DOCUMENT RESUME

ED 196 451 IR 009 090

TITLE	Annual Report, July 1, 1979-June 30, 1980 of the Department of Computer and Information Science, Ohio State University.
INSTITUTION	Ohio State Univ., Columbus. Dept. of Computer and Information Science.
PUB DATE	2 Sep 80
NOTE	90p.: For related document, see ED 171 275.
EDRS PRICE	MF01/PC04 Plus Postage.
DESCRIPTORS	*Academic Education; Annual Reports; *Computer Science; *Departments; Higher Education; *Information Science; *Research
IDENTIFIERS	*Ohio State University

ABSTRACT

This annual report provides information on instructional programs at both the undergraduate and graduate levels and the computing facilities of the Department of Computer and Information Science, and briefly describes the interactions of the department within the university and within the professional community. A list of students awarded doctoral degrees in 1979-80 includes the dissertation title and adviser's name; abstracts of the dissertations are presented in a separate section. Ongoing research projects in five areas of computer and information science--sponsored by both the university and external agencies--are described in some detail, and a bibliography is provided for each area. Appendices include statistical data on the current status and history of the department, a list of courses offered by number and title, information on department faculty and staff members, a list of seminars conducted during the year, and lists of publications, recent technical reports, and activities of faculty and staff. (RAA)

ANNUAL REPORT

JULY 1, 1979 - JUNE 30, 1980

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

THE OHIO STATE UNIVERSITY

COLUMBUS, OHIO 43210

FOREWORD

This publication contains the annual report of the Department of Computer and Information Science and a summary of the research which has been carried on during the 1979-80 academic year. This research has been supported in part by grants from governmental agencies and industry, as well as by The Ohio State University.

The Department of Computer and Information Science is a separate academic unit located administratively in the College of Engineering, operating in part as an interdisciplinary program with the cooperation of many other departments and colleges throughout the University. Under the department is the Computer and Information Science Research Center which is the publishing outlet for a technical report series. Research of the faculty and graduate students in the Department of Computer and Information Science is reported periodically in this series. A bibliography of recent technical reports published by the Center is included in this publication as Appendix F. Copies of some of these reports are still available on a complimentary basis from the Computer and Information Science Research Center, The Ohio State University, 2036 Neil Avenue Mall, Columbus, Ohio 43210. Titles with PB or AD numbers may be obtained from the National Technical Information Service, The U. S. Department of Commerce, 5285 Port Royal Road, Springfield, Virginia, 22161, in paper copy, magnetic tape, or microfiche. Titles with ED numbers may be obtained from the ERIC Document Repooduction Service, P. O. Box 190, Arlington, Virginia, 22210. There is a nominal charge for their service.

Lee J. White, Chairman
Department of Computer
and Information Science
September 1980

TABLE OF CONTENTS

I. THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

Computer and information science deals with the body of knowledge concerned with the quantitative relationships, concepts, theory, and methods common to the processing and utilization of information, and with the theory and operation of the systems which process information. The study of both natural and artificial languages as modes of communication and of natural and artificial systems which process information is fundamental to computer and information science, as is the study of specific systems and specific areas of science and technology relevant to information storage and processing.

The Department of Computer and Information Science is a separate academic unit administratively part of the College of Engineering. In addition, the Department works closely with a number of other departments and colleges throughout the University. Degrees may be obtained in the Colleges of the Arts and Sciences and in the College of Administrative Science in addition to the College of Engineering. The Department has an enrollment of about 200 graduate students and about 500 undergraduate majors.

The program at The Ohio State University emphasizes education, research, and the professional practice and application of computer and information science. The Department offers undergraduate and graduate degrees through the Ph.D. The research activities which are a central part of the program consists of a broad conceptual base supplemented by a number of practically oriented activities. There are numerous sponsors for our research activities. The broad core research program and these other research tasks interact to form an integrated framework.

INSTRUCTIONAL PROGRAMS

The program of the Department of Computer and Information Science is broad and extensive. The number of students enrolled in all courses was 9420. A total of 124 students received baccalaureate degrees, 56 students received the M.S. degree, and 10 students received the Ph.D. degree. The number of applications for graduate study during this period was 509. Seventy graduate students received support from the department. There was a total of 27 full-time faculty and 14 part-time faculty. For additional statistics, see Appendix A.

Undergraduate Programs

Undergraduate degrees in computer and information science are available to students in the College of Engineering, the College of Mathematics and Physical Sciences of the College of Arts and Sciences, and the College

of Administrative Sciences. The particular program chosen depends upon the student's interests and career objectives.

The undergraduate program in the College of Engineering leads to the degree of Bachelor of Science in Computer and Information Science. This program is designed for the student who wants to specialize in computer and information science from within an engineering environment. Hence, the program provides the student with a core of computer and information science, mathematics, and engineering science. Both depth and breadth in computer and information science are assured by specific required course sequences in several areas of engineering and science and yet sufficient flexibility exists so that a student can elect a portion of his technical work in order to develop his individual interests.

There are two undergraduate programs in the College of Mathematics and Physical Sciences. These programs lead either to the degree of Bachelor of Science or the degree of Bachelor of Arts with a major in computer and information science. The programs are cast in a liberal arts setting and are similar in content. The Bachelor of Science program provides a somewhat more technical and thorough education in computer and information science and mathematics while the Bachelor of Arts program is somewhat more flexible and provides an opportunity to relate computer and information science to some other discipline.

The undergraduate program in the College of Administrative Science leads to the degree of Bachelor of Science in Business Administration with a major in computer and information science. This program is designed for the business-oriented student who desires both an education in computer and information science and a general education in the administrative sciences. The program's objective is not to make a computer specialist out of a student, but rather to enable him to recognize the opportunities to use the computer in his managerial activities, to know what to expect from it, and to know how to communicate effectively with computer specialists so that computerized projects will be properly handled from a technical as well as a managerial point of view.

Graduate Programs

The Department offers programs leading to both master's and Ph.D. degrees.

General Requirements

Students should be able to complete a master's degree in one year of full time study (four quarters). A student will normally take a total of four years to complete a Ph.D. program.

Each student is expected to take a course of study corresponding to one of the following nine options.

OPTION I — for the student desiring a theoretical foundation in computer and information science.

OPTION II	for the student specializing in information systems.
OPTION III	for the student specializing in computer systems.
OPTION IV	for the student specializing in numerical analysis.
OPTION V	for the student specializing in operations research.
OPTION VI	for the student specializing in biomedical information processing.
OPTION VII	for the student specializing in administrative science.
OPTION VIII	for the student specializing in mathematics.
OPTION IX	for students specializing in computer hardware and software who have appropriate undergraduate background.

Each of these options provides a background in several aspects of computer and information science, as well as additional mathematical sophistication appropriate to the student's interest. Each of the options may lead to the doctoral program in computer and information science or to the master's degree.

The Master of Science degree may be considered to be either a terminal degree leading to the professional practice and application of one phase or another of computer and information science or it may be considered as the first step towards the Ph.D. degree.

The Core Program

All courses of study require the completion of a core program in computer and information science which consists of the following.

Course			Credit
CIS 680	-	Data Structures	5
CIS 707	-	Mathematical Foundations of Computer and Information Science II	3
CIS 755*	-	Programming Languages	3*
CIS 760*	-	Operating Systems	3
CIS 775*	-	Computer Architecture	3
CIS 885*	-	Seminar on Research Topics in Computer and Information Science	1

Course			Credit
CIS 889	-	Advanced Seminar in Computer and Information Science	2
TOTAL CREDIT HOURS IN CORE			20*

*Course number and credit approved for Autumn 1980.

Master of Science Program

Suggested courses of study, which complete each of the options, consist of additional electives in computer and information science, mathematics and cognate areas. The minimum number of credit hours required for the master's degree is 48 credits for Plan A (with thesis) or 53 credits for Plan B (without thesis). Certain options of the M.S. program may require more than these minima. Every candidate on Plan A is required to write an M.S. thesis and successfully defend that thesis in a final examination while those on Plan B must demonstrate their mastery of the fundamentals of computer and information science by passing the M.S. Comprehensive Examination. However, a student who has passed the Ph.D. General Examination is eligible to receive the master's degree without having to satisfy either of the above requirements, and students planning to study for the Ph.D. are encouraged to obtain the M.S. degree in this manner. In the Comprehensive Examination, the student will be examined on the content of the core courses and one of the nine M.S. options.

Joint Master of Science Program with Mathematics

A special program is available so that a student may receive two master's degrees, one in mathematics and one in computer and information science, after completing 76 quarter hours of course work. Further information about the joint program may be obtained by request.

Doctoral Program

The award of the Ph.D. degree implies that the recipient achieved a mastery of a subject which allows him to work in a particular field in a creative capacity and to stimulate others working in this area. The Qualifying, General, and Final Examinations, taken by the student at various stages of his doctoral studies enable the faculty to ensure that only students of outstanding scholastic ability continue on to receive the doctoral degree.

The doctoral program emphasizes research and the Department encourages prospective Ph.D. candidates to involve themselves in research under the supervision of a faculty member at the earliest possible opportunity.

A major area is generally chosen from about fourteen active areas of faculty research. It will normally be the area in which the student expects to perform dissertation research. In fact, the General Examination is designed, among other things, to ensure the student's readiness to undertake dissertation research.

Two of the fourteen or one of the fourteen and a cognate area may be elected for the minor areas of specialization. A cognate field is defined as a field supporting or closely related to the fourteen Departmental fields and is ordinarily specified by an integrated program of study in other departments of the University. Note, however, that the Ph.D. program can be very flexible and suited to the interests of each individual student. Hence, it is possible to choose areas of specialization outside of the fourteen areas, subject only to academic relevance and approval by the graduate committee.

The General Examination is usually taken in about the 9th quarter of residence and consists of appropriate written and oral portions. The second stage is completed and the student admitted to candidacy when he has received credit for a total of at least 90 quarter hours of graduate work and passed the General Examination.

The third stage, after admission to candidacy, is devoted primarily to research and seminars, the preparation of the dissertation, and the Final Examination. The Final Examination is oral and deals intensively with the portion of the candidate's field of specialization in which his dissertation falls.

The Department does not have a foreign language requirement for the M.S. or Ph.D. degree.

Course Offerings

Currently there are about 90 courses (each one quarter in length) offered by the Department, 25 of which are largely undergraduate with the remainder being primarily graduate courses. In addition to these courses there are over two hundred courses offered by a variety of departments of the University which are of interest to our graduate students who are encouraged to take these courses. See Appendix B for a listing of courses by number and title.

Continuing Education

Occasionally students register with Continuing Education in order to make up deficiencies that prevent them from being accepted in the department's graduate program. From time to time the department also uses the Continuing Education program to present new developments, usually for practicing professionals.

