uncommonly clean.

The teaching system itself – the system that ultimately presents lessons to the student, supervises his progress, and permits him to exercise his skills – was also thought out in terms of goals and criteria. The main ones were the following:

1) An individual student's rate of progress through the lesson material is to be governed by the student's own actions. Each student must be given a separate filing area for his work.

2) The system must remember an individual student's level of progress at all times. The student should not be forced to repeat work, say when re-entering the system after a few days absence, because of system bookkeeping limitations.

3) The system must detect student errors as quickly as possible; whenever possible in a highly localized context. Once an error is detected, it is to be pointed out to the student in as unambiguous a fashion as possible. Finally, the system must permit the student to correct errors on a highly localized basis – without forcing him to reconstruct large portions of error-free work.

4) The system must be totally protected against catastrophic system failure due to student error.

5) The system must be modular, in the sense that changes either in the language to be taught, or in the lessons themselves, can be made easily and independently of one another.

We constructed an experimental system during the Summer of 1967, and about 10 students served as experimental subjects for us during the Fall 1967 semester. On the basis of that experience, and much more thought, we developed the currently running TEACH system, which does, in fact, provide virtually tutorless instruction in programming.

At the heart of the TEACH system is a language which we have chosen to call PL/2. We believe that PL/2 meets the first set of criteria mentioned previously. It is an interactive language that somewhat resembles JOSS, but it differs from JOSS and other JOSS-like languages in several major respects: for example, the presence of block structure,

a context editor, and a function-tracing feature. As a result, PL/2 is somewhat more amenable to rational explanation – say, in the metaphors of mathematics – while it is somewhat less amenable to cookbook explanation (the language may not be as appealing to non-programmers). Siginificantly, PL/2 (like JOSS) may be used as if it were merely an expensive desk calculator. It is sufficiently rich, on the other hand, that quite complex computational tasks are well within its domain.

Surrounding the PL/2 interpreter is another, much simpler interpreter called AGES. Programs written for AGES are called scripts. The AGES language is quite general, but the body of scripts that now exists uses little of that generality. This body of scripts constitutes an elementary course in computer programming.

The most prominent activity of these scripts is simply typing information to the student. The material typed is, in sum, an ordinary text in computer programming. It is divided into eighteen chapters, each with five to ten sections. The chapters and sections are calibrations of an accounting mechanism maintained by the scripts. This mechanism allows a student to leave the system and return days or months later to the point at which he left off.

The scripts are able to call upon the PL/2 interpreter, and occasionally the scripts themselves use the interpreter to supply function definitions and other values to the student. More often, the scripts call upon the interpreter only so the student may use it. For reasons which will be discussed later, the scripts have little control over what a student actually does with the PL/2 interpreter; for, after a script has suggested a problem and given the interpreter to the student, he might just as easily do his physics homework as the problem suggested by TEACH.

Whatever else may be said of this freedom for the student, it presents one serious risk; that is the student will blunder into use of a feature which he cannot control; for example: he will blunder into transfers of control before he understands the procedure for interrupting an infinite loop. To eliminate this risk, the scripts are able to communicate with the syntax scanner of the PL/2 interpreter. In particular, the scanner will not recognize any construction which the scripts have not already discussed. In effect, a student is limited to making simple mistakes – where the word "simple" is moving as fast as he moves, but no faster. The scripts are also able to engage in limited dialogue with students, and can allow the students to request hints about suggested problems, and to even skip certain sections entirely. This feature, along with miscellaneous general-purpose features of AGES, also allows students to review sections either they have skipped or, for some reason, they wish to see again.

C. EXPERIENCE WITH STUDENTS

A preliminary version of the course was taught in the Fall of 1967 to about a dozen graduate students and faculty members in the Political Science Department at M.I.T. Because the scripts were then in crude condition, it was necessary to supplement the students' computer instruction with regular recitation sessions. These sessions provided much guidance for the complete rewrite of the scripts that was undertaken in the Spring of 1968.

Forty new subjects were exposed to the course in the Fall of 1968. Half of these were political scientists similar to the first year's sample. The remainder were a group that might have been taken at random from the large elementary course: mainly freshmen, with upperclassmen from engineering and scientific departments.

At the beginning of the term, each student was given a short handout telling him how to find a computer terminal around M.I.T., how to access TEACH, what to do if out of paper, and so on. The terminals were available 112 hours a week, and were used without sign-up lists or other prescheduling. As each student reached the middle of the course – that is, just after he had reached the material on transfers of control – he was given a handout on flow-charting techniques. Finally, at the very end of the course each student was given a long handout describing compiling, batch-processing, and various other facts of life he might have to face, once out of the course. Except for these handouts, a student's only communication from the instructors came in a personel, computer-stored mail box which TEACH printed for him at the beginning of each on-line session. The instructors used the mail boxes for notes to students and for operational announcements of general interest.

One discovery made early in the term was that different students reacted quite differently to the freedom with which they had been trusted. Some students hoarded their computer time jealously, trying few or none of the suggested problems. Others were fond of extravagant experiments with each possible form of each new mechanism.

Those in the first group were not a new breed: there have always been students who skim reading and turn laboratories into mechanical tests of manual dexterity. There is little to do with such students beyond encouraging them to change their ways. In any event, they damage only themselves. The spendthrifts of the second group, on the other hand, are quite a different problem. We had to decide what to do with a student who, without having finished the entire body of scripts, had already used up his fair share of computer time.

On the one hand, it seemed unlikely that all of the other students would use up their allotments. Whereas one might well discourage such open-handed use of the system, perhaps such a student should not be disconnected from the system until he has far exceeded his quota of system resources. We rejected these notions, and decided to cut off any student overstepping his allotment. We did this because we had come to see TEACH not so much as an automated course, but as an automated book. That is, we discourage inefficient use of TEACH; just as we discourage inefficient use of library books. When a book is due at the library, it is due – whether or not the borrower has made efficient use of it.

To implement this policy, we arranged that at the end of each console session, a student would be informed of the computer time he had used. Each student was then responsible for holding his own time charges within an announced limit. Students were informed of the M.I.T. Information Processing Center's rate structure, and they were encouraged to use ordinary cost-cutting strategies, such as working late at night, in their own behalves. That the time units were given in dollars seems to have lent a felicitous air of reality to these proceedings.

D. PEDAGOGIC PROBLEMS

We have made no tests to determine the efficiency of this teaching method; and since our sample was so small, we are doubtful of the utility of any test which could now be performed. Even if our sample were larger, it remains true that in computer programming what is easiest to test (FORTRAN's IJKLMN rule, for example) is always of least interest to a professional. The effectiveness of the TEACH scripts will be further obscured (in TEACH's favor) by the additional motivation which seems to be inherent in on-line interaction. We do not regard the latter as just a gimmick; but, even if it were the only benefit of on-line instruction, to arbitrarily shun such motivation would be to plead for a Puritanical, cough-medicine theory of education.

We have observed that TEACH does not notice what students do with the PL/2 interpreter: whereas a good teacher, examining a student's program, often discovers problems of conceptualization of which the student was unaware. At the moment, we don't know how to program a good teacher. Certainly TEACH could be more watchful: it could, for example, easily administer Skinnerian training in its spelling rules. But the interpreter's ordinary diagnostics appear to be quite adequate, and we are not convinced that a change in TEACH's style would be an advance.

E. ECONOMIC PROBLEMS

A prime motivation for developing TEACH was the pressure of a course costing about $75 per student. The present TEACH system costs about $150 in machine time and $10 in faculty time per student, but the former cost is an artifact of the present implementation: of the $150, about $90 is swap time; the result of having a large, I/O-bound program in CTSS. In a system which allowed shared user procedures (such as Multics), the $90 would be cut substantially; in a dedicated system, the $90 would be cut to zero. Of the remaining $60, we believe that at least $15 would be saved by reimplementation on a machine more adapted to time-sharing than the 7094.

The only remaining economic issue has to do with terminal devices. Teletypewriters, at 15 characters per second, are quite adequate for short notes, or for material which requires thoughtful reading. For reference materials, however – or for discursive description – this rate is irritatingly slow; and, of course, teletypewriters cannot provide the flow-charts and other graphics to which we should like to expose the student.

Both of these problems might seem to be met with simple graphic terminals, but available graphic terminals are far too expensive for wholesale use with TEACH. Also, none of them provides the hard copy which, we feel, is necessary to the student. There is a definite gap in this area.

F. EVALUATION

We are, at this writing, near the end of a semester in which two sections of the TEACH course have been taught. We are thus in a position to evaluate the effectiveness of the course and to make some recommendations about ways in which research and development on the TEACH system should be continued.

We lack an independent estimate as to the value of the course to students. Within the last week or two, however, we have held intensive interviews with a fairly large sample of students who have taken the course, and who initially knew little or nothing about programming. This has indicated that the students have in fact absorbed the material which we attempted to teach them. We believe we are now entitled to be impressed with their knowledge of programming. Of course, it will undoubtedly require the passage of some time, possibly a year or two, before the effectiveness of TEACH as a first course in programming will be proved or disproved by the performance of its students. But we

feel quite confident that the course had largely achieved the aims we initially set for it and for ourselves.

The technical aim of completing the system and bringing it into production has certainly been very thoroughly met. The TEACH system is smoothly operating. Students almost never report troubles with the system that are due to technical errors. Whatever difficulties are being uncovered are almost entirely of a pedagogical nature. We interpret this as a tribute to the technical execution of the system and hence most certainly to the programming skill of Mr. Yochelson, its chief implementer. A large share of the credit for our technical success must also go to CTSS, which has once more proved to be a reliable and elegant host for a conversational subsystem.

Looking ahead, we see problems in both exporting TEACH and enhancing it. It seems very clear to us that if TEACH were to be implemented on a machine that is widely available – say, a disk-bearing PDP-6 – then it would quickly come to enjoy a rather large public. In view of the critical national shortage of teachers of programming, such a system would indeed be a significant national resource. Locally, the availability of TEACH on a common machine might result in widespread use at MIT. CTSS, of course, is much too highly loaded for it to serve as a host for large-scale use of such a tutorial system.

If we are to make TEACH widely available, we must also decide what will be students' future access to the teaching language. One of our aims has always been to teach PL/2 in such a way that students will then find it very easy to learn other languages; like ALGOL and FORTRAN. Given a TEACH system, however, it is a fairly simple matter to create a somewhat larger integrated system in which the PL/2 language can be used independently. We can certainly conceive of a system which begins with the existing course of instruction – continuing to utilize a PL/2 interpreter – that could go on to permit students to use PL/2 independently of the instructional sequence, though still interpretively. Finally, this system could contain facilities for compiling and executing relatively large programs. But, although we strongly recommend creation of such a system, we feel that such a project is very largely a production job, which M.I.T. faculty should not spend energy pursuing or even, to any great extent, supervising.

The problem of enhancing the current system is very much more interesting, since it is, in every sense of the term, a challenging open-ended research problem: fundamentally, checking on what each student is doing. Remember that presently, when a student is asked to perform certain tasks, the system turns control over to him and leaves control

with him until he types the word "return". The system does not oversee what he actually does beyond performing continuous checking for syntactic errors. If, for example, we ask the student to write a program to compute square roots, then once he types "return" we assume that he has successfully written and tested such a program. Clearly, we should like to enable the machine to inspect his program from a tutorial point of view, so it could help the student if he makes semantic programming errors as well as purely syntactic ones. This is very hard.

Two questions arise in the context of semantic error correction. One of them is the essentially technical issue of error detection. (That alone is a substantial problem in artificial intelligence.) The other is the pedagogical problem of choosing effective strategies for informing a student of what error he has made, and for offering him suitable help at the appropriate time. The system is sufficiently modular that programs for semantic error checking could be inserted without undue difficulty. The problem, of course, is to write these programs. We now feel that we are beginning to assemble the first ideas required for this task.

Our plan, therefore, is to launch into a fairly substantial research program. We wish to explore these ideas: at first in the context of TEACH. Ultimately, we would want a system that is a teacher of programming in more than a trivial sense. Our estimate is that useful intermediate results – although not in the form of a production system – will begin to appear within two years of initiating this effort. A satisfactory system might come into existence a year later; though we do not wish to promise a production system by that time.

PROGRAMMING LINGUISTICS

Programming Languages

A. BCPL Maintenance

B. The PAL Language

C. Compiler Theory and Research

D. Extensible Languages

E. Formal Semantics

Academic Staff

A. Evans, Jr.

J. M. Wozencraft

Instructors, Research Associates, Research Assistants and Others

T. J. Barkalow

L. I. Goodman

R. Greenblatt

J. H. Morris, Jr.

M. Richards

H.-M. Toong

Programming Languages - Arthur Evans, Jr.

A. BCPL MAINTENANCE

The language BCPL has been maintained and upgraded on CTSS. During the spring and summer, Martin Richards made a major improvement in the efficiency of the compiled code.

BCPL continues to gain acceptance in the computer profession, as evidenced by the fact that it has been implemented for use with other computers. A group at Bell Telephone Laboratories has implemented the language for the GE 635 under GECOS and for the GE 645 under Multics. These implementations are for identical languages, and both are complete and operational. They in turn have been used to implement SNOBOL 4, QED (a text editor), and other subsystems.