Faculty

The Department of Computer and Information Science has a full time faculty with a wide range of backgrounds and experience. The faculty is supplemented by staff who have joint appointments with other departments; by staff from other departments or by visiting faculty who teach courses primarily for Computer and Information Science students; by adjunct staff people who are employed in off-campus organizations who teach in the Department of Computer and Information Science; (See Appendix C).

COMPUTING FACILITIES

Computer Centers

There are three computer centers at The Ohio State University. They are: Instruction and Research Computer Center; Hospital Computer Center; and University Systems.

Included in these centers are an AMDAHL 470 V6, IBM 370/168, IBM 370/158 and an IBM 1620, as well as a number of remote batch terminals and remote video and typewriter-like communication terminals for use in the various on-line and time sharing systems.

Instruction and Research Computer Center/Computer and Information Science (IRCC/CIS) Computing Laboratory

The principal research resource of the Department of Computer and Information Science is the Instruction and Research Computer Center/Computer and Information Science (IRCC/CIS) Computing Laboratory. The Laboratory was specifically designed to serve the specialized needs of problem oriented research and instruction impacting on the computer and information sciences. Consequently, it is maintained as a state-of-the-art facility for research and instructional programs in which the computing process is an object of study or is directly and actively involved as an integral element of problem formulation and solution.

The IRCC/CIS Computing Laboratory is administered and operated by the Instruction and Research Computer Center separately from the Center's main service installation. The Laboratory, thus, is maintained primarily for the Department of Computer and Information Science. This arrangement, unique among major universities, permits the Laboratory to be dedicated to research and instruction in the computer and information sciences.

The principal computer of the Laboratory is a DECsystem 20/20 time-sharing system, produced and maintained by the Digital Equipment Corporation, with the following features:

- TOPS-20 operating system
- 256K words of virtual address space for each user
- Two disk drives with a total on-line capacity of 55 million words of storage
- Tape drive capable of processing standard 1/2 inch magnetic tape at 800 or 1600 bpi densities
- A variety of CRT and hardcopy terminals and printers
- Direct-wired high-speed lines and remote dial-up lines
- High-speed paper tape reader/punch unit
- Several graphics peripherals, including several TEKTRONIX display devices and a remotely coupled AG-60 Plasma Graphics Panel.

DEC-supported compilers for the DECsystem 20/20 include FORTRAN, BASIC-PLUS, and ALGOL. DEC also provides a variety of software packages for program development and debugging, and for test editing and production. Locally-supported languages include PASCAL, LISP, SNOBOL, and BLISS. Additional software is available for support of other processors: the PDP-8 and PDP-11, the 6800 and 8080 microprocessors, and the MICRODATA 1600 computer.

A six-node fault-tolerant, double-loop network is presently under construction by the CIS Department, and will constitute an important part of the CIS Computing Laboratory. The network configuration includes a host computer node consisting of the DECsystem 20/20; each of the other five nodes will consist of a DEC 11/23 microcomputer; since each 11/23 unit will be complemented by 128K of MOS memory, a UNIX operating system, a CRT terminal, and a dual floppy disk, it can be operated stand-alone as well as a network resource to be shared. Each node of this network will be equipped with a loop-interface unit, presently under design and development by the Department.

This network will be available for a number of research projects beginning with the 1980-81 academic year. Research in networking techniques, distributed processing hardware and software configurations, parallel processing algorithm development, and distributed data bases can be conducted on a network system available for experimental studies. Research into a number of issues of software engineering, including reliability of distributed software, computer program testing, and distributed system language development can be conducted using a realistic and flexible computer network setting.

INTERACTION WITHIN THE UNIVERSITY

The Department of Computer and Information Science interacts with other departments and research programs within the University because of the multidisciplinary nature of the activities encompassed in this field. A number of the academic faculty have joint appointments in other departments. Staff members of the Department of Computer and Information Science have appointments in the following departments and organizations:

a. Accounting
b. Allied Medicine
c. Art
d. Electrical Engineering
e. Engineering
f. Instruction and Research Computer Center
g. Mathematics
h. Psychology
i. University Libraries
j. University Systems Computer Center

INTERACTION WITHIN THE COMPUTER AND INFORMATION SCIENCE COMMUNITY

Columbus, Ohio, is one of the major centers for information science and for the transfer of information in the United States. A number of organizations are involved with the activities of computer and information science. This affords an opportunity for students and faculty to interact with appropriate personnel in these organizations. Some of these are:

a. Chemical Abstracts Service
b. Battelle Memorial Institute
c. Bell Laboratories
d. City National Bank
e. Columbus and Southern Ohio Electric Company
f. Western Electric Corporation
g. Rockwell International Corp.
h. Industrial Nucleonics
i. State of Ohio Department of Finance; Department of Highways
j. Columbus Board of Education
k. Ohio College Library Center

There are a large number of scientists who come to Columbus in order to visit the Department and who usually present a seminar. The lectures and seminars for the period of this report are listed in Appendix D.

Research efforts of the staff are disseminated to the professional community through several publication channels. A list of current publications of the Department staff is included as Appendix E. In addition, the Research Center issues a technical report series (see Appendix F for reports issued from 1978 to date). Our faculty attends most of the major technical meetings in this country as participants giving papers, assisting on panels, as attendees, and as officials. A list of these activities can be found in Appendix G.

DOCTOR OF PHILOSOPHY DEGREE

The Doctor of Philosophy degree was awarded to the following students during 1979-80. Abstracts of these dissertations are included on pages 46-53 See Appendix H for a complete list of doctorates awarded.

Name	Dissertation	Advisors
Baker, Albert L.	Software Science and Program Complexity Measures	Zweben (Buttelmann)
Flinchbaugh, Bruce	A Computational Theory of Spatio-Temporal Aggregation for Visual Analysis of Objects in Dynamic Environments	Chandrasekaran
Jappinen, Harry	A Perception-Based Developmental Skill Acquisition System	Chandrasekaran
Ko, Ker-I	Computational Complexity of Real Functions and Polynomial Time Approximation	Moore (Buttelmann)
Kwasny, Stan C.	Treatment of Ungrammatical and Extra-Grammatical Phenomena in Natural Language Understanding Systems	Sondheimer (Buttelmann)
Mellby, John R.	The Recognition of Straight Line Patterns by Bus Automata Using Parallel Processing	Rothstein

Name	Dissertation	Advisors
Pardo, Roberto	Interprocess Communication and Synchronization for Distributed Systems	Liu
Teng, Albert Y.	Protocol Constructions for Communication Networks	Liu
Wolf, Jacob J., III	Design and Analysis of the Distributed Double-Loop Computer Network (DDLCN)	Liu
Wong, Patrick M.	A Methodology for the Definition of Data Base Workloads: An Extension of the IPSS Methodology	DeLutis

II. RESEARCH IN COMPUTER AND INFORMATION SCIENCE

RESEARCH PROGRAMS

1. RESEARCH IN COMPUTER PROGRAM TESTING

Faculty Lee J. White
B. Chandrasekaran
Stuart H. Zweben

Students Fernando J. Gomez
Allen W. Haley
Steven J. Zeil

For the past three years, this research group has been working in the area of reliable software in general, and program testing in particular. We have developed an automated testing strategy called the Domain Testing Strategy [1-5] which is very promising for a large class of data processing programs. This method is a form of path analysis strategy, where the process of testing is treated as two operations [6,9,10]:

1) selection of a path or set of paths along which testing is to be conducted; and
2) selection of input data to serve as test cases which will cause the chosen paths to be executed.

For general programs, the problem of detection of reliable test data is known to be unsolvable. For certain classes of programs, however, the Domain Testing Strategy research has shown that it is possible to implement reliable methods of selecting test data for a given path to detect certain types of errors. This research has been supported by the Air Force Office of Scientific Rese search under Grant AFOSR 77-3416.

Another line of research currently being investigated is an approach to program testing based on modularity. In developing the solution to a large complex problem, it is customary to partition the problem into functional units, each of which can be implemented as a short length of computer code or module. These modules can then be refined, implemented, and tested as independent units of the total system and then integrated to form a complete working solution to the overall problem. Research is now underway to establish the extent to which independent module testing can reduce the amount of integrated testing required when the modules are interconnected to form the entire system.

The Domain Testing Strategy comprised a method for automatically generating test data to reliably test a given path, but gave no indication as to how that path should be selected. Recently a method called sufficient testing [11] has been developed which gives some information in selecting paths to be tested. The types of questions addressed are:

(1) After a number of paths have been tested, what is the marginal advantage of choosing yet another test path?

(2) Is there a point at which we may say that no more paths need be chosen through some program construct, i.e., that it has been sufficiently tested?

A set of paths shall be considered a sufficient set for a program construct if the failure to detect some error in that construct, using a reliable method of selecting data points along these paths, implies that this error would go undetected for any path through the program.

The last aspect of this research involves an approach called conceptual programming, in which algorithms are generated for the computer at a conceptual level appropriate for the problem being described. This approach is being applied to the problem of program specification, where the object is to write and process specifications at a high level of abstraction and produce code in a lower level language.

This research is currently being supported by the Air Force Office of Scientific Research under contract F49620-79-0152.

A Domain Strategy for Computer Program Testing

Computer programs contain two types of errors which have been identified as computation errors and domain errors [1-5,8]. A domain error occurs when a specific input follows the wrong path due to an error in the control flow of the program. A path contains a computation error when a specific input follows the correct path, but an error in some assignment statement causes the wrong function to be computed for one or more of the output variables. A testing strategy has been designed to detect domain errors, and the conditions under which this strategy is reliable are given and characterized. A by-product of this domain strategy is a partial ability to detect computation errors. It is the objective of this study to provide an analytical foundation upon which to base practical testing implementations.

There are limitations inherent to any testing strategy, and these also constrain the proposed domain strategy. One such limitation might be termed coincidental correctness, which occurs when a specific test point follows an incorrect path, and yet the output variables coincidentally are the same as if that test point were to follow the correct path. This test point would then be of no assistance in the detection of the domain error which caused the control flow change. Another inherent testing limitation has been previously identified as a missing path error, in which a required predicate does not appear in the given program to be tested. Especially if this predicate were an equality, no testing strategy could systematically determine that such a predicate should be present.

The control flow statements in a computer program partition the input space into a set of mutually exclusive domains, each of which corresponds to a particular program path and consists of input data points which cause that path to be executed. The testing strategy generates test points to examine the boundaries of a domain to detect whether a domain error has occurred, as either one or more of these boundaries will have shifted or else the corresponding predicate relational operator has changed. If test points can be chosen within ε of each boundary, the stragety is shown to be reliable in detecting domain errors of magnitude greater than ε, subject to

the following assumptions:

(1) coincidental correctness does not occur;

(2) missing path errors do not occur;

(3) predicates are linear in the input variables;

(4) the input space is continuous.

Assumptions (1) and (2) have been shown to be inherent to the testing process, and cannot be entirely eliminated. However, recognition of these potential problems can lead to improved testing techniques. The domain testing method has been shown to be applicable for nonlinear boundaries, but the number of required test points may become inordinate and there are complex problems associated with processing nonlinear boundaries in higher dimensions. The continuous input space assumption is not really a limitation of the proposed testing method, but allows the parameter ε to be chosen arbitrarily small. An error analysis for discrete spaces is available [2], and the testing strategy has been proved viable as long as the size of the domain is not comparable to the discrete resolution of the space.

Next let us consider two further assumptions:

(5) predicates are simple; and

(6) adjacent domains compute different functions.

If assumptions (5) and (6) are imposed, the testing strategy is considerably simplified, as no more than one domain need be examined at one time in order to select test points. Moreover, the number of test points required to test each domain grows linearly with both the dimensionality of the input space and the number of predicates along the path being tested. Any program which satisfies these six constraints will be referred to as a linearly domained program (see the section of this article on sufficient testing).

The domain strategy is currently being implemented, and will be utilized as an experimental facility for subsequent research. A most important contribution would be to indicate both programming language constructs and programming techniques which are easier to test, and thus produce more reliable software. The current system will analyze input programs in PL/C and FORTRAN, and as the user interactively specifies paths to be tested, the system will automatically generate a set of reliable test data which then can be applied to the given program.

An Approach to Program Testing Based on Modularity

A serious difficulty with the current state of the Domain Testing Strategy is that the number of test points, while finite, still grow too fast with the number of predicates, or more generally, program size. This makes the strategy of limited use in testing large programs. Further, modular development is often the rule in the development of large software systems. Thus

one would like to be able to test modules separately and, with only an incremental amount of testing overhead, test the total system composed of modules. If the Domain Testing Strategy can be extended to cover modular testing, then growth in the number of test points can also be kept under control.

Towards this end we have begun investigating the conditions under which a module can be tested independently of other modules. Our current results can be summarized as follows. Assume a program can be divided into "blocks" of code and control structure where each block is single-entry, single exit. The following definitions are needed.

D1. Given two blocks A and B, block B is said to be independent of block A, if, irrespective of which path is taken through block A, the path taken through block B will remain unchanged.

D2. If a block A is independent of all blocks which logically precede A, then block A is said to be "completely independent".

D3. If a block, say A, is independent of at least one block which logically precedes it, then block A is said to be "partially independent".

Theorem 1. If a block A is completely independent then the domain structure of block A can be completely tested provided only that the control environment is specified in terms of the input variables.

Theorem 2. If a block A is partially independent, then the domain structure of block A can be tested regardless of the paths taken through blocks of which A is independent.

In order to appreciate the extent of reduction in testing possible with a modular approach, consider a program P consisting of subprogram P1 containing m paths followed by subprogram P2 containing n paths. The integrated program can have a total of m * n paths (see Figure 1 for m=3 and n=4), since any of the m paths in P1 can be followed by any of the n paths in P2.

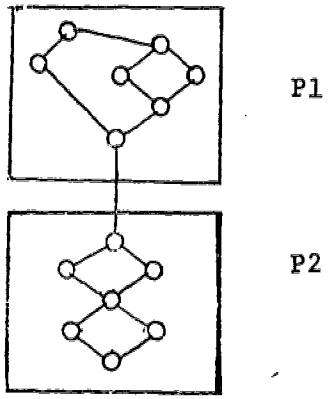


Figure 1. Integration of Subprograms with 3 and 4 paths, respectively.

In the course of developing P however, it is probably the case that both P1 and P2 have been tested separately. It would be desirable if the correctness information obtained in unit testing P1 and P2 could be used in validating P. If the individual modules do not contain a large number of paths, it may be possible to test all possible paths in each module. If the additional testing required at integration time was negligible compared to the unit testing overhead, the result would be a reduction of the magnitude of the testing problem from O(m*n) to O(m+n). While this represents in some sense an ideal situation, it is clear that with such a potential for complexity reduction, even a less than ideal solution might represent a considerable improvement and yet provide a substantial degree of practicality.

This research seeks to explore the extent to which module testing can be helpful in reducing the complexities of testing large programs. The investigation is intended to be both of a general nature and applied to the Domain Testing Strategy in particular.

Sufficient Test Sets for Path Analysis

It is known that the problem of automatically generating reliable test data is unsolvable. All questions of practicality aside, such a claim runs counter to the intuition of the typical programmer who is quite willing to infer the correctness of his program from a small, finite number of test paths. It is the goal of this research to show that, when testing for errors in program predicates, this confidence is not misplaced.

In linearly domained programs, the program predicates and the computations affecting control flow are linear in the input variables. Although linearity itself yields considerable simplification, another implication of this assumption is conceptually more important. Restricting predicate interpretations to a well-behaved functional class makes possible the description of the infinite set of possible predicate errors using a small finite set of linearly independent errors.

In this research we have used this approach to characterize those predicate errors which must escape detection for a given test path in a linearly domained program. This characterization has led to criteria for determining whether a proposed test path is capable of detecting any errors not already revealed by previous tests. These criteria are directly derivable from the assignments and equality predicates encountered along the test paths. The value of a test path is defined in terms of its ability to eliminate one or more of the characteristic errors which had escaped previous tests.

Specifically, this research has thus far established the following results about linearly domained programs [11]:

1) we can ascertain whether any error in a given predicate can be detected using a specific path containing that predicate;

2) we can ascertain whether a proposed path ending with a given predicate need be tested if we have already tested some subset of paths ending with that predicate;

3) a minimal set of subpaths sufficient for testing a given predicate will contain at most m+n+1 subpaths, where m is the number of input values and n the number of program variables. This limit is independent of the complexity of the program control flow.

Moreover, in nearly all cases this limit should be very conservative, as many factors will act so as to reduce the number of required paths to be tested.

These results do not constitute a method for selecting paths for testing. The question of which paths are to be examined under this criterion has not yet been addressed. However, it seems unlikely that the more general question of how to select paths for testing can be answered without some means of judging the path's value to the testing process. Such a means has now been provided, together with the assurance that only a finite number of paths need to be selected.

The model employed does not take program structure into account. It remains to be seen what effects, if any, the use of well-structured control constructs might have on the selection of sufficient sets of test paths. Work is continuing on this model, focusing on the extension of the analysis for linearly domained programs to domain errors caused by incorrect computations and on the applicability of these results to path selection strategies.

Conceptual Programming

While it is hard enough to verify algorithms at a conceptual level, added complexity results in program testing when one takes as the object of testing a program coded in a lower level programming language. A completely different approach to producing more reliable programs is taken in attempts to design program synthesis systems - which typically take specifications in a high level of abstraction and produce code in a lower level language. In addition to being simpler to program in this manner, this approach can also reduce the errors associated with the production of code.

The approach we are beginning to take for this problem can be characterized as conceptual programming, in the sense that the algorithms are given to the computer at a conceptual level which is natural to the algorithm being described. It can also be viewed as natural language programming, since natural language is the embodiment for such concepts. The overhead for parsing and dealing with many other aspects of natural language can be minimized if the underlying structure of concepts is properly clarified. In fact, very little complex parsing seems to be needed, if the conceptual structure is suitably realized.

Currently the following preliminary work has been completed. An analysis of the design requirements of the limited natural language front-end has been undertaken. The conceptual structure underlying the programming domain, as applicable to common data structures such as lists, arrays, stacks, etc., has been obtained. The overwhelming constraint is that, in a

deep sense, the complexity of the synthesis procedure must be limited. Many current approaches to program synthesis run aground when this limitation is imposed. Our thesis is that this complexity can be controlled only by a proper understanding of the structure of concepts which constitute an "understanding" of the domain.

References

[1] L.J. White, E.I. Cohen, B. Chandrasekaran, "A Domain Testing Strategy for Computer Program Testing," OSU-CIS Research Center Technical Report TR-78-4, August 1978.

[2] L.J. White, F.C. Teng, H.C. Kuo, D.W. Coleman, "An Error Analysis of the Domain Testing Strategy," OSU-CIS Research Center Technical Report TR-78-2, December 1978.

[3] L.J. White and E.I. Cohen, "A Domain Testing Strategy for Computer Program Testing," Infotech State of the Art Report, "Software Testing," 1978, Volumes I and II.

[4] L.J. White, E.I. Cohen, and B. Chandrasekaran, "Discussion of 'A Survey of Program Testing Issues' by John B. Goodenough," discussant item in Recent Directions in Software Technology, MIT Press, 1979.

[5] E.I. Cohen, A Finite Domain-Testing Strategy for Computer Program Testing, Ph.D. Dissertation, 1978, Ohio State University.

[6] L.A. Clarke, "Automatic Test Data Selection Techniques", in Infotech State of the Art Report on Software Testing, Vol. 2, 1979.

[7] J.B. Goodenough and S.L. Gerhart, "Toward a Theory of Test Data Selection," IEEE Transactions on Software Engineering, Vol. SE-1, No. 2, pp. 156-173, June 1975.

[8] W.E. Howden, "Reliability of the Path Analysis Testing Strategy", IEEE Transactions on Software Engineering, Vol. SE-2, No. 3, pp. 208-215, September 1976.

[9] C.V. Ramamoorthy, S.F. Ho, and W.T. Chen, "On the Automated Generation of Program Test Data," IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, pp. 293-300, December 1976.

[10] L.J. White and E.I. Cohen, "A Domain Strategy for Computer Program Testing," IEEE Transactions on Software Engineering, Vol. SE-6, No. 3, pp. 247-257, May 1980.

[11] S.J. Zeil and L.J. White, "Sufficient Test Sets for Path Analysis Testing Strategies," submitted to the 5th International Conference on Software Engineering, 1980; also OSU-CIS Research Center Technical Report TR-80-6, July 1980.

2. RESEARCH IN PROGRAMMING ENVIRONMENTS

Faculty Jayashree Ramanathan

Students Hongchik J. Kuo
Charles J. Shubra
Dilip Soni

The TRIAD project on program development systems (pds) was initiated in the Fall of 1979 and has as its general goal the development of a friendly programming environment. The activities of the research group can be summarized as follows.

1) The precise characteristics of the ideal pds which provides support for developing reliable programs have been determined. These characteristics distinguish a pds from standard translators (for languages such as FORTRAN, COBOL, PL/I), text editors and other methodologies, and identify a pds as a more powerful and desirable tool.

2) A formal model, called attributed metaforms, which serves as the organizational force underlying the design and implementation of such a pds has been proposed.

3) The usefulness of this proposed model is being carefully analyzed. We have determined that this model is in fact adequate to facilitate the pds functions.

4) A preliminary implementation of a pds has begun. This pds is called a Tree based Interactive program Analyzer and Developer (or TRIAD). Various research issues are being formulated and addressed based on this preliminary design implementation phase.