We have cooperated with a group at Lincoln Laboratory in implementing BCPL on the TX-2. This task is almost complete.

Last spring, Martin Richards and Professor Evans taught a graduate seminar on compiler theory and construction, in which a detailed examination of the BCPL compiler was a major teaching vehicle. Many students chose as a term project the implementations of BCPL on various computers. Although, as expected, none of these projects seems to have produced a working compiler, knowledge of and appreciation for BCPL has increased nonetheless.

The implementation of BCPL on the IBM 360 under OS was completed during the summer, under the direction of Martin Richards. Documentation of this project is now almost complete.

B. THE PAL LANGUAGE

As a result of experience gained using PAL in the M.I.T. undergraduate programming course 6.231 last spring, minor revisions of PAL were made during the summer. The most important change was to make specification of the language syntax more teachable, in that it is now possible to put the entire syntax on one page. It is already clear from student feedback that the improvement was worthwhile.

Research is underway to make a major improvement in the efficiency of PAL's runtime interpreter. A Doctoral student (Herbert Weinblatt) hopes to reduce, by perhaps 50 percent, the time required to run PAL programs. This research is also relevant to the compiler work of the next section.

C. COMPILER THEORY AND RESEARCH

A Doctoral research project by Frank DeRemer is underway to facilitate automatic generation of the syntax analysis part of a compiler from the Backus-Naur specification of a language syntax. Although this has already been done elsewhere, the present effort is noteworthy for several reasons: most previous work has been either very theoretical (and as a result not too practical) or quite ad hoc (and as a result has little theoretical grounding). It appears that the scheme to be produced will be fairly efficient. Further, the work is being tied in closely to the theory of programming languages in two specific ways:

1) The set of grammars for which the technique is applicable will be characterized accurately. It seems now that this set is the LR(k) languages of Knuth.

2) The recognizer generated will be a single-stack pushdown automation, a scheme with solid theoretical underpinnings.

A translator-writing system developed elsewhere (the GENRAP system of Computer Associates) is being implemented locally in BCPL as a Master's research project. This will provide a useful tool for other development underway.

A MAD compiler for Multics has been written in BCPL as a Master's thesis by Henry Ancona. An important tool in this effort was a loader for reductions analysis programs, written as the subject of a senior undergraduate thesis by Leonard Goodman.

D. EXTENSIBLE LANGUAGES

A project on extensible languages is just getting underway. Graduate student Robert Thomas is developing techniques for specifying the syntax and semantics of syntactic extensions to a programming language, using BCPL as a vehicle. When this research reaches the stage where it becomes appropriate to do an implementation, the research of DeRemer (reported above) should provide a valuable building block.

The area of extensible languages is expected to become of greater importance to this group.

E. FORMAL SEMANTICS

Research has continued in the area of formal specification of the semantics of programming languages. To a large extent, this research has used PAL as a vehicle. In addition, the results of this research, although currently incomplete, are being applied in the research of Robert Thomas on extensible languages.

In connection with this area, Professor Evans has been active in a standards committee (X3.4.2c2) concerned with techniques for formalizing the definition of PL/I.

BLANK PAGE

RESEARCH LABORATORY OF ELECTRONICS

Academic Staff

G. D. Bernard	W. L. Henke	D. Prerau
A. Bers	F. F. Lee	L. D. Smullin
W. L. Black	J. A. Mangano	K. N. Stevens
R. J. Briggs	G. H. Matthews	D. E. Troxel
S. C. Brown	R. R. Parker	
H. A. Haus	P. L. Penfield, Jr.	

Non-Academic Research Staff

J. Harwitt	E. R. Jensen	O. J. Tretiak
J. H. Hewitt	R. A. Sayers	

Instructors, Research Associates, Research Assistants and Others

S. R. Brueck	J. E. Green	H. M. Schneider
S. L. Chou	T. K. Gustafson	W. Stallings
R. Crochiere	E. G. Guttmann	R. N. Wallace
J. A. Davis	B. R. Kusse	D. A. Wright
G. W. Goddard	R. R. Parker	I. T. Young

Introduction

The Research Laboratory of Electronics provides facilities for academic research in general physics, plasma dynamics, and communications sciences. The research reported in the following sections was supported principally by the Joint Services Electronics Program of the Army, Navy and Air Force, with additional support from the Atomic Energy Commission, the National Institutes of Health, and the National Science Foundation. The faculty and students listed on the preceding page made substantial use of Project MAC facilities in RLE research programs covering a broad spectrum of subjects.

Stability Analysis of Continuous Systems - Richard J. Briggs and Gary W. Goddard

A number of possible methods for determining the stability of uniform, time-invariant systems have been investigated. The basic method involves a mapping of the dispersion relation for waves, $D(\omega, k) = 0$, into the complex ω and k planes. A number of different examples have been considered to illustrate the computational economy of the various algorithms. The work was completed in April 1968 and submitted as an M.S. thesis to the Department of Electrical Engineering. (See Goddard, Appendix B.)

Automatic Machine Recognition of Human Erythrocyte Types - James E. Green

Methods for automation of routine medical tests has evoked much interest during the last several years. One such test which has received much attention in this laboratory is the analysis of blood smears. Ian T. Young of this laboratory is presently working on the automatic classification of leukocytes (white cells) in blood smears using color information. (See Young, Appendix C.) This report concerns progress toward recognition and classification of normal and abnormal erythrocyte (red cell) types in human peripheral blood smears. Work on this project is partially supported by PHS Grant P01GM-15006-01.

The first objective of this study was to duplicate successfully the results of a well-trained medical technician. Work is still progressing toward this end. Future objectives are to use the thoroughness of the computer to detect borderline and subclinical abnormalities that the technician might normally miss. It is also hoped that a device to perform these analyses can be constructed economically, so that non-research hospitals can afford to purchase the device.

Well-prepared blood smears from patients with normal and representative abnormal erythrocytes were obtained from the hematology laboratory at Boston City Hospital. Color transparencies were made of selected portions of the slides at 1000X through a light microscope. These transparencies were scanned on the laboratory's multicolor scanner SCAD* (see Tretiak, Appendix C), on a 108 x 162 point grid. Each point was coverted to a 9-bit brightness value. Although three colors were scanned, only the green record was subsequently processed, as the red cell information was found to be essentially the same in each record. Each record was then reduced to six bits per point, packed onto the CTSS disc at six points per computer word, and a picture reproduced on the line printer using an overprinting routine to simulate grey levels.

The first problem encountered was how to separate cells in the picture from background and trash. If a histogram of point brightness was computed from a picture containing only red cells, it was found that the histogram contained a large, rather sharp peak in the bright region, and a broad peak in the darker region. The sharper peak was found to correlate well with background points, and the broad peak with cell points. Some cell points, however, especially those in the pale, central region of the cell, were as bright as some background points, so a simple clip level would not separate cells from background. Upon examining the brightness values across an individual cell, it was found that the brightness changed rapidly at the edge of the cell, the densest points being located near the cell's margin. Thus we decided to use a combination of a brightness histogram to select a clip level for the cell edge points, combined with a contour tracing algorithm to encircle the individual cells. After selecting a clip level for the cell edges from the picture histogram, the program scans across the picture, point-by-point, until a point darker than the clip level is found. Starting with this point, the edge of the dark region is traced until the original point is again found. If a sufficient number of points are contained within the contour, the object is considered to be a cell; otherwise the object is discarded.

After the cells have been separated from the picture background, there remains the task of feature extraction to make possible classification of the cells. At present, a complete set of these features has not been selected, but several of the more important ones are now incorporated into the computer programs. Among these are size, shape, weight, and radial distribution of the hemoglobin. Size is determined by simply

---

*Tretiak, O. J., "Picture Processing", M.I.T. R.L.E. Quarterly Progress Report No. 83, October 1966, pp. 129-142

counting the number of sample points within the cell contour, and multiplying by a suitable constant. Weight is calculated as the sum of some function of the difference between the average brightness of the background and the value at each point within the cell boundary. If the function is chosen correctly the "weight" should be a measure of the total amount of hemoglobin in each cell. A measure of the shape is derived by calculating proper moments of inertia of each cell, and using these to calculate the eccentricity. The radial distribution of hemoglobin is estimated by calculating the ratio of the radius of gyration of the cell points to the average radius of the cell. These features were found to be reasonably constant for cells in normal blood.

Many other features could be tried, and will be as the work progresses. At this point – due to the time-consuming nature of the process required to transfer a picture from the scanner-to-tape, to converted CTSS-compatible tape, to CTSS disc, and also due to inherent difficulties in the scanner — very little cell data has actually been analyzed by the computer. These difficulties will be removed shortly when the scanner is interfaced with an in-lab PDP-9. This system will make it possible to process data at a much faster rate, and allow meaningful statistics to be compiled on parameter spread among "normal" cells compared with abnormal cells.

Mechanization of the Interpretation of Vaginal Smears - Endre G. Guttman

The purpose of this project is to study the possibility of mechanizing the interpretation of vaginal smears for early detection of cancer of the cervix uteri.

Although the method of cytodiagnosis developed by George Papanicolaou has been challenged by acridine orange fluorescent microscopy, it is still the most widely used method. Skilled technicians interpret the slides under an optical microscope. Located on the slides are exfoliated cells from the vaginal tract. There are several morphological characteristics of the cells upon which a criterion can be established to discriminate cancer cells from normal cells: size and chromatin density of the nuclei, nuclear cytoplasmic ratio, etc.

Earlier efforts to mechanize Papanicolaou's method did not completely succeed; for example, Toller's cytoanalyzer. The present project uses a flying-spot scanner and a general-purpose computer system.

The input data to our system are vaginal smears collected and fixed by physicians. These smears are automatically stained by the Pananicolaou method in the Cytology Laboratory of the Massachusetts General Hospital.

Microphotographs are taken by us from the smears which are scanned by the flying-spot scanner designed by Dr. Oleh J. Tretiak of R.L.E. The resulting picture has 108 horizontal and 173 vertical points with 64 gray levels. The scanner's output tape is read into the Project MAC time-sharing system.

The problem can be subdivided into three sub-problems:

1) Delineation of the objects from the background (machine perception),

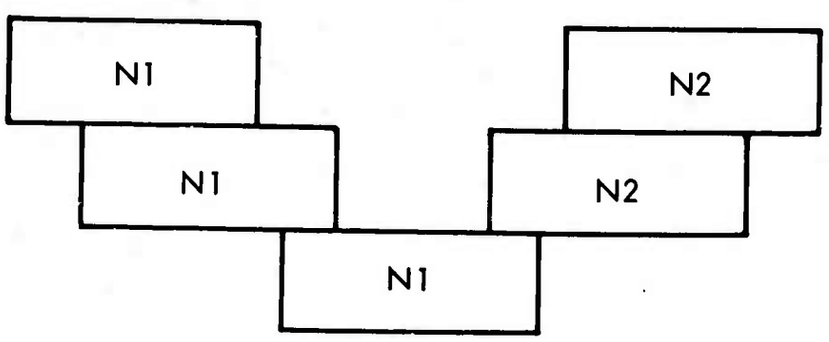
2) Separation of cells from artifacts (definition of a cell),

3) Separation of cancer cells and normal cells (malignancy criterion algorithm).

We decided to solve the last subproblem first, and get involved as little as possible with the first two. A number of subprograms have been written and tested already (non-overlapping, area-extracting contour tracing, chromatin density, and hystograms).

The area-extracting algorithm operates on one line at a time. We suppose that the previous line has been processed and the intervals of black points, as well as the names of the areas to which these intervals belong, have been found and are available to us. We now locate the connected intervals of black points in the current line.

Whenever an interval in the current line is connected to at least one interval from the preceding line, it is assigned to that name. If there is no connecting interval in the preceding line, a new name is assigned to this interval. The only difficulty that is encountered is shown below.



N2 = N1

If an interval is connected to regions with two different numbers, it is given a name equal to that of the smaller of the two numbers, and an entry in a separate list is made signifying that the two names (numbers) correspond to the same region. When this process is completed, the data from the previous line and the name equivalence data may be written on some bulk device.

After all the lines have been processed as described, two lists will have been generated; one containing the intervals and their tentative names, and one with the name equivalence data. The latter contains a string of name-pairs. The pairs are first sorted so that the smaller name of each pair appears first in each doublet, and the set of pairs is sorted by their first names. We then form a dictionary that specifies the actual name to be assigned to each tentative name. A pass is made through the list of intervals, and the final names are substituted. The next step is to connect areas of nuclei to their respective cytoplasmas, and finally to deal with overlapped cells.

Thermodynamics and Self-Steepening of Light Pulses - Hermann A. Haus and Ture K. Gustafson

This research deals with various aspects of quasi-monochromatic optical pulse propagation in media which exhibit the Kerr effect. In particular, optical shock waves develop in regions of very intense laser light, which gives rise to both a "self-steepening" of the pulse envelope and a "self-modulation" of the pulse. The former is similar, in many respects, to acoustical shock wave development. The latter results from the non-linear generation of a phase perturbation.