The subsections below give details of the motivation for the project, the description of the project and the short and long term research goals.

Motivation

Though programming is essentially a creative process, it has been realized by various researchers and practitioners [see, for example, 3, 4, 5, 12, 15] that many aspects of program development can be automated to aid in developing programs that are reliable and maintainable. Such semi-automatic tools may be generally termed as program development systems and are expected to bridge the gap between 'manual' programming, where the programmer applies a methodology to develop a program in the absence of computer aids, and completely automated programming, which is an area of active research in Artificial Intelligence.

A pds differs from popularly used compilers, for langauges such as COBOL, FORTRAN, and PL/I in several ways. Statements in these languages primarily specify operations on data. The compiler's analysis is restricted to the final product -- the program. Newer, general purpose languages such as ADA [4], ALPHARD [13], CLU [9], EUCLID [10], and PASCAL [8] provide

linguistic mechanisms (such as case statements and data abstractions), which in addition, organize the data and operations in some way. Thus, these mechanisms implicitly guide the programming process so that the program statements specify operations on data and are also reliable and maintainable to some extent. However, program development itself is arbitrary and no explicit history of this development is retained to aid in program development and maintenance.

There has also been a recent emphasis on program development methodologies with associated notations. Examples of some of the more popular methodologies are HIPO [14], JACKSON [7], WARNIER [16], SADT [12], and PSL/PSA [15]. These methodologies attempt to provide more explicit guidance for the programmer's thought process during program development. However, by and large, each methodology is very general and addresses many different applications or problem domains. As a result, methodologies are not very effective in aiding the programmer's thought process by reducing the design choices in any specific problem domain. The proliferation of methodologies stems from a need to have methodologies specifically tailored for different problem domains and, therefore, be more effective in these domains. However, most methodologies do not precisely identify the problem domains for which they are well suited. In addition, most methodologies are not accompanied by software tools which analyze the program development process as well as the finished product which is the program.

The ideal pds should be able to first guide program development according to a methodology and then compile the final, developed program. The pds should analyze the development process as well as the final program and provide useful messages for the programmer during all stages. The methodology, used by the pds, should explicitly and effectively aid in the thought process for developing programs in a precisely defined, restricted problem domain.

The design and implementation of such a pds is a new area for research. New research issues are raised:

1) Grammar based translation theories are inadequate because they only address the final product - the program - and do not address program development.

2) A concise notation for describing the methodologies must be developed. This description must drive the pds so that the development and analysis of the program are tuned to the methodology.

3) A library of methodologies, each designed to provide aid within a problem domain must be created.

4) Not much is known about programming in various problem domains though our intuition and collective knowledge points to the existence of several well defined problem domains. Some problem domains of interest to us fall into the following general categories:

   a) distributed programming
   b) real-time programming
   c) data oriented programming
   d) simulations, etc.....

TRIAD Description

We are currently involved in the design and implementation of a pds called TRIAD. The three major components of TRIAD are a set of tools including the analyzer, a data base of system maintained program components and a data base for the development histories of programs. The programmer will use the TRIAD facilities by using a command language which is interpreted by an executive. (See Figure 1)

In TRIAD we use attributed grammar forms [6,2] to model program development according to a particular methodology. The TRIAD analyzer is facilitated by the use of a precise description of program development based on a methodology. TRIAD also maintains a history of a program from its abstract, functional form to its concrete, implementation form. This history (or refinement tree) reflects a methodology description denoted as an attributed grammar form of the programmer's choice. Each refinement tree, therefore, belongs to a family of trees reflecting a methodology and the analysis routines of TRIAD exploit this knowledge to provide useful messages during all stages of program development and program modification. The effectiveness of the TRIAD analyzer, in providing useful feedback to the programmer during the development process, is entirely based on the fact that the program development itself can be systematically represented in a machine analyzable form. The attributes and rules for evaluating these attributes in the attributed grammar forms permit the analyzer to generate messages based on a global data flow analysis of the program's refinement tree.

The model is powerful enough and arises naturally just as grammars are very natural for modeling the translation process. We have illustrated this by examining how development, using some of the more complex existing methodologies, such as JACKSON [7], SADT [2], and PSL/PSA [17], can be modeled. Figure 2a, 2b, and 2c give an intuitive illustration of the use of the model to describe a methodology and model program development. The methodology description could, of course, be previously created by a project supervisor to effectively aid in the development of programs specific to an application area. We are examining existing methodologies in order to create a set of methodology descriptions for precisely defined problem domains in the area of real time programming, distributed processing, etc.

To summarize, we use attributed metaforms as a unifying model for a pds, to facilitate the interactive analysis of program development and the integration of various existing tools. We view such an integrated pds facility, based on existing analysis techniques and tools, to be more effective than its individual component parts for developing reliable software. Also, this pds does not preclude the integration of more sophisticated tools as they are developed. The integrated pds facility is being constructed now for production use and is expected to support all phases of the software life cycle in the near future.

Short Term Research Goals

The distinguishing characteristics of the various problem domains are poorly formulated even though our collective knowledge of programming does

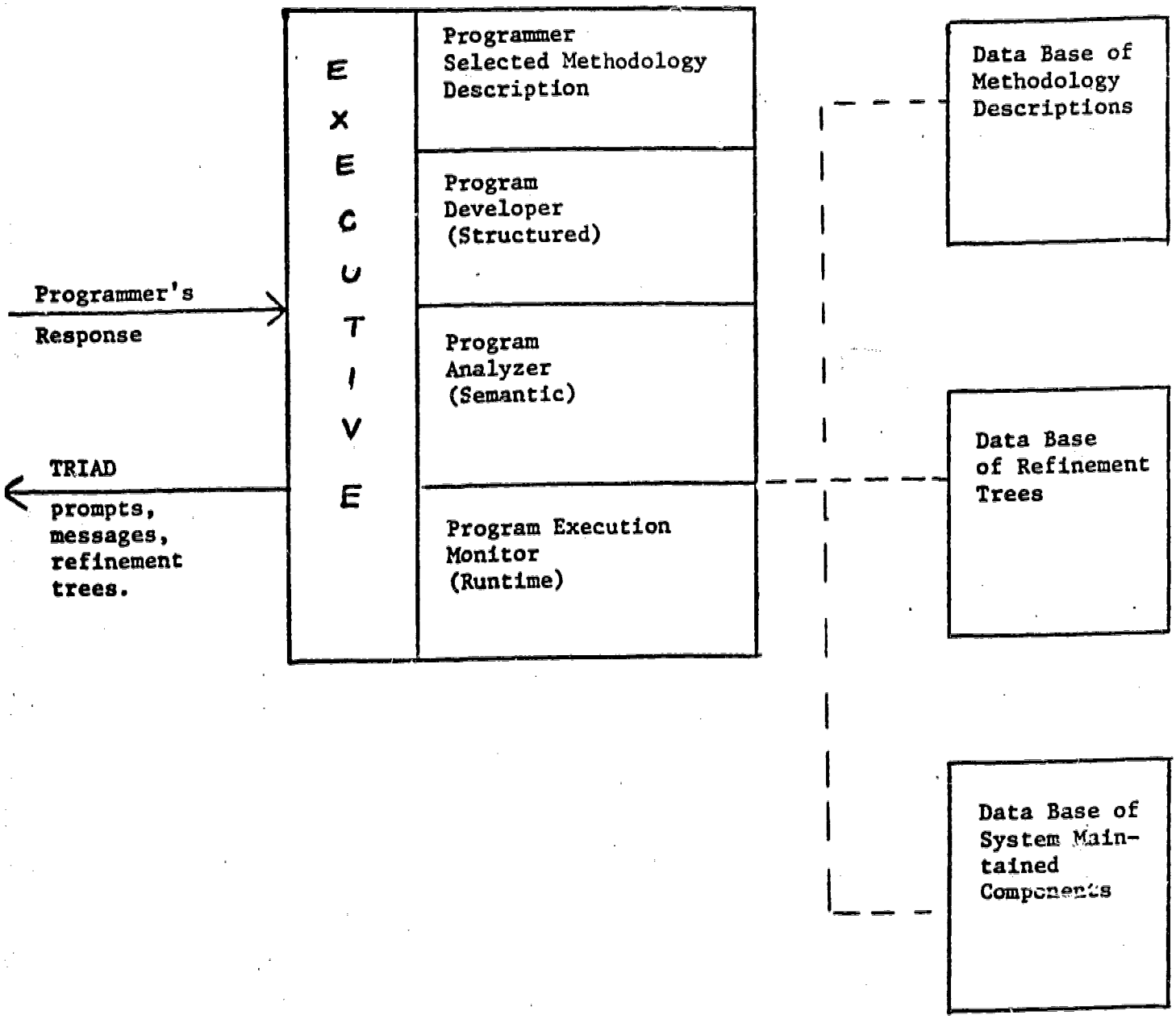


FIGURE 1 TRIAD Overview

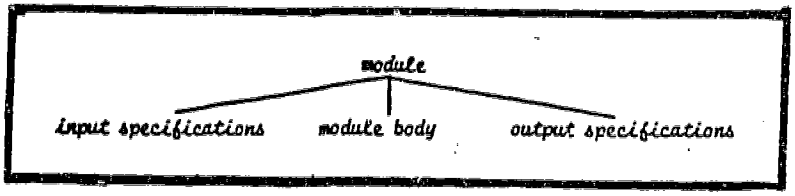


Fig. 2a. General Description of a Methodology Step

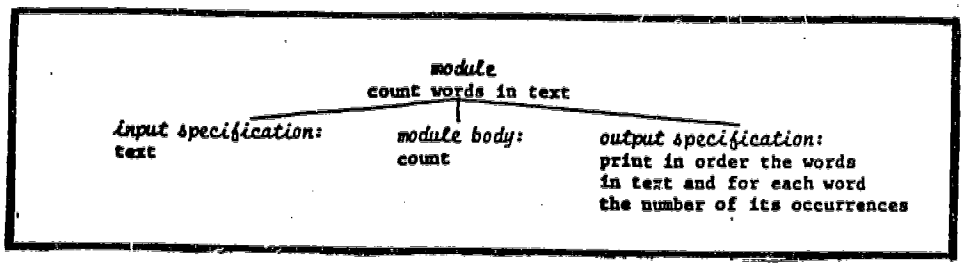


Fig. 2b. Specific Methodology Step. The italicized pds prompts are followed by the programmer's response. These prompts are based on the General Description.

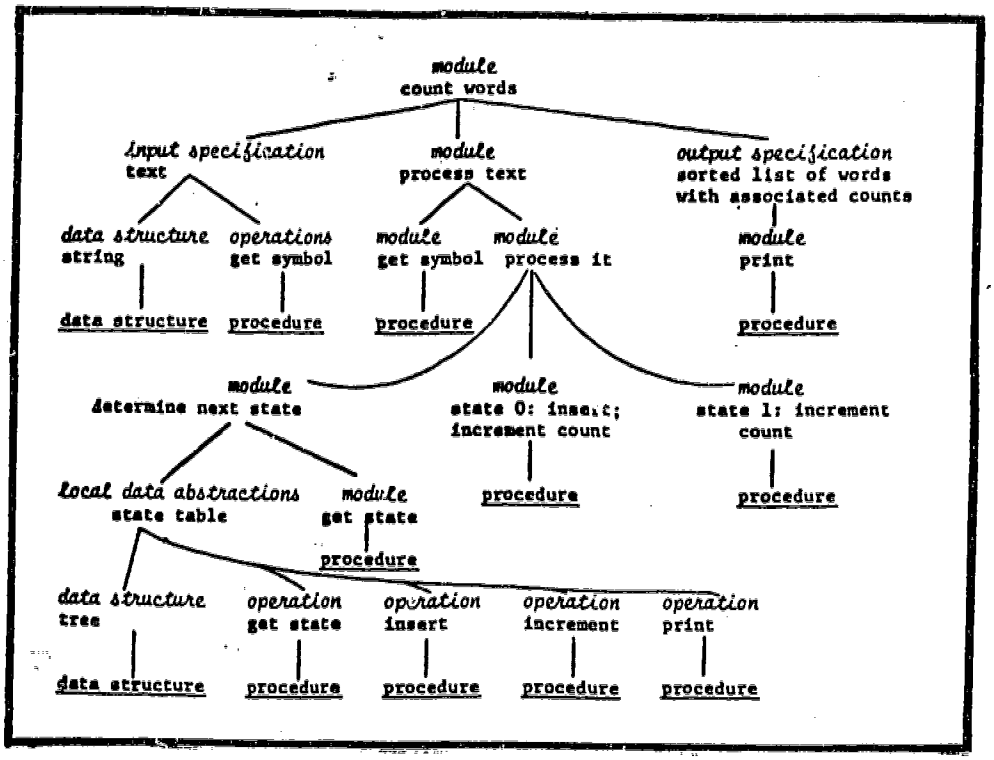


Fig. 2c. Refinement Tree for a Text Processing Problem. Procedure and data structure denote actual code in a high level language like PASCAL.

imply the existence of several programming categories such as concurrent, real-time, distributed, sequential, database, and on-line. A further refinement of these categories could be based on whether the problem requires simulation, numerical techniques, text processing, or is data structure oriented. By studying program development patterns in each problem domain, well-tailored methodologies could be created and incorporated into a pds methodology library.

To the extent possible, a pds analyzer should analyze the development process and suggest the use of existing library modules. Such a semi-automatic translation facility can be readily incorporated into a pds if an operational notation is used to record program development. The pds analyzer should also synthesize a variety of other helpful messages. Global data flow analysis techniques are currently used in compilers to provide such messages for programs [11]. These methods must be further developed to perform global data flow analysis on the development of the program as well as the final program. The design language used for program development is important. If a functional specification language is used, program verification could be attempted. If an operational specification is provided, automatic translation can be attempted.

A pds with an integrated set of tools, including a sophisticated analyzer, as described above, should be constructed for actual production use rather than as an experimental curiosity. The design of such a pds should, of course, allow for the integration of more advanced tools as they are developed. Hence, the design of the pds itself is important.

Long Term Research Goals

The understanding of specification languages must improve to the point where the verification of large production systems is actually possible. In addition, languages must be designed to enable the execution of abstract programs for testing systems before implementation details are developed. In general, the execution of such abstract programs will be extremely inefficient. Hence, the need to supply implementation details for production programs. The automatic translation from a program specification written in a functional language to one in an operational language is also an important area for research. Work in this area has been done by Balzer [1].

As mentioned before, important research is being conducted in developing automatic systems for writing programs. Such Artificial Intelligence systems will not be immediately available for producing general production programs. However, effective experimental systems have been developed for very restricted problem domains.

References

[1] R. Balzer, N. Goldman, "Principles of Good Software Specification and their Implications for Specification Languages," Proceedings of IEEE Conference on Specification of Reliable Software, Cambridge, MA, April 1979.

[2] M. Blattner, J. Ramanathan, "Attributed Metaforms for High Level Design and Analysis of Algorithms," Proceedings of the Conference on Information Sciences and Systems, April 1979.

[3] T. E. Cheatham, J. A. Townley, and G. H. Holloway, "A System for Program Refinement," Proceedings of the 4th International Conference on Software Engineering, Munich, Germany, 1979, pp. 53-62.

[4] Stoneman Environment Requirements, Department of Defense, February 1980.

[5] R. W. Floyd, "Toward Interactive Design of Correct Programs," IFIP Conference, 1971, Amsterdam, The Netherlands, 1972, pp. 7-10.

[6] S. Ginsburg, "A Survey of Grammar Forms - 1977," Sixth Int'l. Symp. on Math Foundations of Computer Science, TatranskaLomnica, Czechoslovakia, 5-9, 1977.

[7] M. A. Jackson, "Principles of Program Design," Academic Press, New York, 1975.

[8] K. Jensen and N. Wirth, PASCAL User Manual and Report, Springer-Verlag, New York, 1975.

[9] B. Liskov, A. Snyder, R. Atkinson, and C. Schaffert, "Abstraction Mechanisms in CLU," CACM 20, 8 (August 1977), pp. 564-576.

[10] G. J. Popek, J. T. Horning, B. W. Lampson, J. G. Mitchell, and R. L. London, "Notes on the Design of EUCLID," Proceedings of an ACM Conference on Language Design for Reliable Software, Ed., D.B. Wortman, March 1977.

[11] J. Ramanathan and M. Blattner, "Program Forms and Program Form Analysis," AFIPS Conference Proceedings for the National Computer Conference, New York City, 1979.

[12] D. T. Ross and K. E. Schoman, Jr., "Structured Analysis for Requirements Design," IEEE Transactions on Software Engineering, Vol. SE3, No. 1, January 1977.

[13] M. Shaw, W. A. Wulf, and R. L. London, "Abstraction and Verification in Alphard: Defining and Specifying Interaction and Generators," Communications of the ACM, August 1977, Vol. 20, No. 8.

[14] J. F. Stay, "HIPO and Interactive Program Design," IBM Systems Journal, 1976.

[15] D. Teichroew and E. A. Hershey, III, "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems," IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, January 1977.

[16] J. D. Warnier, "Logical Construction of Programs," Van Nostrand, New York, 1974.

[17] T. Teitelbaum and T. Reps, "The Cornell Program Synthesizer: A Syntax Directed Programming Environment," Department of Computer Science, Cornell University, Ithaca, New York, 1980.

3. DATABASE COMPUTER RESEARCH

Faculty: David K. Hsiao
Douglas S. Kerr
Richard Underwood

Students: Jaishankar Menon
Tamer M. Ozsu

The research on database computer architecture has both short-term and long-term goals. For the short-term, we are concentrating on a research program in multi-mini database management systems. For the long-term, we are striving for an ideal database machine architecture. Let us describe briefly these two programs.

Research Program in Multi-Mini Database Management Systems

It is generally knwon that the use of a single general-purpose digital computer with dedicated software for database management to offload the mainframe host computer from database management tasks yields no appreciable gains in performance and functionality. Research is therefore being pursued to replace this software backend approach to database management with an approach which will yield good performance and new functionality.

The proposed research utilizes a system of six PDP 11-44s and one VAX 11/780, known as the test vehicle. Each of the PDP 11-44 computer systems consists of a primary memory box of 256 kbytes, at least one disk of 67 Mbytes and a memory-to-memory bus connected to the VAX 11/780 which has 1.5 Mbytes of primary memory and assorted peripheral devices. The configuration is intended to achieve concurrent operations among the six PDP 11-44s and their respective disks. The VAX 11/780 schedules the concurrent operations on the PDP 11-44s, communicates with other computer systems (known as frontends) and serves as the control of the entire configuration (i.e., the database computer backend). For our study, the VAX 11/780 also interfaces with the systems programmers. (See Figure 1).

The aim of the proposed research is to investigate whether, for the management of large databases, the use of multiple mini-computer systems in a parallel configuration is feasible and desirable. By feasible we mean that it is possible to configure a number of (slave) minicomputers each of which is driven by identical database management software and controlled by a (master) minicomputer for concurrent operations on the database spread over the disk storage local to the slave computers. This approach to large databases may be desirable because only off-the-shelf equipment of the same kind is utilized to achieve high performance without requiring specially-built hardware and because identical database management software is replicated on the slave computers. The approach makes the expansion of the capacity and concurrency of the database management system easy.

To study the feasibility, we intend to investigate the software architecture issues and the hardware limitations of the master and slaves. We also intend to investigate the replicable software for the slaves. Since these