Self-modulation and possible self-steepening have been observed to occur within "trapped-filaments", that is, regions in which the diffraction of light beams is balanced by a focussing effect arising out of the non-linear dielectric coefficient. The theory of the Kerr effect is being applied to an ensemble of non-interacting linear CS_2 molecules to estimate the radii and field intensities associated with such steady-state filaments.

Models of Language Perception - G. Hubert Matthews

During the past year, the initial phase of this research was completed, i.e., the development of tools needed to carry out the actual proposed research. This consists of a computer program intended to aid a linguist in constructing a transformational grammar. Such a

program might be compared to a literal-minded, indefatigable assistant who knows the formalism of transformational grammar. This assistant can check the operation of a pre-defined set of transformations against a predefined tree, help with the bookkeeping of changes to the grammer and to a set of test trees, but he does not come up with original ideas or use his intuition. Thus, the linguist must be explicit in writing his grammar, a feature which has obvious theoretical advantages. The program is designed so that it can be used interactively with the linguist.

A description of the program has been written, though it is not sufficiently detailed to enable a linguist who has no experience with computers to use the program. It is hoped, however, that it is sufficient to enable him to judge whether or not it would be useful to him. If a sufficient number of people wish to use the program we will write a more detailed manual which would allow someone without prior computer experience to use the program with little outside help.

A. PROGRAM FUNCTION

There are three services that the program will perform for the user through available sets of functions:

1) The user can specify, change, and print (in a "natural" form) a tree. Typically, this set would be used to construct the base trees to which the transformations are to be applied.

2) The user can define transformations and specify the conventions for applying them (e.g., order optionality).

3) The transformations defined by the function in 2 can be applied to one of the trees specified by those in 1. The user (if he wishes) can see the result of applying each transformation as it applies. If a transformation that can apply is optional, the user is asked whether he wants it to apply.

It is expected that the rule tester would be of use to a linguist who has written (or would write if he had such a device to test it) a transformational grammar (or fragment thereof) in which his rules were explicit. He could use the rule tester for the following:

1) To find out whether his rules actually do what he thinks they do (i.e., whether they actually do map the base trees of his base component into what he thinks are the corresponding surface trees), and to modify the grammar when it turns out that they do not;

2) To observe the workings of his rules in detail.

The rule tester might also be of use in teaching an introductory course in transformational grammar. A student using this system could appreciate the notions "transformation", "deep structure", "structural description", etc., by observing the workings of the machinery of transformational application without having to do the tedious pencil work otherwise required. Also, the student would benefit from having the errors that he inevitably will make pointed out to him immediately.

B. RULE TESTER DEVELOPMENT

Future development of the rule tester depends crucially on system users. If no one uses it, there is no point in doing any further work on it. If a user appears who would like the system to do something which it does not do now, we will attempt to modify the program to that end. Such modifications might be: the ability to specify, for each node in a tree, a list of inherent feature names and values; and to test, in determining whether a transformation applies to a tree, whether a node matching a segment of a structural description has certain features. Contextual features have not been added, since they do not (with the usual exception of Walbiri) seem to have any bearing on the application of transformations. If anyone wishes to study lexical insertion, however, it would be possible to add contextual features. Further improvements can be made fairly easily. For example, the notion of "command" has not been programmed, but it could be if someone were interested in using the rule tester to study this phenomenon.

C. PROPOSED RESEARCH

We have begun, and plan to continue, the research originally intended. Given a transformational grammar of a natural language as a statement of a speaker's knowledge of the language, it is natural to ask how it could be used by the speaker to understand the sentences which he hears. To date, several proposals have been offered, and a few of these have been tested with a computer. However, none of these takes into account the fact that a speaker normally does this in real time. The candidate has made two new proposals, both of which attempt to deal with problems arising out of the real-time requirement. The difference between these two proposals is: one postulates that a speaker attempts to find a surface structure in real-time for sentences he hears and subsequently maps onto a deep structure for long-term memory; whereas the other postulates that a speaker does not actually deal with the surface structure at all, but rather finds — in real time — the deep structure of the sentence.

Subtransit-Time Oscillations in the Avalanche Region of a pn Junction - Paul L. Penfield, Jr. and Ronald Crochiere

The object of this project is to formulate a circuit model for a pn junction in the avalanche region and to compare it with experimental results.

The model developed is non-linear and time-dependent. For this reason, analysis of the circuit has been carried out on the Project MAC computer. Branch equations were written for each circuit element and related to other branch equations by Kirchoff's laws. Time is incremented in steps much smaller than the smallest circuit time constant. Currents and voltages are then recalculated for each time interval. The analysis of a circuit may require 10,000 or more time intervals. This type of analysis has proved to be very successful.

Computer Display of Smooth Solids - William Stallings

This research is concerned with the display of three-dimensional scenes by projection of solid objects onto a viewing plane. The most difficult problem encountered has been the elimination of hidden surfaces. A method was developed that applies to arbitrary scenes. That is, any object or group of objects, including non-convex objects and objects without edges (smooth solids), can be displayed.

The method used is that of fitting a geodesic structure to the surface of a solid. Specifically, a solid is approximated by a mesh of small triangles covering the surface. A linked data structure which corresponds to this mesh is used. With detailed information available about the surfaces, the shape and visibility of objects can be determined from any viewpoint. Algorithms using list-processing techniques were developed to do this and to produce drawings of viewing-plane projections.

Simulation of The Cochlea - Alton P. Tripp, Jr.

The object of this research is to obtain a suitable model for computer simulation of the mechanical behavior of the cochlea of the ear. As there are many variables involved, it is desirable to have interplay between the computer and researcher, so that effects of changes of the many variables may be observed. In particular, visual observation of the results of simulation is desired, and for this reason the MAC computer with its KLUDGE display was picked. Initial work on the computational aspects of the problem has been performed, but a convergent iterative program has not yet been devised. Work will continue on paper and in the computer to solve the modelling and computational problems.

Document Room - John H. Hewitt

Linked to Project TIP, the Document Room has answered reference questions arising from requests for literature searches received from several of the groups in the laboratory. We have used not only Dr. M. M. Kessler's Hayden Library file of thirty-four physics journals but also the technical reports on plasma physics in Professors A. Bers' and S. C. Brown's collections. The initial thought of putting Professor W. P. Allis' collection of plasma reports on Project MAC — as are Professors Bers' and Brown's — was abandoned because of lack of funds, plus the delay in reprogramming for a new version of TIP.

For a large part of the past year, the R. L. E. Document Room also administered the CTSS time allocations for Group 13.

BLANK PAGE

SCHOOL OF ENGINEERING

The MAP System

Computer-Aided Design of Space Forms

Design of Three-Dimensional Cubic Curves

Two-Dimensional Stress Analysis

Analysis and Simulation of Multiport Systems

Academic Staff

C.A. Berg
J.W. Brackett
S.A. Coons
R.A. Humphrey
R. Kaplow
F.A. McClintock
H.S. Mickley
J.H. Milligram
R.P. Parmelee
H.M. Paynter
R.C. Rosenberg
T.B. Sheridan
R.L. Taggert

Non-Academic Research Staff

B.F. Boudreau
C. Duren
H.F. Gronemann
R.L. Kusik
T. Lin
J.B. Lovins
R.S. Marcus
R.L. Stetson

Instructors, Research Associates, Research Assistants and Others

E.R. Banks
W.L. Bass
P. Choong
C. Chryssostomidis
D.L. Flamm
J.C. Free
R.P. Greer
T.L. Hawk
R. Hodges
W.R. Kampe
N. Leal-Cantu
P.D. McMorran
P.F. Meyfarth
H.A. Radizkowski, II
C.C. Tillman, Jr.
C. Weissgerber
P. Wieselmann

The MAP System - Roy Kaplow and John W. Brackett

The MAP System for On-Line Mathematical Assistance, described in MAC-TR-24, has been available on CTSS since late 1965 and has been used by persons in many departments. This operational experience has been the basis for designing a new system (tentatively entitled MAP-II) for mathematical assistance, which will be implemented on the GE 645 within the Multics operating system. Although the operations initially available within the new system will use numerical techniques, the system is designed to include non-numeric facilities.

Improvements were made to the MAP System during 1967, but it was decided that the internal structure of the system should be rewritten, because of the modifications necessary to extend the system to deal with data which is a function of two or three variables. In mid-1967 the Multics system was scheduled to be available during the 1967-1968 academic year and plans were made accordingly. The slippage in Multics availability has resulted in this project accomplishing less than had been planned. The initial design of the new system was done in cooperation with Kenneth Busch and Wade Bartlett of Bell Telephone Laboratories, Whippany, New Jersey, and plans were made for further cooperation during implementation of the system. However, increasing delays in Multics availability caused Bell Laboratories to decide that further cooperation would be premature; it is hoped that when Multics becomes operational, further cooperation may be possible.

Far more general and efficient than the present MAP design, the new MAP-II system retains only those outward aspects of user interaction which have proven to be desirable. The internal structure of the system will be centered around 1) a data base which will accommodate general classifications of data as well as numeric data of one-, two- or three-dimensions (real and complex), and 2) an operator table which will accommodate multi-argument operators, and which will contain information (in bit-format) about argument compatibility, to allow checking and/or user interrogation during the interpret mode. The user language itself has been greatly generalized. All operators which produce single-data entity results are includable in the equation format as well as in specifying arguments for "stand-alone" multi-result or non-computational operators. Logical branching has been included in the language, and such branches may be included, in fact, directly in equations to determine the output as well as the input variables. Basic to the system's design are facilities by which a user can add his own operations to those provided by the system in order to increase the capabilities of the system in his own area of interest.

During the past year, the facilities of the AED system have been used to create an experimental language parser (and tree-constructor) on CTSS to experiment with desirable language facilities. The following are a few samples of acceptable statements; the operators themselves have not been implemented on CTSS.

```
broad (x,y) = exp (-q*x**2-p*y**2)

result = inftrans (ftransform(data)/ftransform(broad))

check = (convolution(result, broad) - data)/data

a(x, if c then 2 else 3) = ln (sin|x|)

a(x,2) = ln (sin|x|)/2
```

The parser employs the AED RWORD package and the AEDJR system, thereby allowing the types of items to be found in the input string, and the rules for parsing the language, to be largely embedded in tables. The use of these facilities has promoted experiments with major changes in the format of the input language. Correct AED first-pass structures are generated by the experimental parser, but no interpretation of the structures is carried out, since full implementation of MAP-II on CTSS is not planned. However, it would have been attempted if we had realized in late 1966 the delays which would occur in Multics availability. The work which has been done using the AED system should be directly transferable to Multics when the AED-1 compiler is operational in 1969. Initial work on the graph-plotting facilities required in MAP-II has also been done; the general-purpose, graph-plotting routines (described in memo MAC-M-224), have been converted from 7094 assembly language to Fortran IV, thereby (hopefully) making them useable on both the GE 645 and the IBM/360.

Computer-Aided Design of Space Forms - Steven A. Coons

During this reporting period, research was continued on the mathematics of Computer-Aided Design of Space Forms. This work essentially examines the notion of rational polynomial functions in two independent variables.

It appears that quadric surfaces, surfaces of revolution, and other "classic" surfaces are also special cases of the general surface equation set forth in MAC-TR-41. (See Appendix D.) This signifies that surfaces such as spheres, ellipsoids, cylinders, and cones are compatible directly

with more general "free-form" surfaces, so that the single general surface equation forms the basis for a single algorithm to generate them all.

This is of considerable importance in the mathematical description of form, since spheres, cylinders, and surfaces of revolution occur frequently in design, and should not be different in general structure from more general shapes if these classic and non-classic forms are to be joined together to describe an object.

Design of Three-Dimensional Cubic Curves - A. Robin Forrest

The major portion of this work was performed in parallel with that of Professor Steven A. Coons (see MAC-TR-41) and involved an extension of work by Dr. L. G. Roberts at Lincoln Laboratory on conic sections and homogeneous coordinates to three-dimensional cubic curves. The cubic curves devised permit curves to be produced which twist in three-dimensional space and reduce in certain cases to the general conic section, circular arcs, and straight lines. It is thus possible to derive, from one basic formula, all the types of curves used by the aerospace industry. This is a significant advance. Professor Coons is implementing this at Harvard; I shall do likewise at Cambridge, England; and Douglas Ross will probably use them in the AED Polyface Package as boundary curves. Several sessions have been held with Douglas Ross concerning the graphics to be associated with his Polyface Package, and I hope to be able to advise him on the method of surface description he should use.

Project MAC facilities were used to gain experience with a reliable time-sharing system and to write brief AED programs. The reliability and the large number of languages available are two impressive features of the MAC system.

Two-Dimensional Stress Analysis - Paul A. Wieselmann

The production of plane stress analysis by the complex variable method (or method of Muskhelishvili) rests entirely on having a function to conformally map a particular region into a canonical domain where the problem is in fact solved. After much study and numerical experimentation, a sufficiently general, accurate, and computationally fast technique and programs were obtained to provide the necessary mapping function for the stress analysis program. These programs comprise a coherent system for the solution of the first and second boundary-value

problems of plane elastostatics in simply connected finite and infinite regions. The system also has the special advantage of being able to treat boundary cracks in a precise manner as part of its normal operation.