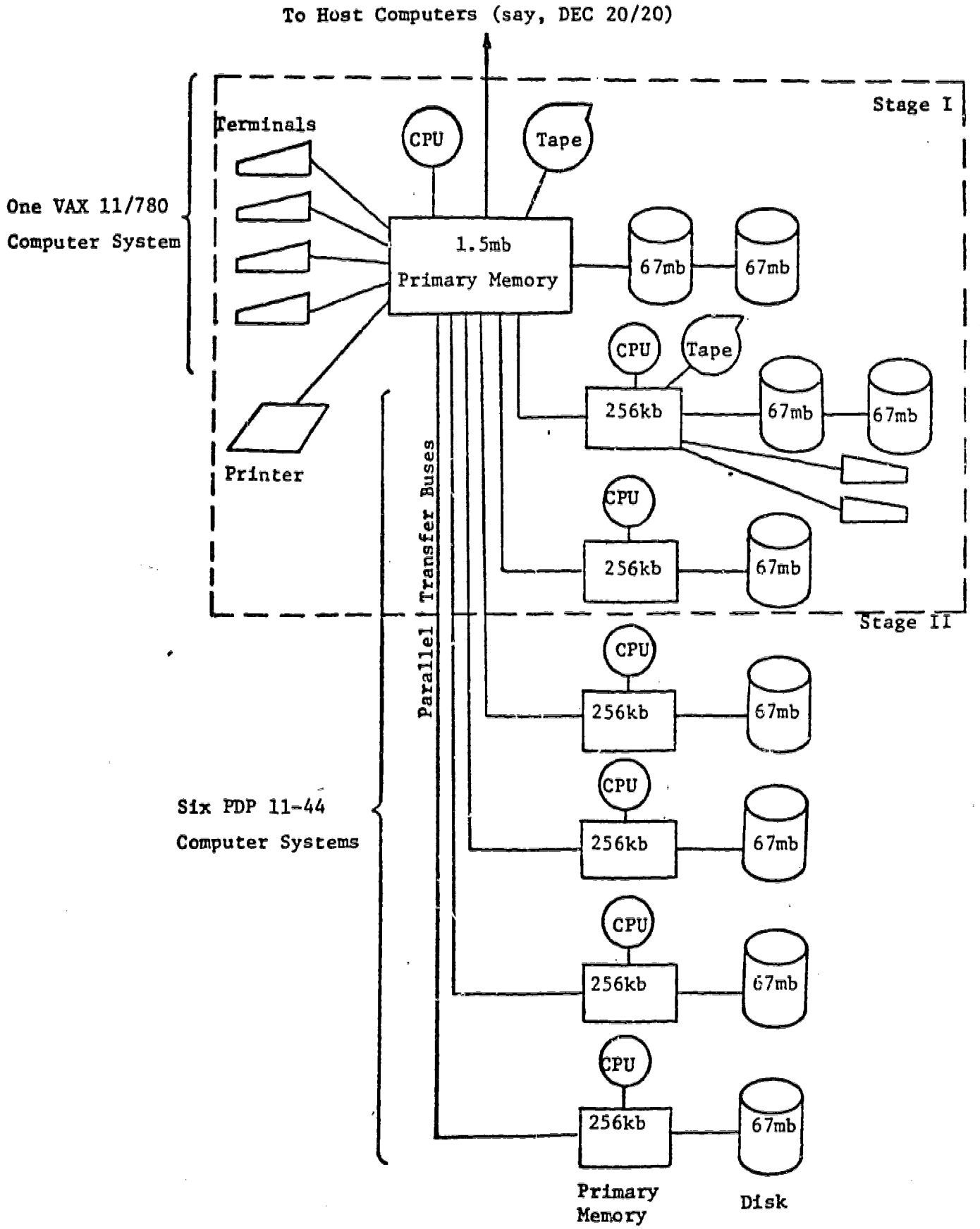


Figure 1: A Test Vehicle for Database Computer System Research

slaves are to be operated concurrently with corresponding single-channel disks, we can investigate the effects of either single-query-and-multiple-database-streams or multiple-queries-and-multiple-database-streams operations for performance improvement. To study the desirability, we intend to consider factors relating to the problem of capacity growth and cost effectiveness. The central issue may be whether we can realize a high-performance and great-capacity database management backend with the cheapest possible minis, large number of single-channel disks and replicable software cost-effectively.

The above program is termed short-term, because it assumes only available hardware and present technology. Furthermore, both feasibility and desirability issues can be resolved in the time frame of two to three years.

Research Program on Database Machine Architecture

In the long-run, the solution to large capacity and high performance database management may lie in special-purpose machines, known as the database computers. With the advent of LSI and VLSI, block access memories (such as magnetic bubbles and charge-coupled devices), together with modified and improved on-line disk technology, it is perhaps time to replace complex and inefficient database management system software (DBMS) with innovative and cost-performance effective hardware solutions to very large database management. This search for an ultimate database machine architecture to provide large capacity, high performance and low cost database management is the goal of this research undertaking.

Initially, the test vehicle (see Figure 1) is used to emulate the architecture of the database computer DBC. As specialized hardware, DBC is designed to handle very large databases (say, beyond 10 billion characters), to perform database management operations effectively, and to yield high throughput unattainable by general-purpose computers with conventional database management software. Due to the complexity of database applications and the diversity of database management, analytical evaluation of database computer performance has been mostly based on broad assumptions and simplified settings. Although these evaluation results have been published (see References), they are too gross to be useful for the identification of performance bottlenecks caused by the components of the database computer. Without a closer examination and refined evaluation of potential performance bottlenecks of various hardware components, it is not possible to locate the strong and weak points of the database computer architecture. Consequently, improvements to the database computer design cannot be readily carried out.

By emulating the database computer on the test vehicle, experimental and realistic performance evaluation of DBC in supporting various database applications can be conducted. The evaluations can focus on database management in terms of modes of operations (e.g., retrieval-intensive vs. update-intensive), models of databases (say, relational vs. CODASYL) and time and space constraints of the man/database interaction (e.g., real-time requirements with redundant data entries and integrity requirements with serial updates).

The information gained from the performance evaluation can then be extrapolated to reflect the performance of DBC. Furthermore, the information can be used to verify the analytical results found earlier.

The objectives of the research are four: (1) To identify the performance evaluation techniques and methodology that are unique to database computer architecture. (2) To validate the analytical study of the DBC design against the experimental results conducted on the test vehicle. (3) To relate the findings on the emulation to the design details of DBC in particular and of database computers in general. And (4) To recommend modifications and improvements of the DBC architecture in order to support very large databases, attain high throughput and perform effective database management.

More specifically, the test vehicle is being configured to reflect the design of the three major components of DBC. The three components are (1) the database command and control processor DBCCP (i.e., the central processing and control unit of DBC) which executes the DBC commands, clusters the records for input and updates, schedules various subtasks and activities of DBC, and communicates with the host computer, (2) the on-line mass memory MM (i.e., the repository of the database) which provides high-volume and high-performance database management with tracks-in-parallel readout and write-in and content-addressability features, and (3) the structure memory SM (i.e., the index storage and processing unit) which enables access control information to be stored and processed readily so that accesses to the database can be restricted to relevant and authorized data, thereby narrowing the content-addressable space of the mass memory MM. Physically, the PDP 11-44s will be used to emulate the MM and SM and the VAX 11/780 will be used to emulate the DBCCP.

Ultimately, this program will lead to the study of other database machine architectures and their relative cost and performance gain and loss compared to the performance and cost of the DBC design.

Government, Industry and University Support for the Programs

The test vehicle of 6 PDP 11-44s and one VAX 11/780 and assorted disks, terminals, tapes and printer will be funded in two stages. In the first stage, two PDP 11-44s with three disk drives, one tape station and two terminals along with ont VAX 11/780, two disk drives, one tape station, four terminals and one printer will be funded for the 1980-81 period. In addition, parallel transfer buses connecting the PDP 11-44s and VAX 11/780 will also be included. The equipment and its maintenance are funded jointly by the Digital Equipment Corporation, the Office of Naval Research and The Ohio State University. It is hoped that orderly and fruitful progress in the first stage will justify an additional funding of the second stage of equipment, namely, the remaining four PDP 11-44s and their assorted devices.

References

We have been active in database computer research since 1976. A list of relevant publications is included herewith. All the published work is in the areas of theory, design and analysis. There has been no experimental and empirical work done on the database computer.

On the Design of a Database Computer, known as DBC:

[1] J. Banerjee, R. Baum, and D. K. Hsiao, "Concepts and Capabilities of a Database Computer," ACM Transactions on Database Systems (TODS), 3, 4, December 1978, pp. 347-384.

[2] J. Banerjee and D. K. Hsiao, "DBC - A Database Computer for Very Large Databases," IEEE Transactions on Computers, C-28, 6, 1979, pp. 414-429.

[3] K. Kanna, D. K. Hsiao, and D. S. Kerr, "A Microprogramming Keyword Transformation Unit for a Database Computer," Proceedings of the 10th Annual Workshop on Microprogramming, New York, October 1977, pp. 71-79.

[4] D. K. Hsiao, K. Kannan, and D. Kerr, "Structure Memory Designs for a Data Base Computer," Proceedings of ACM Conference '77, Seattle, October 1977, pp. 343-350.

[5] K. Kannan, "The Design of a Mass Memory for a Database Computer," Proceedings of the Fifth Annual Symposium on Computer Architecture, April 1978, pp. 44-51.

[6] J. Banerjee and D. K. Hsiao, "Parallel Bitonic Record Sort - An Effective Algorithm of the Realization of a Post Processor," OSU-CIS Research Center Technical Report TR-79-1, March 1979.

[7] J. Banerjee, D. K. Hsiao, and J. Menon, "The Cluster and Security Mechanisms of a Database Computer (DBC)," OSU-CIS Research Center Technical Report TR-79-2, April 1979.

[8] D. K. Hsiao and J. Menon, "The Post Processing Functions of a Database Computer," OSU-CIS Research Center Technical Report TR-79-6, July 1979.

[9] D. K. Hsiao and J. Menon, "Design and Analysis of Update Mechanisms of a Database Computer (DBC)," OSU-CIS Research Center Technical Report TR-80-3, June 1980.

[10] D. K. Hsiao and J. Menon, "Parallel Record-Sorting Methods for Hardware Realization," OSU-CIS Research Center Technical Report TR-80-7, July 1980.

On DBC's Capability in Supporting Existing Database Applications with Improved Throughput:

[11] J. Banerjee, D. K. Hsiao, and F. Ng, "Database Transformation, Query Translation and Performance Analysis of a New Database Computer in Supporting Hierarchical Database Management," IEEE Transactions on Software Engineering, SE-6,1 January 1980, pp. 91-109.

[12] J. Banerjee and D. K. Hsiao, "A Methodology for Supporting Existing CODASYL Databases with New Database Machines", Proceedings of ACM Conference '78 , Washington, D.C., December 1978.

[13] J. Banerjee and D. K. Hsiao, "The Use of a Database Machine for Supporting Relational Databases," Proceedings of the 5th Annual Workshop on Computer Architecture for Non-numeric Processing, Syracuse, NY, August 1978.

[14] J. Banerjee and D. K. Hsiao, "Performance Study of a Database Machine in Supporting Relational Databases," Proceedings of the 4th International Conference on Very Large Data Bases, Berlin, Germany, September 1978.

On General Treatment on Database Computers:

[15] R. I. Baum and D. K. Hsiao, "Database Computers - A Step Towards Data Utilities," IEEE Transactions on Computers, C-25, 12, December 1976, pp. 1254-1259.

[16] D. K. Hsiao, "Database Computers," Advances in Computers, Academic Press, V. 19, June 1980, pp. 1-64.

[17] D. K. Hsiao, "The Role of Database Computer Prototypes," to appear in the Proceedings of the 13th International Hawaii Conference on System Science, Honolulu, Hawaii, January 1980.

[18] D. K. Hsiao and S. E. Madnick, "Data Base Machine Architecture in the Context of Information Technology Evolution," Proceedings of the 3rd International Conference on Very Large Data Bases, Japan, October 1977, pp. 63-84.

[19] D. K. Hsiao, "Future Database Machines," Future Systems, Infotech State of the Art Report, November 1977, (U.S. distributors: Auerbach Publisher, Ltd.), pp. 307-330.

[20] J. Banerjee and D. K. Hsiao, "Data Network - A Computer Network of General-purpose Front-end Computers and Special-purpose Back-end Data Base Machines," Proceedings of the International Symposium on Computer Network Protocols, Leige, Belgium, February 1978.

[21] D. K. Hsiao, "Database Machines are Coming; Database Machines are Coming! - A Guest Editor's Introduction," Computer Magazine, 11, 3, 1979, pp. 7-9.

[22] D. Kerr, "Database Machines with Large Content-Addressable Blocks and Structural Information Processors," Computer Magazine, 11, 3, 1979, pp. 64-79.

4. RESEARCH IN PROGRAM AND ARCHITECTURE DESIGN FOR REAL-TIME APPLICATIONS

Faculty: Bruce W. Weide

Students: Jose Alegria
Win Bo
Mark E. Brown
Russell Gonsalves
Luis Hernandez
John Lohse

A new research project on real-time applications has just begun in 1980. We have explored many aspects of real-time programming for data acquisition and process control, ranging from the design aids useful to an applications programmer at one end, to hardware and VLSI (very large scale integration) implementation of special-purpose hardware at the other. The project now encompasses research into graphical design tools and languages, interpreters and run-time support for real-time applications, simulation and analysis of real-time control systems, and computer architecture and VLSI design.

Real-time applications often demand much specialized knowledge (for instance, of interrupt handling, synchronization, programmable clocks, and non-standard I/O devices) of programmers or engineers whose background includes relatively little computer science. Unfortunately, nearly every methodology proposed for real-time program design and development fails to recognize this fact of life, and concentrates on ease of access to low level hardware features via high order language constructs [4,5,6,7,8]. We view this approach as intolerable because it continues to ignore the sparse computer science background of the vast majority of those who would like to write software for data acquisition and process control.

As an alternative, we are developing a programming design system suitable for non-experts in concurrent and real-time programming that removes from the program designer the responsibility for dealing with implementation of synchronization and communication that are inherently necessary in real-time applications. Our approach is based on a data-flow, rather than the more conventional control-flow, model [2,3], and is, therefore, a prime candidate for support by graphical design tools.

We are trying to emphasize in this research the overall integration of this general underlying model, the graphical program development system, and a novel implementation of the resulting graphical description of a program directly in hardware that is very well suited to eventual construction using VLSI. Although the detailed technical material that follows is concerned mainly with this architecture, it should be noted that the architecture has been developed in conjunction with and concurrently with the program design system (methodology and computer aids to program development). This synergism has so far been extremely valuable, since it has promoted discussion of actual real-time problems with persons who are responsible for programming data acquisition and process control applications. Such interactions have had a profound effect on our thinking, and have, in fact, led us to the conclusions mentioned above that current approaches to real-time problems are inadequate. Furthermore, they have directed our efforts toward a unique data-driven model that corresponds closely in notation to that used in con-

trol theory, thereby reducing the distance between the program specifications and what can be compiled to run on our special hardware. This greatly simplifies the program design process.

The Data-Flow Model

In a data-flow model, computations are initiated by the availability of data at the inputs of a computation module (called a box) or at the inputs of a special real-time-oriented module (called a link). Boxes and links are joined together in a real-time data-flow graph by directed edges along which tokens representing data may flow. A box performs some computation on its input data tokens when all are present at the inputs of the box and when none are present at the outputs. At the completion of execution of the box, the input tokens are simultaneously removed and the output tokens appear on the output links. This computation is supplied by the programmer in some form, and is guaranteed to operate totally independently of all other boxes and links in the graph. As a result, only sequential computations, with which most real-time application programmers are already comfortable, need be described using ordinary programming language (string-oriented) syntax.

Each link is able to remember state information (including that related to actual clock time) and potentially change state and produce an output token whenever a data token appears at either of its two inputs. Link operations are performed automatically by the system. We have found that it apparently suffices to have fewer than half a dozen different link types to easily accommodate all the applications programs we have written using such graphs. This means that the programmer can quickly learn the syntax and semantics of our data-flow graphs and concentrate on the truly creative (but well directed) process of connecting the components together to design his system. It is the goal of the graphical program development system to facilitate and guide this step.

The direct compilation of such a graph into executable code for an ordinary von Neumann machine is a formidable but not impossible task. Essentially, the graph can be represented internally by a fairly complex data structure that is accompanied by an interpreter that can determine when boxes and links are ready to 'fire" and schedules them in some manner on the single processor machine. This approach has one advantage: portability. In theory, only this interpreter and the runtime support, and perhaps some device-specific code in the boxes, would need to be changed to move to another system. With the situation in the microprocessor industry being that the processor ordered today is obsolete by the time it is delivered, portability is a major concern in practice. For this reason, the interpreter and run-time system for a PDP-11 are now being implemented to show the feasibility of this approach.

On the other hand, this approach fails where most other such attempts have failed: it is too slow for the required real-time responses demanded by many applications. As a result of this shortcoming, we have completed a preliminary design for a highly parallel architecture that directly implements our data-flow graphs.

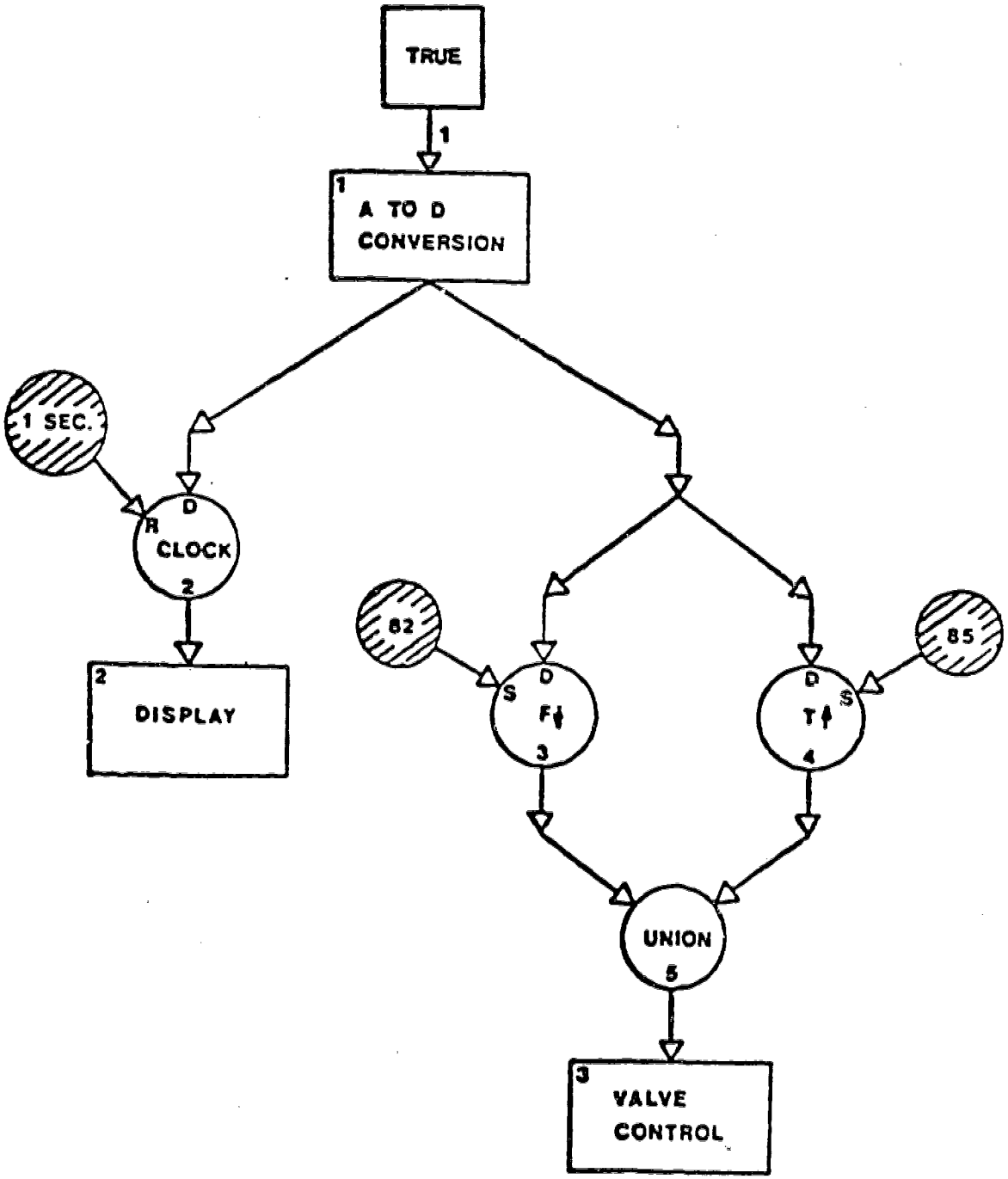
Real-Time Oriented Data-Flow Primitives: An Example

As a simple example of the kinds of problems encountered in the world of process control, suppose we have some chemical process that takes place in a chamber and that produces heat. A coolant is available to keep the process from overheating, and the task is to control the valve that allows the coolant to flow through the chamber. At the same time, an operator's display that indicates the chamber temperature is to be updated every second. A temperature sensor (in the form of an A/D converter) provides this reading, and the control algorithm is simply to open the coolant valve whenever the temperature exceeds 85 degrees, and to close it again whenever it falls back below 82.

Clearly, ordinary Fortran or Pascal will not allow a solution to this problem, due in part to their inability to cope with the requirement of updating their operator's display once a second. The code necessary to access the A/D converter and the operator's display could be written in some high order languages, or directly in assembly language, but dealing with the true "real-time" constraint would involve a substantial programming effort including perhaps a scheduler and some routines to manipulate a real-time clock. Even in Ada, these tasks are not subsumed by the system. Furthermore, it is not at all clear how to coordinate the activities that are occuring simultaneously, nor how to control access to common data, in this case the temperature. The methodologies for designing real-time programs in ordinary languages are not only weak, in that they offer little guidance to the programmer who is trying to wind his way through the maze of possible designs, but rely on the programmer's knowledge of fancy communication and synchronization mechanisms offered by the target language. As mentioned above, most programmers who need to develop solutions to real-time problems are not highly trained nor particularly skillful in this area.

Figure 1 shows how the solution to the temperature control problem would be expressed using our data-flow notation. The approach to creating this graph using our methodology is to first identify the devices that must be accessed (the A/D converter, the display, and valve control) and to create a box to operate each. In this case, the A/D converter box takes as input only some indication that sampling should take place, which we have chosen in the example to be a data token having the value "true". This box produces as output a token having the value of the temperature sampled. At this stage, A/D conversion is truly a "black box" with one input and one output. Similarly, the display box has a single input, the value to be displayed, and produces no output. Of course, it does have some effect, namely updating the operator's display.

The third box is somewhat more complicated, but still simple in absolute terms. We decide that if it receives a "true" input token, it should do whatever is necessary to open the coolant valve, or if it receives a "false" token to close the valve. Presumably this is by means of a switch or a D/A converter, but at this level of the design that is irrelevant. We assume only that the valve remains either opened or closed until it is explicitly changed by the valve control box.