A synopsis of the system is as follows. From a point description of the boundary, the Schwarz-Christoffel integral for the exact mapping of a polygonal boundary is obtained. A suitably accurate Fourier expansion of this integral then is made. Special terms may be introduced into the series to give it precise behavior at corners and tips of cracks; e.g., in studying stress concentrations and stress intensity factors. The boundary conditions are specified by the user as tractions along segments, or as displacements along segments. With the aid of the mapping function, the problem is solved on the unit circle as a cannonical domain. The system then opens itself to interrogation about the stresses and displacements. At all times the system is interruptable for interrogation and assistance by the user or, more importantly, for direction; e.g., when boundary condition or boundary perturbation studies are made.

The system is now fully operational. Important data in the field of fracture mechanics has already been obtained for special crack problems. A Doctoral thesis in mechanical engineering, a MAC technical report, and a user's guide are all being written. The system will soon be available to the general CTSS community, either as a complete stress analysis system or as a resource for use in studies where conformal maps are necessary.

Analysis and Simulation of Multiport Systems - Ronald C. Rosenberg, E. Roger Banks and Peter D. McMorran.

The multiport representation of physical and engineering components, in which each distinct power is shown explicitly, offers a number of advantages over conventional (signal-oriented) representations. The bond graph notation, which depicts each power as a line or bond and systems as sets of interconnected multiports, serves as a basis for developing a theory of multiport analysis and concomitant simulation techniques. One important feature of this approach is its ability to treat systems with mixed energy domains (e.g., systems involving electrical, mechanical, and fluid power simultaneously) in a uniform fashion. A second advantage is that the structure of a system model is made explicit by virtue of the multiport elements, even when the system has many non-linear components.

ENPORT-2 is a digital computer program capable of reading bond graphs in coded form and generating symbolic representations of their

governing equations. A large class of non-linear systems has been included within this program's range, and further development offers much promise in the areas of non-linear simulation and automated identification techniques for complex engineering systems.

Current work centers on exploiting a canonical form for multiport system representation (the gyrostructure) in both its theoretical and computational aspects. Peter McMorran has developed a simulation technique based on chordal linearization of non-linearities and a stepwise linear integration procedure which is very effective for certain types of problems. Work continues in an effort to extend this success to broader classes of systems.

Enlarging the set of multiport elements acceptable to ENPORT-2 has been the responsibility of Roger Banks. He has concentrated on the definition, storage, and retrieval aspects of enlarging the working bond graph set, so that it now includes such "devices" as PUMP, SHAFT, MOTOR, GEARBOX, and many others. The program has been developed in such a way that the user of an engineering multiport need not understand the computable model which is stored for it; he merely calls for it in the bond graph.

BLANK PAGE

SCHOOL OF HUMANITIES AND SOCIAL SCIENCES

The COMCOM Project

Academic Staff

E. Kuh

I. d Pool

Non-Academic Research Staff

M. Eisner

D.M. Griffel

S.D. McIntosh

R. Sommer

Instructors, Research Associates, Research Assistants, and Others

O.J. Dodson

A.J. Gottlieb

K. Grossen

J.F. Kramer

A. Moulton

H.L. Selsnick

The COMCOM Project - Herbert L. Selesnick

This project, directed by Professor Ithiel de Sola Pool, attempts to simulate a mass-media communication system, so that a social scientist, given demographic-population data and media-exposure data for the population, can schedule a variety of messages via several media vehicles and reproduce the cumulative exposure among different demographic-population types. A background version of the simulation has been reprogrammed to run on foreground at Project MAC.

In the past year, I have been using the simulation to study mass communication in the Soviet Union. Using Soviet-published data on circulation figures for most of the important print media, data on the distribution of electronic media, interviews with Soviet refugees and visitors to West Europe, and time budget studies, I have estimated a large number of Soviet audience parameters. These parameter estimates have been input to the COMCOM simulation to construct a static representation of the Soviet mass-media system. We are now in the process of simulating the flow of messages within this system — via the mass media to the Soviet population — during the Cuban missile crisis and the aftermath of President Kennedy's assassination. We have reproduced a hypothetical sequence of messages appearing in the Soviet mass media during these two periods, based on a content analysis of secondary sources such as samplings of Soviet press and radio coverage, contemporaneous issues of the major Soviet print media, and transcripts of foreign radio broadcasts to the Soviet Union. We have also used published studies of the Soviet press, radio and TV, and journalistic accounts of media treatment of these two international crises to estimate how various subgroups of the Soviet population might have responded to media material at these times. The resulting message "scenario" has been input to the static mass-media system to simulate Soviet exposure to news of these two events. The resulting exposure histories are being investigated to answer the following questions:

1) What proportion of the Soviet population heard about each event?

2) How did that proportion differ from city to country, from party member to non-party member, from highly to poorly educated, from young to old, from men to women?

3) Which people heard the Western version of events?

4) How long did it take the majority to hear it?

5) How many heard it confirmed a second and a third time?

In the future, we expect to use the COMCOM simulation to study Soviet and Chinese mass media exposure to messages about the Sino-Soviet rift and to messages about the current Czechoslovakian-Soviet rift.

In addition to the support received from Project MAC, the COMCOM project is supported by the Advanced Research Projects Agency under Contract No. 920F-9717 with the Center for International Studies, M.I.T., and monitored by the Air Force Office of Scientific Research under Grant No. AF(49)638-1237.

SCHOOL OF SCIENCE

DISHPAN

Academic Staff

K. Biemann

J.G. Charney

Instructors, Research Associates, Research Assistants and Others

R.C. Gammill

M.A. Geller

DISHPAN - Robert C. Gammill

The purpose of this project was to explore the usefulness of the ARDS-II graphical display console for plotting contour maps of meteorological parameters with geographic overlays. This problem is of interest, because on-line display consoles have never been fast enough in the past to present all the information in a meteorological contour map without flickering intolerably. The ARDS-II, by allowing a great deal of information to be presented once and stored on the tube face circumvents this problem. The remaining question was whether programs could be written to produce the desired maps economically and fast enough to be useful to an interactive user. It turned out to be not only feasible, but very effective.

Several programs were written during this project. The first to be completed was a general-purpose plotting program that allows a user to display a simple word map as geographic land mass outlines in either polar sterographic or orthographic projection. This program is quite popular as a demonstration of the capabilities of the ARDS-II display. A hemispheric map produced by this program requires some 20 to 24 seconds of machine time to generate and plot. If the generated display commands are saved and plotted, the plotting alone requires about 11 seconds. Considerable savings can be made over these figures by decreasing the amount of detail plotted via a user-specified parameter. It should be noted that the program was written in MAD with no attempt at optimization.

The rest of the programs written during the reporting period were contour programs. The first contour program produced all the lines passing through a particular grid square, and then moved to the next square. (A grid square is the area marked out by four grid points.) This method had the disadvantage that two plotting commands were necessary for each line segment plotted; the first to move the scope's beam to the beginning point, the second to plot the segment. Despite the extreme simplicity of the program, the time necessary to transmit so many commands to the display, and the space necessary to store them, made this program appealling. Several other experimental contouring programs were written, culminating in a program which contoured by "following". This method is much more complex, but minimizes the number of display commands which must be generated, since it follows each contour until it closes on itself or touches a boundary. This final contour program allowed area outlines to be plotted in either polar stereographic or orthographic projection. Simple maps containing 30 to 40 individual contours require approximately 13

seconds of computer time to generate and plot. Complex maps with 80 to 90 contours require approximately 30 seconds. (All times include CTSS swap time.)

Thus, a meteorologist can easily plot a simple contour map with a pregenerated geographic overlay in approximately 24 seconds of 7094 time. This figure is achieved without optimization of the MAD programs, and it seems likely that an increase in speed by a factor of two could be achieved without noticeable degradation in appearance. Feasibility of contour mapping in CTSS on the ARDS-II seems amply demonstrated.

The following photographs were taken in the course of experimentation and the same field is contoured in every case, since only one test data array was available. Figure 16-1a is a polar stereographic projection of the globe with a contour overlay. (A view is considered to have a scale factor of 1.0 when the full globe projection just fills the screen. The views may be reduced or enlarged by integers or decimal numbers to fill the screen with a selected area. Any geographical point may be centered on the screen by specifying its exact latitude and longitude.) Figure 16-1b is the same view presented twice as large with many more contours. Figures 16-2a and 16-2b show orthographic projections of the globe as seen from the North Pole, with different numbers of contours. The times to execute each program can be distinguished on the photographs. The command "r contur" causes the execution of the contour program, while the command "r dispic" causes the plotting of pregenerated plot commands to produce a map of the world.

a) Scale 1 with partial overlay

b) Scale 2 with detailed overlay

Figure 12. North Polar Stereographic Projections of the Earth

a) Scale 1 with partial overlay

b) Scale 1 with detailed overlay

Figure 13. North Polar Orthographic Projections of the Earth

TECHNICAL INFORMATION PROGRAM

Introduction

Physics Input Processing

Data Base Production and Management

Book Production

File Editing

Research on Document Relationships

Special Bibliography Production

Code Conversion

Data Adaptation

Administrative Information Handling

Computer Production of Library Cards

Serials and Journals

Acquisitions and Accounting

Soft Copy Displays

Subsystem Tuning Aids

Academic Staff

M. M. Kessler

Non-Academic Research Staff

T. F. Dempsey
D. M. Jordan
W. D. Mathews
L. H. Morton
W. Nissen
K. D. Rude
P. M. Sheehan

Students

E. M. Mattison
R. J. Wald

Introduction - Meyer M. Kessler

The Technical Information Program (TIP) is mainly supported by the National Science Foundation. Special applications have been supported by Project MAC, the M.I.T. Libraries, and the Graduate School Office. Although most of the work reported here was not directly supported by Project MAC, we continue to be heavily indebted to MAC for earlier support of our work and for the continuing flow of ideas, suggestions and encouragement we have received.

Project TIP has emerged from a phase of development and prototype operation of its computer systems into a period of application and production. Much thorough testing provides the basis for further design advance. TIP has derived much from its close association with Project MAC, and the constant influence of the computer has insured that the systems constructed are economically sound and sufficiently general. The Library has benefitted from a healthy cross-identification of its purposes with the broader data-collection efforts of the whole scientific community. During the last year, TIP has intensified its study of the management of large data bases, their efficient interrogation, arrangement, and internal organization. Since our last report we can mention the following:

1) Progress continues in the transfer of certain TIP functions to the American Institute of Physics for national application. Among the on-going cooperative functions undertaken with the AIP are: a) transfer of data-generation functions (It is expected that sometime in 1968-69 the AIP will produce all the machine-useable data for physics.), b) two AIP staff members working with Project TIP personnel to adapt our procedures to their needs, and c) continuation of TIP as general consultants in the evolution of a National Physics Information Center.

2) At M.I.T. the on-line use of TIP systems is constantly growing. In addition, a serious selective dissemination of information (SDI) effort was started with a group of chemists and chemical engineers. This experiment is being closely monitored and may be applied to other groups of scientists.

3) A text management system, using TIP programs applied to locally generated data, was organized for a project in the Political Science Department.

4) The application of TIP programs to administrative files and data took a giant step forward with the conversion of several Graduate School functions to TIP operations.

5) In the Library, the computer-produced Current Serials and Journals (CSAJ) project has been expanded to include a wider operational scope. During the 1967-68 fiscal year, the library staff added over 1,600 titles to the computer files and made over 1,300 changes to existing entries. Work now in progress will result in a completely automatic system from acquisitions to cataloging. Monographs will be covered, as well as serials and journals. A study was made of the integration of our CSAJ system with that of Wellesley College. Acceptance of the system by Wellesley is apparently contingent on the availability of funds.

6) Studies are now in progress to evaluate the costs and engineering problems of producing catalog cards by computer. Initial results indicate that the plan is feasible and will result in the acquisition of machine-useable data as a by-product.

7) A catalog for machine storage and retrieval is being generated as a sub-element of the Science Library. This will serve as a prototype for the evaluation of on-line machine-useable catalogs.

8) A TIP User Station, consisting of a computer console, microfilm retrieval units, and associated equipment has been installed in the Science Library. This station is operational, and in addition to its usefulness for library patrons it is now possible for the librarians themselves to construct moderate-size computer files as special reference tools.

We anticipate a further broadening of TIP applications in the year ahead. Refinement of our designs, and adaptation to more generally available computers, is under way.

Physics Input Processing - Timothy F. Dempsey

The production of a physics literature data-base continued during the past year. A subsystem was developed to prepare the data for presentation to proofreaders. This subsystem performs preliminary editing of the data; extraneous punctuation and parentheses are discarded, and unrecognizable data is separated to make it stand out.

In addition to the formatted copy of the typed data, two auxiliary outputs are produced. The first of these contains statistics for each article and for the whole issue. Statistics about number of authors and citations in an article, the number of characters typed and the number of lines in the formatted article, provide a succinct and extremely useful overview of the size and content of each article. This file also contains the rudiments of a computer-based control system for the typing shop. The size of the statistics file is about ten percent the size of the formatted output file and its production adds very little processing time.

The other auxiliary output file is more expensive to produce, both in size and time; but its value is equal to, or may even surpass, that of the statistics file. This other file is called the "citation index file". It is produced in a form suitable as input to the TIP system and to the SORT subsystem. Each item in the file contains a field for every "subfield" in the original input. In addition, fields indicating the identity of the article in which the citation is found, and the relative position of the citation within the article, are added to the item. After the file is ordered using the SORT subsystem and after journal citations are selected using the TIP command, the resulting file is presented to a proofreader along with the formatted full text. Using the citation index, the proofreader can readily detect invalid or inconsistent coding in the CODEN, volume, page, or data-base.

Besides being a proofreading tool, the citation index is a useful data-base in itself. Accordingly, these data are also stored on magnetic tape and are periodically combined with other citation indices and again sorted to provide a more meaningful data base.

Although the program described above is the heart of the TIP physics data-processing application, it is by no means the only part or the most expensive. Most of the computer cost incurred by Project TIP for this application is spent in presenting data to the above process and in disposing of its output. Investigations are underway, and will be pursued during the coming twelve months, to determine how to bring these overhead costs down to a more reasonable level.

Data Base Production and Management - Kenneth D. Rude

With the development and implementation of a more sophisticated TIP command, it became desirable to create more complex and much larger data bases. The physics data base, which TIP has been building for some years, has been expanded to include more journals and far more information from each journal article. Other data bases are also being created.

Having accepted the burden of providing large, diverse data bases, we were confronted with the problem of managing their creation, storage, and use. An awareness of the difficulties arising from large data bases does not come full blown when the decision is made to have them, but rather gradually during the implementation and then continuously during the entire life of the data base. The goal is to develop an efficient, smoothly functioning data production operation which uses TIP subsystems to process and maintain the TIP data bases.

So far we have learned the following: stuff gets lost; bottlenecks form; personnel (and personal) relationships change as the work force grows larger; and inefficiencies become absolutely more costly. To overcome these problems we have discovered that:

1) Scheduling of data production work is imperative;

2) Diligent record keeping is necessary to control work flow;

3) Formal, written rules and policies are required;

4) Careful cost accounting must be performed; and

5) Statistics must be gathered, to develop a basis for comparison between possible alternate processing schemes.

No doubt our headaches have just begun.

Book Production - William D. Mathews

Project TIP assisted in the production of Survey Research on Comparative Social Change: A Bibliography, published by the M.I.T. Press. The book contains abstracts of research articles and reports. TIP supplied the authors of the book with a set of subsystems making it possible to edit, arrange, index, and format the book entirely by computer. An important by-product is the machine-readable master file which can be automatically updated. Future editions of the book and its index can easily be produced.

File Editing - Lewis H. Morton

QEDIT is a high-speed editor for ASCII files. It is non-interactive, taking its instructions from a file. Only one type of request is permitted, that of substituting one string for another. The command format is:

RUN QEDIT INPUT FILE OUTPUT FILE LOCAL GLOBAL

INPUT FILE will be transformed into OUTPUT FILE using instructions from files LOCAL QEDIT and GLOBAL QEDIT. Requests from the "Global" file are exhaustively executed on each input line. The file is then scanned and each request from the local file is applied once, in the order found. A system of controls exist, so that a local request can be ignored if it fails within a given number of lines. The request interpreted uses EDIT syntax (see our publication TIP-TM-105), allowing matching with OR'ed criteria and indefinite length and content strings.

A TIP-reducable file named OUTPUT REPORT is also created. This contains one item for each local change made, with its line number, and input and output text. An additional item gives a summary of local and global changes made in the entire file. This is the fastest ASCII editor for large files currently available on CTSS.

Research on Document Relationships - Kenneth D. Rude

A serious problem facing the library profession is that of adequately exposing the contents of their collections. Researchers working in the field of documentation have long been intrigued with the idea of using a computer to automatically generate indices, or bibliographies, or some way of getting the right information to the right people without having to rely so much upon human indexers or cataloguers. At Project TIP, considerable experience has been gained in a technique called bibliographic coupling, in which the strength of relationships among papers is measured in terms of the number of common citations they possess.

The American Institute of Physics (AIP) is now attempting to devise a scheme of classifying the physics literature in a way which is both useful to physicists and amenable to computer retrieval techniques. During the past year, Dr. Schiminovich of AIP has worked closely with Project TIP personnel devising a scheme to automatically create and modify a classification for the physics literature, and, hopefully, in the long run to automatically classify papers.

The method employed is to begin with a fixed set of papers – the set being relevant to a certain period or topic. This large set of papers is subdivided into groups called "clumps" – a clump being determined by the amount of bibliographic coupling which exists between the papers. The clumps are then analyzed to discover a common subject heading which would be appropriate to the whole clump. Preliminary work of this kind has been reported.*

Project TIP has facilities that lend themselves well to this kind of research. These facilities include a large physics data base, in which citation information is readily available for high-speed processing, and a number of general and special-purpose programs to develop matrices showing the inter-relation of citations between papers.

Special Bibliography Production - Kenneth D. Rude

This past summer, several physicists intended to compile a bibliography on muons. After several weeks it became obvious that the job would take much longer than anticipated. Project TIP was approached, and we assisted by searching the physics literature on CTSS. The work was completed before the summer was over and has been published.†

The search strategy used was essentially this: a sample run was performed to determine which words were relevant other than "muon"; that is, which synonyms, symbols, and phrases might be found in the titles of articles written on the subject of muons. We also used the sample to decide whether or not it would be useful to search the literature for certain author's names. We decided it would not be useful.

Armed with a list of "key words" we searched the physics data base for the years 1963 through 1967. Meanwhile, the physicists were hand processing data not in the TIP data bank, such as specialized journals, Summer School notes, and some Proceedings. As the computer printouts became available, they were screened by the physicists to remove articles deemed not relevant to muon physics. A sample comparison indicated that the computer search missed about one percent of all relevant articles in the journals searched.

---

* N. Price and S. Schiminovich, Information Storage Retrieval, Vol. 4, (1968), pp. 271-280

† L. M. Golub and K. M. Tsipis, Muon Bibliography, MIT Laboratory for Nuclear Science, Technical Report MIT-2098-486, October 1968

Code Conversion - David M. Jordan

TIP's subsystem for converting data between character code systems, CVFILE, has recently been rewritten and improved. The original CTSS design mistake of six and twelve-bit code systems is still with us, so CVFILE continues to work with those modes. More importantly, however, modifications have been made to allow greater flexibility in converting between various nine-bit codes. Since paper-tape machines continue to be the most efficient method of generating large amounts of data, CVFILE uses a table which allows the use of shift characters to switch between tables. A major modification has been made to allow paper-tape output, thus allowing a user to generate a nine-bit code system to suit his own needs. Further improvements have been made in the area of increased control over the output format, with such options as spacing, killing, erasing, and escaping made easily usable.

Because of the large amounts of data generated at TIP, paper-tape input, conversion, and output is a necessity. To help accomplish this, input and output routines have been written for the PDP-7 and the 7094 to allow direct reading and punching of paper tape from the ESL KLUDGE. The difference in cost between console input and paper-tape input is more than a factor of five, so TIP has found that these routines are very economical, and other users with similar problems might find this type of conversion helpful.

Data Adaptation - David M. Jordan

In the past year, we have attempted to establish methods of handling data originating from outside sources. To provide a significant test of these methods, a magnetic tape containing information on graduate students was obtained from the Registrar's office. The information was then transformed twice; first to make it physically acceptable to CTSS, and later to make the data logically usable by TIP subsystems. The resulting data was then used to create a directory of graduate students. The experience indicated that large-scale data conversion and usage is quite practical.

Administrative Information Handling - Edward M. Mattison

Project TIP, in conjunction with the Graduate School Office (GSO), has operated a six-month pilot system for handling information on graduate students. The system contains each student's personal data (name, address, age, etc.), educational background, selective service information, and fellowship and scholarship awards.

The TIP-GSO system has proved capable of producing warrants for fellowship recipients, bursar's typed "requests for payment", and printed lists of data. Other forms of output, which could provide input to information systems in other parts of the institute, are possible. In a related experiment, the student data base maintained by the registrar's office was converted to searchable form compatible with TIP-GSO data.

To make computer manipulation of administrative data a reality on a large scale, revisions must be made in the existing methods of operation. Patterns of information flow must be made more direct; and sources and users of data defined more clearly. Output should be planned with the user in mind. Security procedures to control access to the data, and verification programs to check its accuracy, are imperative. In all these considerations, cooperation among the users is critical, and some method must be devised to share the effort and cost of maintaining these information systems among all the users.

Computer Production of Library Cards - Edward M. Mattison

A trial system for the computer production of library catalog cards has been designed and operated. Input and output for the system is via paper-tape readers and printers. From each item is produced the required set of catalog cards, and a by-product of the system is a machine-readable copy of the catalog card data. Each cataloged work also becomes a TIP data item.

After on-line editing, the text of a card is operated on by the TIP retrieval subsystem. The body text of each card is combined with each of the headings and an ASCII file is written containing a copy of every card required. Cards for author, subject, title, and so forth are automatically produced. This file is then translated back into paper-tape code, a tape is punched from the computer, and the catalog cards are typed on continuous feed forms by a paper-tape printer.

The master file provides an immediately available bibliography. Searches can be made on the call numbers, authors, titles, subject headings, or information contained in the body of the card. Selected book lists and special bibliographies can also be produced. Computer production of catalog cards becomes economically competitive with conventional means of production when the operation provides such usable by-products as a searchable catalog.

Serials and Journals - Patricia M. Sheehan

After two successful computer-produced publications of Current Serials and Journals in the MIT Libraries, the file maintenance programs passed from research status in Project TIP to a production basis in the libraries. CSAJ programs have been modified to handle integration of Wellesley holdings whenever funds become available.

Acquisitions and Accounting - Patricia M. Sheehan

Design effort is now concentrated on library acquisition functions. Up to this point, a desire to keep the volume of data and the range of programs within manageable limits has restricted experiments to serials and journals. It has become quite obvious, however, that to include monographs in the acquisition phase would not hamper the conversion effort but would, in fact, avoid the complications of dual clerical routines. Accordingly, serials, journals, and monographs are handled in a single hybrid system combining standard TIP programs with specialized routines unique to library requirements.

Under the new system, the acquisition flow will be controlled by computer from the point of order, through receipt and payment, to completion of cataloging. (While the actual preparation of catalog cards belongs to another step in the system, a pilot project based on the holdings of the Science Library has indicated its feasibility.) Any of the data, or user-defined sets of data, are accessible through TIP. Regular products include:

1) schedules logging progress and flagging problems,

2) accounting reprints,

3) budget controls,

4) renewal charts, and

5) assorted statistical analyses.

Soft Copy Displays - Lewis H. Morton

A set of programs has been developed to display static pictures on either the ARDS or KLUDGE. The programs use an ASCII character stream to specify the order of picture elements displayed. Nine bits

of information are used for each element; the high-order bits within each byte are interpreted as orientation bits. The left-most bit indicates reflection along a horizontal axis, and the second bit indicates reflection on the vertical axis. Thus any of the 128 possible picture elements may appear in any of four orientations.

At display time, all elements that are to be used are stored in memory in a standard format, and the collection is called a "font". A two-word pointer to each element is placed in a table. In the first word are the code by which the element is known and the "print-width" of the element. "Print-width" is the number of scope positions by which the x position of the beam will be displaced by the element. It is assumed that the element will leave the beam in the same y position. The second word is a pointer to the actual data that makes up the element, in the form of coded lines. The lines are packed one to a computer word, 15 bits each for delta x and delta y. Also contained in this word is a bit indicating whether or not the beam is to be intensified for this line.

Each element must be initially coded as a matrix of points. These matrices are preprocessed into a font and corresponding table. The matrix format is open, allowing any x and y dimension for the element. The matrices are strung together in a file, separated by words of format "777777xxxxxx", where the x's indicate the length of the element description in words. The following word gives the element's ASCII code and print width. Following this is any number of submatrices, each of the following form:

1) The first word gives the x and y coordinate of the upper left-most point in the submatrix;

2) The next word gives X and Y, the width and height of the submatrix, in scope points;

3) Each succeeding word, for Y words, is interpreted as a row in the submatrix;

4) X bits, right justified in the word, will be taken to indicate whether the corresponding point in the submatrix is intensified;

5) Since a word only has 36 bits, larger x counts will cause the submatrix to be filled in with zeros to the left of the 36 bits in the word.

All x and y indications are signed 15-bit quantities, with the x indication in the left half word, and the y in the right. It should be noted that submatrices may overlap to produce the most efficient packing scheme.

The file of all of these matrices is then preprocessed by FONT. This program converts the matrix to line notation, and creates a font table. These are then written into a file for future reference. The command line for this program is:

RUN FONT NAME1

where NAME1 is the first name of the input file NAME1 MATRIX. The output file is NAME1 FONT.

To display an ASCII file, the command line is:

FUN FLASH NAME1 NAME2 FONT LEAD DELTAX DELTAY XZERO YZERO -ROT-

where NAME1 NAME2 is a file containing the ASCII strings to be decoded into pictures. The other arguments all affect picture positioning on the display area.

The program is able to determine the console type at which the user is logged in, and is able to translate the coded lines into appropriate console commands and transmit them to the console. Therefore, the command is identical for all display consoles.

Lines (characters between each set of "new-line" codes) are read from NAME1 NAME2 and given to SRITE, which decodes, interprets the orientation bits, and concentrates the element lines. Two parameters may be given to SRITE indicating the total x length and y length of the resulting subpicture. SRITE will guarantee that the sum of the print widths of the individual elements in the subpicture does not exceed the specified x length. After concatenating the element lines, SRITE adds one more unintensified line which restores the trace to the original x position, but subtracts the given y parameter from the initial y position. Therefore, the net effect of SRITE on the beam position is to lower it by the indicated number of scope positions. LEAD is the y spacing introduced by this subroutine. DELTAX and DELTAY give the total area to be occupied by the display. XZERO and YZERO give the upper leftmost point in the display. And the optional argument ROT will cause the coordinate system to be rotated counterclockwise 90 degrees.

Subsystem Tuning Aids - William D. Mathews

The designer of a subsystem seldom has good information about the total use of the functions he has built. Intuitions about which components are wasting time are usually wrong.

In the past year, Project TIP has developed some fairly sophisticated programs to allow designers to inspect the performance of individual components in the subsystem. Instruction timings and usage profiles can be obtained for all subroutines in a software package.

APPENDICES

Appendix A

Project MAC Memoranda

Appendix B

M. I. T. Thesis

Appendix C

External Publication

Appendix D

Project MAC Technical Reports

APPENDIX A

PROJECT MAC MEMORANDA

MEMORANDUM MAC-M-No.	SUBJECT	AUTHOR	DATE
352	The BCPL Reference Manual	M. Richards	7/21/67
353	The Design and Programming of a Display Interface System Integrating Multi-Access and Satellite Computers (70429-M-190)	D. Ross R. Stotz D. Thornhill C. Lang	7/06/67
354	AED Flash 40: The Top-Down Mouse (APRAL) (70429-M-191)	I. Wenger	7/12/67
355	Recommendations for an Inter-Computer Communication Network for M.I.T.	A. Bhushan R. Stotz J. Ward	7/21/67
356	The ESL Display Console/ PDP-7 System (70429-M-192)	D. Thornhill H. Levin R. Stotz	8/10/67
357	Decomposition of a Visual Scene into Bodies (AI No. 139)	A. Guzman	9/67
358	Perceptrons and Pattern Recognition (AI No. 140)	M. Minsky S. Papert	9/67
359	Revisions to the OPS-3 (CC 279)	R. Berman M. Jones W. Stuart	10/10/67
360	A Fast Parsing Scheme for Hand-Printed Mathematical Expressions (AI No. 145)	W. Martin	10/19/67

MEMORANDUM MAC-M-No.	SUBJECT	AUTHOR	DATE
361	AED Flash 41: 7094-to-360 Conversion using AED (70429-M-194)	C. Feldmann	10/19/67
362	Techniques for the Digital Simulation of Complex Systems	J. Donovan	10/22/67
363	Direction for Updating the AED-0 Programmer's Guide, plus an Index of Terms, and a Description of AED-0 Items (70429-M-198-1)	D. Fulton D. Ross	1/16/68
364	How to Use BCPL on CTSS	M. Richards	2/14/68
365	AED Flash 42: Basic Input/ Output Package (70429-M-199)	C. Feldmann D. Ross	3/07/68
366	Converting McCracken's Algol Book to Describe AED-0 (70429-M-200)	D. Ross D. Fulton	2/14/68
367	A Left-to-Right then Right-to-Left Parsing Algorithm (AI No. 155)	W. Martin	2/68
368	Graphic Output Performance of the ARDS Terminal with the Tektronix Type 611 Storage Display Unit	J. Ward	3/13/68
369	A Program for Drilling Students in Freshman Calculus Integration Problems	J. Moses	3/68

MEMORANDUM MAC-M-No.	SUBJECT	AUTHOR	DATE
371	Numerical Solution of Elliptic Boundary Value Problems by Spline Functions	J. Shah	4/68
372	Some Considerations of Supervisor Program Design for Multiplexed Computer System	F. Corbató J. Saltzer	5/68
373	Running AED on the MIT 360 (70429-M-201)	C. Feldmann	5/02/68
374	Large Capacity Beam Addressable Memories	R. Kay	5/10/68
376	An Interactive Syntax Definition Facility	R. Eanes	6/17/68
377	ITS 1.4 Reference Manual	D. Eastlake	6/68
	....REVISIONS....		
340-2	Telephone Extensions for Dataphones, Teletypes, and 1050's (CC 230-11)	M. Solomita	4/10/68
352-1	The BCPL Reference Manual	M. Richards	2/16/68
363-1	Direction for Updating the AED-0 Programmer's Guide, plus an Index of Terms, and a Description of AED-0 Items (70429-M-198-1)	D. Fulton D. Ross	2/19/68
366.1	Errata sheet to MAC-M-366	D. Ross D. Fulton	2/14/68
373.1	Addendum to MAC-M-373	C. Feldmann	5/27/68

APPENDIX B

M. I. T. THESES

Alsop, J. W., A Canonic Translator, Department of Electrical Engineering, B.S. Thesis, November 1967 (See also MAC-TR-46, Appendix D.)

Avrin, D. E., Computer Display of Protein Electron Density Functions, Department of Electrical Engineering, M.S. Thesis, August 1967

Banks, E. R., On the Simulation of Engineering Multiports, Department of Mechanical Engineering, M. S. Thesis, February 1968

Barkalow, T. J., Student Use of a Time-Sharing System, Department of Electrical Engineering, M.S. Thesis, August 1967

Beltran-Barragen, F., Scattering from Periodic Structures Related to the Insect Corneal Nipple Array, Department of Electrical Engineering, Ph.D. Thesis, February 1968

Chapman, D. G., Audio-Coupled Telephone Transmission of Medium-Speed Digital Data, Department of Electrical Engineering, M.S. Thesis, June 1968

Charniak, E., CARPS, A Program which Solves Calculus Word Problems, Department of Electrical Engineering, M.S. Thesis, June 1968 (See also MAC-TR-51, Appendix D.)

Clark, D. D., A Reductions Analysis System for Parsing PL/I, Department of Electrical Engineering, Ph.D. Thesis, June 1968

Davis, J. A., Computer Models of the Beam-Plasma Interaction, Department of Electrical Engineering, Ph.D. Thesis, June 1968

Denning, P. J., Resource Allocation in Multiprocess Computer Systems, Department of Electrical Engineering, Ph.D. Thesis, June 1968 (See also MAC-TR-50, Appendix D.)

Doyle, J. T., Issues of Undecidability in Canonic Systems, Department of Electrical Engineering, M.S. Thesis, February 1968

Geffner, S. L., Eye-Tracking Program, Department of Mathematics, M.S. Thesis, January 1968

Gold, M. M., A Methodology for Evaluating Time-Shared Computer System Usage, Carnegie-Mellon University Department of Computer Science, Ph.D. Thesis, August 1967

Goodman, L. I., A Loader for Reduction Language, Department of Electrical Engineering, B.S. Thesis, June 1968

Gorman, K. C., The Design of an Asynchronous Arithmetic Unit Permitting Concurrent Computations, Department of Electrical Engineering, M.S. Thesis, June 1968

Hamilton, J. A., Investigation of a Table-Driven Compiler System, Department of Electrical Engineering, M.S. Thesis, June 1968

Hebalkar, P. G., Asynchronous Cooperative Multiprocessing within Multics, Department of electrical Engineering, M.S. Thesis, June 1968

Horn, B. K. P., The Application of Fourier Transform Methods to Image Processing, Department of Electrical Engineering, M.S. Thesis, June 1968

Johnson, T. J. R., An Algorithm for the Resource Constrained Project Scheduling Problem, Sloan School of Management, Ph.D. Thesis, September 9, 1967

Kaliski, M.E., and K. P. Polzen, LOTUS: On-Line Simulation of Block-Diagram Systems, Department of Electrical Engineering, M.S. Thesis, February 1968

Kampe, W. R., II, Rapid Pre-Indexing by Machine, Department of Electrical Engineering, M.S. Thesis, June 1968

Knudsen, M. J., Slides, An Economical Visual-Aids Accessory for Computer-Aided Instruction, Department of Electrical Engineering, M.S. Thesis, June 1968

Kramer, A. N., Automatic Karyotyping of Human Chromosomes, Department of Electrical Engineering, M.S. Thesis, June 1968

Leal-Cantu, N., On the Simulation of Dynamic Systems with Lumped Parameters and Time Delays, Department of Mechanical Engineering, M.S. Thesis, August 1967 (See also MAC-TR-45, Appendix D.)

Luconi, F. L., Asynchronous Computational Structures, Department of Electrical Engineering, Ph.D. Thesis, February 1968 (See also MAC-TR-49, Appendix D.)

Mangano, J., Excitation of the Ion Cyclotron Wave in a Beam-Plasma Discharge, Department of Electrical Engineering, E.E. Thesis, June 1968

Mathison, S. L., and P. M. Walker, Public Policy Issues Arising from Interdependence of Computers and Communications, Sloan School of Management, M.S. Thesis, June 1968

Moses, J., Symbolic Integration, Department of Mathematics, Ph.D. Thesis, December 1967 (See also MAC-TR-47, Appendix D.)

Polzen, K. P., and M. E. Kaliski, LOTUS: On-Line Simulation of Block-Diagram Systems, Department of Electrical Engineering, M.S. Thesis, February 1968

Ribak, R., Subsystem Sharing in Parallel, Asynchronous Processing, Department of Electrical Engineering, M.S. Thesis, June 1968

Rivierre, J. A., Fault-Detection Experiments in Sequential Machines, Department of Electrical Engineering, M.S. Thesis, June 1968

Rodriguez-Bezos, J. E., A Graph Model for Parallel Computations, Department of Electrical Engineering, Sc.D. Thesis, June 1968

Sussman, J. R., Recording System for a Storage Tube Display Terminal, Department of Electrical Engineering, M.S. Thesis, June 1968

Willems, J., Nonlinear Harmonic Analysis, Department of Electrical Engineering, Ph.D. Thesis, June 1968

Walker, P. M., and S. L. Mathison, Public Policy Issues Arising from Interdependence of Computers and Communications, Sloan School of Management, M.S. Thesis, June 1968

Waltz, D. L., A Versatile Electromechanical Hand, Department of Electrical Engineering, M.S. Thesis, February 1968

APPENDIX C

EXTERNAL PUBLICATION

Bartsch, R. R., "Beam-Plasma Discharge: System D", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 169-173

Bernard, G. D., and J. L. Allen, "Superposition Optics — A New Theory", R.L.E. Quarterly Progress Report No. 86, July 1967, pp. 113-121

Bernard, G. D., and W. H. Miller, "Corneal Interference Filters in the Compound Eyes of Flies", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 106-110

Bers, A., "Interactions of Acoustic Waves with Drifting Electrons in a Magnetic Field", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 204-209

Bers, A., "Spontaneous Radiofrequency Emission from Hot-Electron Plasmas — Research Objectives", R.L.E. Quarterly Progress Report No. 88, January 1968, p. 211

Bers, A. and G. Bekefi, "Active Plasma Effects in Solids — Research Objectives", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 201-202

Bers, A. and R. J. Briggs, "Beam Plasma Interactions: Experiments and Theory", R.L.E. Quarterly Progress Report No. 88, January 1968, p. 168

Bers, A. and H. M. Schneider, "Oscillations in an Inhomogeneous Cold Plasma", R.L.E. Quarterly Progress Report No. 89, April 1968, pp. 123-126

Blum, M., "On the Size of Machines", Information and Control, vol. 11, no. 3, September 1967, pp. 257-265

Blum, M. and C. Hewitt, "Automata on a Two-Dimensional Tape", IEEE Conference Record of 1967, Eighth Annual Symposium of Switching and Automata, 18-20 October, pp. 155-160

Briggs, R. J., "Critical Lengths for Absolute Instabilities", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 186-192

Brown, Elaine S., "Gaussian Quadrature — A Numerical Technique for Integration", R. L. E. Quarterly Progress Report No. 87, October 1967, pp. 231-233

Brueck, S. R. J., and A. Bers, "Collisional Effects on Acoustic-Wave Amplification at Microwave Frequencies", R. L. E. Quarterly Progress Report No. 89, April 1968, pp. 156-159

Chou, S-L, and A. Bers, "Thin Electron-Beam Interactions with Ions in a Plasma-filled Waveguide", R. L. E. Quarterly Progress Report No. 87, October 1967, pp. 89-99

Chou, S-L., and A. Bers, "Beam Space-Charge-Wave Interaction with the Backward Wave in a Cold-Plasma Waveguide", R. L. E. Quarterly Progress Report No. 88, January 1968, pp. 183-185

Chou, S-L., and A. Bers, "Further Discussion on Electron Beam Space-Charge-Wave Interaction with the Backward Wave in a Cold-Plasma Waveguide", R. L. E. Quarterly Progress Report No. 89, April 1968, pp. 131-136

Chu, L. J., H. A. Haus, and P. Penfield, Jr., "Electrodynamics of Media — Research Objectives", R. L. E. Quarterly Progress Report No. 88, January 1968, p. 189

Daley, R. C., and J. B. Dennis, "Virtual Memory, Processes, and Sharing in Multics", ACM Symposium on Operating System Principles, 1-4 October 1967

Davis, J. A., "Computer Simulation of the Beam-Plasma Interaction", R. L. E. Quarterly Progress Report No. 86, July 1967, pp. 156-158

Davis, J. A., "Computer Models of the Beam-Plasma Interaction", R. L. E. Quarterly Progress Report No. 87, October 1967, pp. 81-88

Davis, J. A., "Growth of the Lossless, One-Dimensional Beam-Plasma Interaction in Space and Time", R. L. E. Quarterly Progress Report No. 89, April 1968, pp. 137-140

Denning, P. J., "The Working Set Model for Program Behavior", ACM Symposium on Operating System Principles, 1-4 October 1967; also in Communications of the ACM, vol. 11, no. 5, May 1968, pp. 323-334

Dennis, J. B., "A Position Paper on Computing and Communications", ACM Symposium on Operating System Principles, 1-4 October 1967

Dennis, J. B., "Computer Research — Research Objectives and Status of Research", R.L.E. Quarterly Progress Report No. 88, January 1968, p. 349

Dennis, J. B., and R. C. Daley, "Virtual Memory, Processes, and Sharing in Multics", Communications of the ACM, vol. 11, no. 5, May 1968, pp. 306-312

Dertouzos, M. L., "An Introduction to On-Line Circuit Design", Proceedings of the IEEE, vol. 55, no. 11, November 1967, pp. 1961-1971

Donovan, J. J., and H. F. Ledgard, "A Formal System for the Specification of the Syntax and Translation of Computer Languages", AFIPS 1967 Fall Joint Computer Conference Proceedings, vol. 31, 14-16 November 1967

Eden, M., and O. J. Tretiak, "Cognitive Information Processing — Research Objectives and Summary Research", R.L.E. Quarterly Progress Report No. 88, January 1968, p. 309

Engelman, C., "MATHLAB 1968", Proceeding of IFIP 68 Conference, Edinburgh, Scotland

Evans, A., Jr., "PAL - A Language Designed for Teaching Programming Linguistics", Proceedings of the ACM National Conference, 1968

Evans, D. S., and J. Katzenelson, "Data Structure and Man-Machine Communications for Network Problems", Proceedings of the IEEE, vol. 55, no. 3, July 1967

Fano, R. M., "The Place of Time-Sharing", Engineering Education, April 1968, pp. 917-923

Fenichel, R. R., Reviews #11739, 11757, 11812, 11902, 11952, 11953, 11954, 11955, 11956, 12136, 12137, 12370, 14294, Computing Reviews, May-June 1967 through May 1968

Fenichel, R. R., "Algorithm #329: Distribution of Indistinguishable Objects into Distinguishable Slots", Communications of the ACM, May 1968

Feuerzeig, W., and S. Papert, "Programming Languages as a Conceptual Framework for Teaching Mathematics", Proceedings of NATO Conference on C.A.I., April 1968

Forte, A., "Music and Computing: The Present Situation", Proceedings of the Fall Joint Computer Conference, 1967; and Computers and the Humanities, vol. 2, no. 1, September 1967

Goddard, G. W., and R. J. Briggs, "Alternative Stability Analyses", R.L.E. Quarterly Progress Report No. 87, October 1967, pp. 100-105

Gorry, G. A., Sequential Diagnosis by Computer, Sloan School of Management, Working Paper 299-67, March 1968

Gorry, G. A., and G. O. Barnett, "Experience with a Model of Sequential Diagnosis", Computers and Biomedical Research, vol. 1, no. 5, May 1968, pp. 490-507

Gorry, G. A., and F. F. Shapiro, An Adaptive Group Theoretic Algorithm for Integer Programming Problems, Operations Research Center, Technical Report No. 39, M.I.T., May 1968

Graham, R. M., "Protection in an Information Processing Utility", Communications of the ACM, vol. 2, no. 5, May 1968, pp. 365-369

Grassmann, P. H., and O. J. Tretiak, "Differential Leukocyte Analysis by Optical Methods", R.L.E. Quarterly Progress Report No. 89, April 1968, pp. 243-244

Greenblatt, R. D., D. E. Eastlake, and S. D. Crocker, "The Greenblatt Chess Program", AFIPS 1967 Fall Joint Computer Conference Proceedings, vol. 31, 14-16 November 1967

Grochow, J. M., and T. P. Skinner, "An Integrated Disk-Tape Operating System for the 338 Buffered Display Computer", Proceedings of the DECUS Fall Symposium, 10-11 November 1967, pp. 129-135

Gustafson, T. K., and H. A. Haus, "Time-Averaged Energy Functions of Nonlinear Optical Media", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 98-104

Haring, D., "The Beam Pen: A Novel High-Speed, Input/Output Device for Cathode-Ray-Tube Display Systems", AFIPS Conference Proceedings, vol. 27, Part 1, pp. 847-855

Harwitt, Joan, "Modification of the I. B. M. Scientific Package for General Use", R. L. E. Quarterly Progress Report No. 86, July 1967

Haus, H. A., "Quantum Form of Manley-Rowe Relations", R. L. E. Quarterly Progress Report No. 86, July 1967, pp. 105-108

Haus, H. A., "Closed-Form Solution of Steady-State Electromagnetic Shock", R. L. E. Quarterly Progress Report No. 89, April 1968, pp. 65-70

Haus, H. A., and P. Penfield, Jr., "Interpretation of Energy and Power in Uniformly Moving Media", R. L. E. Quarterly Progress Report No. 88, January 1968, pp. 90-92

Haus, H. A., and P. Penfield, Jr., "Force on Magnetic Dipole and Electric Current Loop", R. L. E. Quarterly Progress Report No. 88, January 1968, pp. 93-98

Herba, F., and R. R. Parker, "Minimum-B Mirror Magnetic Field in System A", R. L. E. Quarterly Progress Report No. 89, April 1968, pp. 116-118

Isaacs, Elaine, "Confluent Hypergeometric Function", R. L. E. Quarterly Progress Report No. 86, July 1967, pp. 374-375

Jones, M. M., "On-Line Simulation", Proceedings of the ACM National Meeting, August 1967, pp. 591-599

Karnopp, D. C., and R. C. Rosenberg, "Power Bond Graphs", Control Engineering, May 1968

Karnopp, D. C., and R. C. Rosenberg, Analysis and Simulation of Multiport Systems, M. I. T. Press, Cambridge, Mass., June 1968

Karnopp, D. C., and R. C. Rosenberg, Physical System Dynamics, 16 mm film showing the ENPORT system, 22 minutes, black and white, silent

Kohavi, Z., and P. Lavallee, "Design of Sequential Machines with Faulty Detection Capabilities", IEEE Transactions on Electronic Computers, vol. EC-16, no. 4, August 1967, pp. 473-84

Kohavi, Z. and I. Kohavi, "Further Techniques for the Design of Fault-Detection Experiments for Sequential Machines", Proceedings of Hawaii International Conference on System Science, University of Hawaii, Honolulu, Hawaii, January 1968

Kusse, B. R., and A. Bers, "Interaction of a Spiraling Electron Beam with a Plasma", R.L.E. Quarterly Progress Report No. 86, July 1967, pp. 154-156

Kusse, B. R., "Interactions of a Spiraling Electron Beam with a Plasma", R.L.E. Quarterly Progress Report No. 87, October 1967, pp. 115-120

Kusse, B. R., and A. Bers, "Spiraling Beam-Plasma Interactions", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 175-182

Lapin, R. B ., Translation Between Artificial Programming Languages, Electronic Systems Laboratory Report ESL-R-306, April 1967

Lee, F. F., T. P. Barnwell, and E. R. Jensen, "Reading Machine Studies", R.L.E. Quarterly Progress Report No. 86, July 1967, pp. 305-307

Lee, F. F., and D. E. Troxel, "Cognitive Information Processing — Research Objectives and Summary of Research", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 305-307

Licklider, J. C. R., "Computer Graphics as a Medium of Artistic Expression", Metropolitan Museum of Art Computer Conference, 15-17 April 1968

Licklider, J. C. R., and R. W. Taylor, with E. Herbert, "The Computer as a Communication Device", Science and Technology, April 1968, pp. 21-31

Licklider, J. C. R., "Interactive Information Processing, Retrieval and Transfer", AGARD/NATO Symposium on Storage and Retrieval of Information, Munich, Germany, 18-21 June 1968

Licklider, J. C. R., "Man-Machine Communication", Annual Review of Information Science and Technology, vol. 3, June 1968

Linford, R. K., and L. D. Smullin, "Ion Energy Analysis in a Beam-Plasma Discharge", R.L.E. Quarterly Progress Report No. 89, April 1968, pp. 111-116

Little, J. D. C., and L. M. Lodish, MEDIAC, An On-Line Media Selection System, Sloan School of Management, Working Paper No. 298-67, November 1967

Little, J. D. C., and L. M. Lodish, A Media Selection Calculus, Sloan School of Management, Working Paper No. 304-68, January 1968

Little, J. D. C., Adaptive Control Systems in Marketing, Sloan School of Management, Working Paper No. 342-68, June 1968

Liu, C. L., "A Note on Definite Stochastic Sequential Machines", Proceedings of the Second Annual Princeton Conference on Information Sciences and Systems, March 1968, pp. 223-227

Luccio, F., "A Comment on Index Register Allocation", Communications of the ACM, vol. 10, no. 9, September 1967

Luconi, F. L., "Completely Functional Asynchronous Computational Structures", IEEE Conference Record of 1967 Eighth Annual Symposium on Switching and Automata, 18-20 October 1967

Martin, W. A., and J. H. Griesmer, "Automatically Breaking Mathematical Expressions into Several Lines", IEEE Transactions on Electronic Computers, vol. 10, no. 6, November 1967

Mathews, W. D., "TIP Retrieval System at MIT", Information Retrieval: a Critical View, George Schecter (ed.); based on the Third Annual Colloquium on Information Retrieval, May 1966, Thompson Book Co., 1967

McNaughton, R., "Parenthesis Grammars", Journal of the ACM, vol. 14, no. 3, July 1967

Miller, J. R., DATANAL: An Interpretive Language for On-Line Analysis of Empirical Data, Sloan School of Management, Working Paper No. 275-67, August 1967

Miller, J. R., "On-Line Analysis for Social Scientists", Social Science Information, vol. 7(2), April 1968, pp. 171-191 (See MAC-TR-40, Appendix D.)

Miller, J. R., "Research on DATANAL: An On-Line Computer Language for Data Generation and Analysis", Proceedings of the Research on Management Information Systems, Carnegie-Mellon University Conference, June 1968

Miller, W. H., and G. D. Bernard, "Physical Optics of Invertebrate Eyes — Skipper Glow", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 114-119

Mills, R. G., "Man-Machine Communication and Problem Solving", Annual Review of Information Science and Technology, vol. 11 (Chapter 8), October 1967, pp. 223-254

Ness, D. N., and C. R. Sprague, "Privacy and a National Data Bank", Banking, June 1968, p. 50

Parker, R. R., "Plasmas and Controlled Nuclear Fusion System C", R.L.E. Quarterly Progress Report No. 87, October 1967, pp. 73-80

Parker, R. R., "Plasmas and Controlled Nuclear Fusion, Plasma Heating Experiment", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 167-168

Pennell, M. M., and J. Harwitt, "Polynomial Root Finding", R.L.E. Quarterly Progress Report No. 86, July 1967, pp. 371-372

Platts, D. A., and A. Bers, "Acoustic Wave Amplification-Transport Theory", R.L.E. Quarterly Progress Report No. 89; April 1968, pp. 164-167

River, E. C., and E. C. Isaacs, "Reading Binary Tapes in Undefined Record Format on the 360/65", R.L.E. Quarterly Progress Report No. 87, October 1967, p. 231

Rosenberg, R. C., "A Computer-Simulated Dynamic Systems Laboratory", Computers in Mathematics, Physics, and Engineering Education, NEREM, Boston, Massachusetts, November 1967

Rosenberg, R. C., "On-Line Engineering Analysis at M.I.T., Including a Case Study", The Impact of Computers on Engineering, ASME Winter Annual Meeting, Pittsburgh, Pennsylvania, November 1967

Rosenberg, R. C., "A Low-Cost Output Terminal for Time-Shared Computers", Computer Design, May 1968

Ross, D. T., and J. E. Ward, Investigations in Computer-Aided Design for Numerically Controlled Production, E.S.L. Interim Engineering Progress Report (June 1966 - 30 November 1966), ESL-IR-320, August 1967

Ross, D. T., "The Automated Engineering Design (AED) Approach to Generalized Computer-Aided Design", Proceedings of ACM National Meeting, August 1967

Ross, D. T., "The AED Free Storage Package", Communications of the ACM, vol. 10, no. 8, August 1967

Ross, D. T., "Features Essential for a Workable Algol X", ALGOL Bulletin, no. 26, August 1967; and ACM Sigplan Notices, vol. 2, no. 11, November 1967

Ross, D. T., and J. E. Ward, Investigations in Computer-Aided Design for Numerically Controlled Production E.S.L. Final Report (December 1959-May 1967), Air Force Report No. AFML-TR-68, May 1968

Ruyle, A., J. W. Brackett, and R. Kaplow, "The Status of Systems for On-Line Mathematical Assistance", Proceedings ACM 22nd National Conference, Thompson Book Company, Washington, D.C., 1967, p. 151

Schneider, H. M., "Dynamics of the Plasma Boundary", R.L.E. Quarterly Progress Report No. 86, July 1967, pp. 159-164

Schneider, H. M., "Dynamics of the Plasma Boundary", R.L.E. Quarterly Progress Report No. 87, October 1967, pp. 106-114

Schneider, H. M., "Computer Simulation of an Inhomogeneous Plasma", R.L.E. Quarterly Progress Report No. 89, April 1968, pp. 127-130

Scott-Morton, M., "Interactive Visual Display Systems and Management Problem Solving", Industrial Management Review, Fall 1967

Selwyn, L. L., and D. E. Farrar, "Taxes, Corporate Financial Policy and Return to Investors", National Tax Journal, December 1967

Smullin, L. D., "Plasmas and Controlled Nuclear Fusion — Studies of the Beam-Plasma Discharge", R.L.E. Quarterly Progress Report No. 88, January 1968, p. 167

Speck, C. E., and A. Bers, "Identification of a High-Frequency Micro-instability", R.L.E. Quarterly Progress Report No. 86, July 1967, pp. 195-199

Stevens, K. N., and M. Halle, "Speech Communications — Research Objectives", R.L.E. Quarterly Progress Report No. 88, January 1968, p. 281

Stevens, K. N., "Acoustic Correlations of Place of Articulation for Stop and Fricative Consonants", R.L.E. Quarterly Progress Report No. 89, April 1968, pp. 199-205

Stotz, R. H., "A New Display Terminal", Computer Design, April 1968

Tretiak, O. J., "Field Shading and Noise in the Scad Scanner", R.L.E. Quarterly Progress Report No. 89, April 1968, pp. 227-230

Troxel, D. E., and G. F. Pfister, "Optimum Threshold Level for the Reading Machine Opaque Scanner", R.L.E. Quarterly Progress Report No. 86, July 1967, pp. 303-305

Troxel, D. E., and E. Rosenfeld, "Control of a Reading Machine by the Blind", R.L.E. Quarterly Progress Report No. 87, October 1967, pp. 153-155

Van Horn, E. C., "Three Criteria for Designing Computing Systems", ACM Symposium on Operating System Principles, 1-4 October 1967

Wallace, R. N., "Low-Field Microwave Emission from Contactless Indium Antimonide Samples", R.L.E. Quarterly Progress Report No. 87, October 1967, pp. 121-128

Wallace, R. N., "Effects of Contacts on Low-Field Microwave Emission from Indium Antimonide Loops with Induced Electric Fields", R.L.E. Quarterly Progress Report No. 89, April 1968, pp. 151-155

Weiss, R., "Laser Excitation of the 5s - 6s Forbidden Transition in an Atomic Beam of Rubidium", R.L.E. Quarterly Progress Report No. 88, January 1968, pp. 67-69

Weiss, R., "Gravitation Research — Research Objectives", R.L.E. Quarterly Progress Report No. 88, January 1968, p. 61

Young, I. T., "Biological Image Processing — Automated Leukocyte Recognition", R.L.E. Quarterly Progress Report No. 89, April 1968, pp. 231-242

Zadeh, L. A., "Fuzzy Algorithms", Information and Control, April 1968

Zadeh, L. A., "Computer Science as a Discipline", Journal of Engineering Education, April 1968

Zadeh, L. A., "Probability Measures of Fuzzy Events", Science and Engineering, June 1968

APPENDIX D

PROJECT MAC TECHNICAL REPORTS

REPORT NOS.	DDC NOS.	TITLE	AUTHOR(S)	DATE
MAC-TR-1 (THESIS)	AD-604-730	Natural Language Input for a Computer Problem Solving Language	Bobrow, D.G.	6/64
MAC-TR-2 (THESIS)	AD-608-499	SIR: A Computer Program for Semantic Information Retrieval	Raphael, B.	6/64
MAC-TR-3	AD-608-501	System Requirements for Multiple-Access, Time-Shared Computers	Corbató, F.J.	5/64
MAC-TR-4	AD-604-678	Verbal and Graphical Language for the AED System: A Progress Report	Ross, D.T. Feldman, C.G.	5/64
MAC-TR-6	AD-605-679	STRESS: A Problem-Oriented Language for Structural Engineering	Biggs, J.M. Logcher, R.D.	5/64
MAC-TR-7	AD-604-680	OPL-1: An Open-Ended Programming System within CTSS	Weizenbaum, J.	4/64
MAC-TR-8	AD-604-681	The OPS-1 Manual	Greenberger, M.	5/64
MAC-TR-11	AD-608-500	Program Structure in a Multi-Access Computer	Dennis, J.B.	5/64

REPORT NOS.	DDC NOS.	TITLE	AUTHOR(S)	DATE
MAC-TR-12	AD-609-296	The MAC System: A Progress Report	Fano, R.M.	10/64
MAC-TR-13	AD-609-288	A New Methodology for Computer Simulation	Greenberger, M.	10/64
MAC-TR-14	AD-661-807	Use of CTSS in a Teaching Environment	Roos, D.	11/64
MAC-TR-16	AD-612-702	CTSS Technical Notes	Saltzer, J.H.	3/65
MAC-TR-17	AD-462-158	Time-Sharing on a Multi-Console Computer	Samuel, A.L.	3/65
MAC-TR-18 (THESIS)	AD-470-715	An Analysis of Time-Shared Computer Systems	Scheer, A.L.	6/65
MAC-TR-19 (THESIS)	AD-474-018	A Heuristic Approach to Alternate Routing in a Job Shop	Russo, F.J.	6/65
MAC-TR-20 (THESIS)	AD-474-019	CALCULAID: An On-line System for Algebraic Computation and Analysis	Wantman, M.E.	9/65
MAC-TR-21 (THESIS)	AD-624-943	Queueing Models for File Memory Operation	Denning, P.J.	10/65

REPORT NOS.	DDC NOS.	TITLE	AUTHOR(S)	DATE
MAC-TR-22	AD-625-728	The Priority Problem	Greenberger, M.	11/65
MAC-TR-23	AD-627-537	Programming Semantics for Multiprogrammed Computations	Dennis, J.B. Van Horn, E.C.	12/65
MAC-TR-24	AD-476-443	MAP: A System for On-line Mathematical Analysis	Kaplow, R. Strong, S.L. Brackett, J.W.	1/66
MAC-TR-25 (THESIS)	AD-631-396	Investigation of an Analog Technique to Decrease Pen-Tracking Time in Computer Displays	Stratton, W.D.	3/66
MAC-TR-26 (THESIS)	AD-631-269	Design of a Low-Cost Character Generator for Remote Computer Displays	Cheek, T.B.	3/66
MAC-TR-27 (THESIS)	AD-633-678	OCAS: On-Line Cryptanalytic Aid System	Edwards, D.J.	5/66
MAC-TR-28 (THESIS)	AD-637-215	Input/Output in Time-Shared Segmented, Multiprocessor Systems	Smith, A.A.	6/66
MAC-TR-29 (THESIS)	AD-636-275	Search Procedures Based on Measures of Relatedness Between Documents	Ivie, E.L.	6/66

REPORT NOS.	DDC NOS.	TITLE	AUTHOR(S)	DATE
MAC-TR-30 (THESIS)	AD-635-966	Traffic Control in a Multiplexed Computer Systems	Saltzer, J.H.	7/66
MAC-TR-31 (THESIS)	AD-637-192	Models and Data Structures for Digital Logic Simulation	Smith, D.L.	8/66
MAC-TR-32 (THESIS)	AD-638-446	PILOT: A Step Toward Man-Computer Symbiosis	Tietelman, W.	9/66
MAC-TR-33 (THESIS)	AD-645-660	ADEPT: A Heuristic Program for Proving Theorems of Group Theory	Norton, L.M.	10/66
MAC-TR-34 (THESIS)	AD-650-407	Computer Design for Asynchronously Reproducible Multiprocessing	Van Horn, E.C.	11/66
MAC-TR-35 (THESIS)	AD-657-282	An On-Line System for Algebraic Manipulation	Fenichel, R.	12/66
MAC-TR-36 (THESIS)	AD-657-283	Symbolic Mathematical Laboratory	Martin, W.	1/67
MAC-TR-37 (THESIS)	AD-656-041	Some Aspects of Pattern Recognition by Computer	Guzmān, A.	2/67

REPORT NOS.	DDC NOS.	TITLE	AUTHOR(S)	DATE
MAC-TR-38	AD-662-027	A Low-Cost Output Terminal	Rosenberg, R. Kennedy, D. Humphrey, R.	3/66
MAC-TR-39	AD-661-806	Syntax-Based Analytic Reading of Musical Scores	Forte, A.	4/67
MAC-TR-40	AD-668-009	On-Line Analysis for Social Scientists	Miller, J.	5/67
MAC-TR-41	AD-663-504	Surfaces for Computer-Aided Design of Space Forms	Coons, S.A.	6/67
MAC-TR-42	AD-668-960	Design and Implementation of a Table-Driven Compiler System	Liu, C.L. Chang, G.D. Marks, R.E.	7/67
MAC-TR-43 (THESIS)	AD-662-224	Program Analysis by Digital Computer	Wilde, D.U.	8/67
MAC-TR-44 (THESIS)	AD-662-665	A System for Computer-Aided Diagnosis	Gorry, G.A.	9/67
MAC-TR-45 (THESIS)	AD-663-502	On the Simulation of Dynamic Systems with Lumped Parameters and Time Delays	Leal-Cantu, N.	10/67
MAC-TR-46 (THESIS)	AD-663-503	A Canonic Translator	Alsop, J.W.	11/67

REPORT NOS.	DDC NOS.	TITLE	AUTHOR(S)	DATE
MAC-TR-47 (THESIS)	AD-662-666	Symbolic Integration	Moses, J.	12/67
MAC-TR-48 (THESIS)	AD-662-225	Incremental Simulation on a Time-Shared Computer	Jones, M. M.	1/68
MAC-TR-49 (THESIS)	AD-677-602	Asynchronous Computational Structures	Luconi, F. L.	2/68
MAC-TR-50 (THESIS)	AD-675-554	Resource Allocation in Multi-process Computer Systems	Denning, P. J.	5/68
MAC-TR-51 (THESIS)	AD-673-670	CARPS, A Program Which Solves	Charniak, E.	7/68
MAC-PR-1	AD-465-088	Progress to July 1964	Fano, R. M., et al	
MAC-PR-2	AD-629-494	Progress Report II July 1964 to July 1965	Fano, R. M., et al	
MAC-PR-3	AD-648-346	Progress Report III July 1965 to July 1966	Fano, R. M., et al	
MAC-PR-4	AD-681-342	Progress Report IV July 1966 to July 1967	Fano, R. M., et al	

Cover design by the M.I.T. Office of Publications. Editing and interior design by Project MAC Publications Office. Cold-type composition by Allen Wayne Technical Corporation, New York, New York. Printing and binding by Lew A. Cummings Company, Inc., Manchester, New Hampshire.

AUTHORS INDEX

UNCLASSIFIED
Security Classification

AD 687770

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Massachusetts Institute of Technology
Project MAC

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFED

2b. GROUP
None

3. REPORT TITLE
Project MAC
Progress Report V July 1967 to July 1968

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
annual progress

5. AUTHOR(S) *(Last name, first name, initial)*
Collection of material from many Project MAC participants;
R. M. Fano, Director

6. REPORT DATE
1 July 1968

7a. TOTAL NO. OF PAGES
196

7b. NO. OF REFS
—

8a. CONTRACT OR GRANT NO.
Office of Naval Research, Nonr-4102(01)

b. PROJECT NO.
NR-048-189

c.
RR 003-09-01

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
MAC-PR-5

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. AVAILABILITY/LIMITATION NOTICES
This document has been approved for public release and sale;
its distribution is unlimited.

11. SUPPLEMENTARY NOTES
None

12. SPONSORING MILITARY ACTIVITY
Advanced Research Projects Agency
3D-200 Pentagon
Washington, D.C. 20301

13. ABSTRACT

The broad goal of Project MAC is experimental investigation of new ways in which on-line use of computers can aid people in their individual work, whether research, engineering design, management, or education.

This is the fifth annual Progress Report summarizing the research carried out under the sponsorship of Project MAC. Details of this research may be found in the publications listed in the Appendices at the end of this report.

14. KEY WORDS
Computers
Machine-aided cognition
Multiple-access computers
On-line computers
Real-time computers
Time-sharing
Time-shared computers

DD FORM 1 NOV 68 1473 (M.I.T.)

UNCLASSIFIED
Security Classification

END