TEMPERATURE CONTROL

FIGURE 1 - A Real-Time Data-Flow Graph Example

We are now ready to connect these boxes and some links together to make the entire system. As a general rule, sampling should be done at as fast a rate as possible, in the absence of some specification of sampling period, so we choose to input to the A/D conversion box the constant "true"; that is, to immediately place a true-valued token at its input arc whenever the previous one is consumed, causing A/D conversion to occur. This will make sure that so long as the previous A/D conversion is completed and that the rest of the system is ready for another temperature sample, one will be produced as soon as possible.

In a similar fashion, we note that the input to the display box is to be produced only once per second, and is to be the most recent temperature reading from the A/D converter. A new link called the clock link is used here. It has a rate input R (one second) and a data input D (the temperature), and works as follows. Whenever a new data token appears at the data input, its value is remembered as the most recent one and that data token is removed from the input arc. At intervals determined by the rate input (also remembered until changed) the most recent data value received is placed in a token on the output arc, assuming that arc is empty. (If it is not, this "tick" of the clock is skipped.) Placement of a clock link between the A/D conversion box and the display box, and with an initial one second token on the rate input, will cause the display box to be fired every second with the most recent temperature reading.

Finally, we must use the temperature sensed by the A/D converter to produce a "true" token whenever the temperature goes above 85, and a "false" token whenever it falls below 82. The proper tool for this is the threshold link, which has two inputs and produces one output. This link remembers its most recent data values at the two inputs (setpoint S and data D), and emits a token on its output arc whenever the magnitude relationship between S and D changes. Threshold links come in four varieties, since the change in relationship may be either from $S \leq D$ to $S > D$ or from $S \geq D$ to $S < D$, and the output token may be either "true" or "false". In Figure 1, there is a threshold link labeled "F↓" with setpoint 82 to indicate that a "false" token should be emitted whenever the temperature falls below 82, and one labeled "T↑" to produce a "true" token when it moves above 85. Notice that these links do not simply do comparisons, but produce outputs only when the relationship between S and D changes in the direction indicated.

It is now necessary to join the two output arcs of these links to produce a single input to the valve control box, which is done by a union link. This link acts in a FIFO manner to route one of its two inputs to the output. If only one input token is present (as should be the case here) that token is simply placed on the output, assuming the output arc is unoccupied. If an input token arrives while another is still on the other input, it must wait until the first is passed on, but is guaranteed to be forwarded next.

While these threshold and union links would not have been necessary in this example, their purpose being easily subsumed by the valve control box which could just as easily have received the temperature as input, they illustrate an important design consideration. If the setpoints had not been constants, but were provided by the operator through, say, a pair of dials, then the entire valve control portion of the graph would have to be redesign-

ed. With this version, modifiability is enhanced. Recognition and formalization of such design rules and criteria for modularization are an important aspect of the research into the supporting program design system.

It should be clear that, except for timing problems that might arise due to the slow response of the boxes, this system should properly implement the temperature control algorithm. Those timing problems would have to be overcome in any solution, and are not unique to ours. An important point here is that with the design tools we envision in our program design and development system, including color graphics interface during design and extensive analysis and simulation before field testing of the solution, such problems should be identifiable before they cause serious difficulties. Much work has been devoted to such problems by others, and we do not expect to explore these issues further than is necessary to implement the analysis tools we consider necessary in a production system.

A graph such as that in Figure 1 will be drawn on a graphics terminal by the programmer, and will be directly executable by our special hardware, assuming the programmer has provided executable code for the three boxes. The next section briefly describes hardware to do this.

A Highly Parallel Architecture

In a preliminary architecture to implement real-time data-flow graphs, we see the link operations as being performed directly by special-purpose parallel hardware that is provided on a processor-per-link basis in what we term graph memory. Execution of code describing the boxes will be by a set of more conventional processors, each with local memory, and whose number impacts only the potential speed of execution of the boxes when more than one is executable (has fired). Only descriptors of boxes and information regarding their readiness for execution are stored in graph mamory; the actual code and local data storage are in a standard memory called program and data memory in the overview of Figure 2.

The general configuration of graph memory for the temperature control example is shown in Figure 3. Each box is represented by a pointer to its code in program memory, and a pointer to local data in data memory. The code is presumed to be re-entrant, with all data references given by offsets into a local data area, so that code may be shared by several boxes if necessary. In addition to this information, there is some state information for each box to keep track of when it is ready to fire, and connection information so that data and firing information may be communicated to those boxes and links that it is attached to in the graph.

Each link is stored explicitly in graph memory by remembering its two input data values, state information concerning readiness to fire (different from that of the boxes), and connection information. We envision the operation of links as being controlled by a small ROM program or PLA of a few dozen state transitions, with a state change occurring whenever any input token arrives or any output token leaves. Each link must have, according to our initial design, at most 200 bits of RAM organized into two data registers, a counter, and six connection pointers, plus enough logic to decrement and test for zero and do certain register transfers. Depending on the obvious

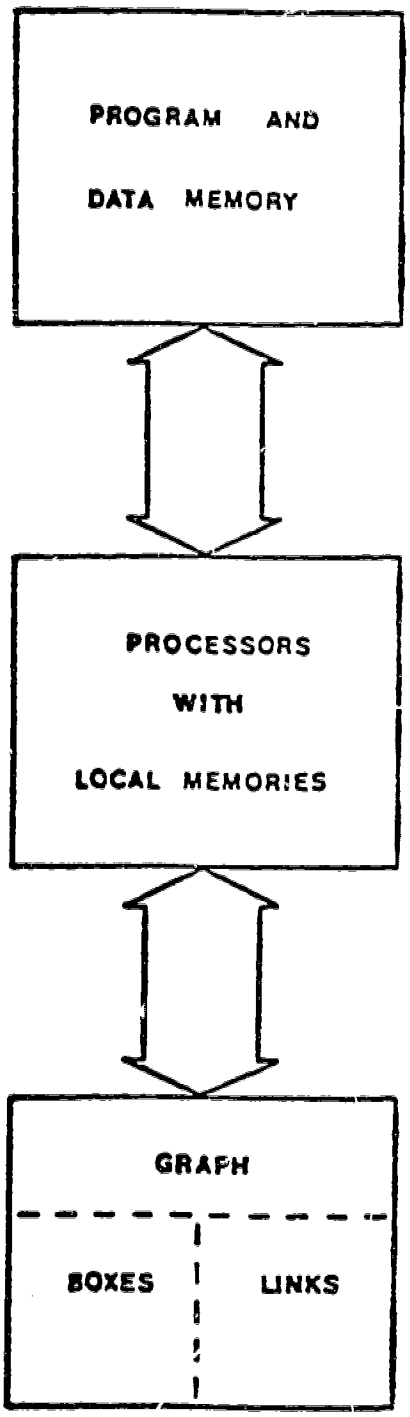


FIGURE 2 - Architecture Overview

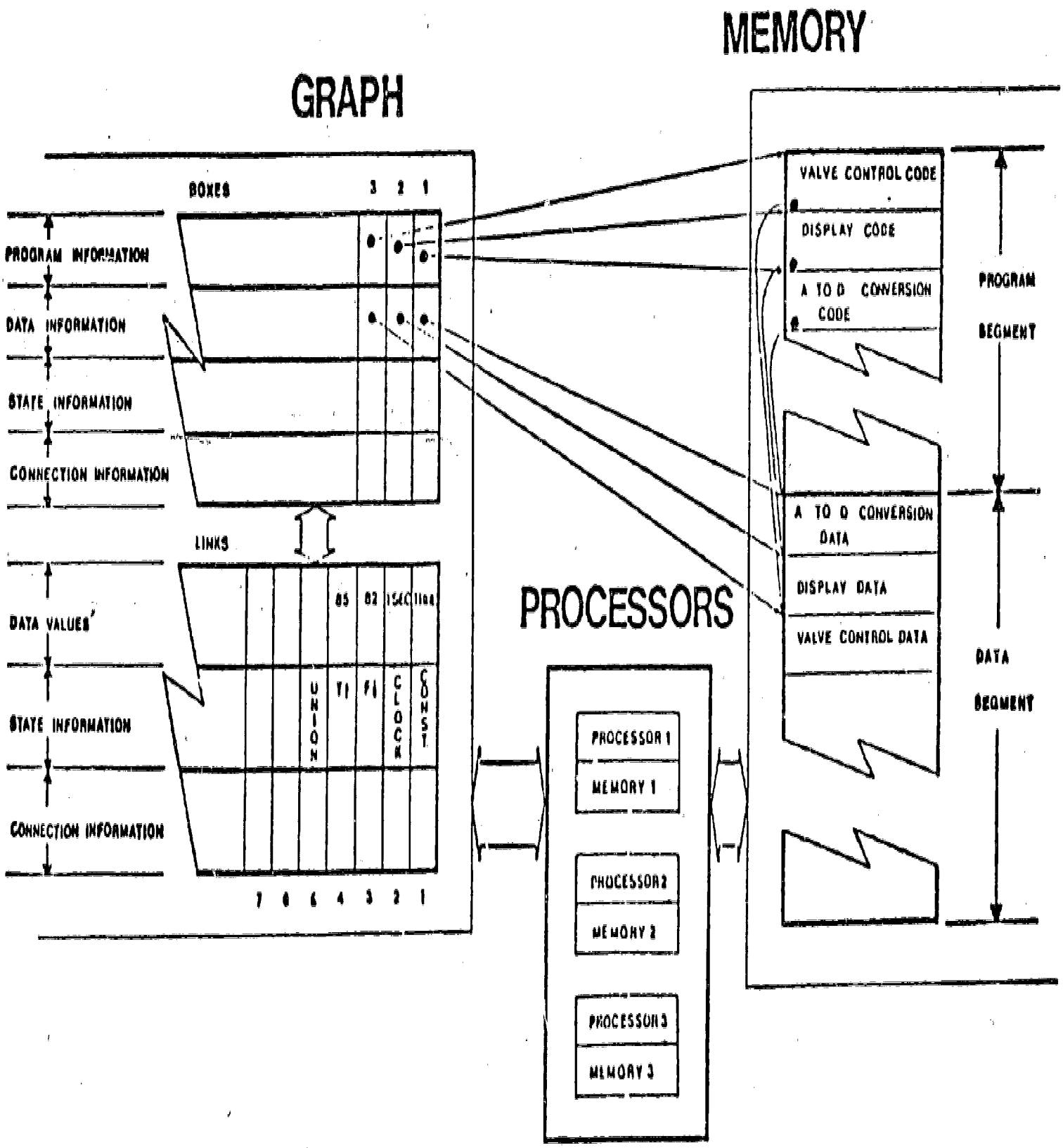


FIGURE 3 - Details of Temperature Control Example

time-space trade-offs in representing the state transitions and on the communication costs involved in presenting data to the links and getting data from them, we have calculated that between 20 and 100 links could be implemented on a single VLSI chip with current technology.

There must be some control of communication between graph memory and the processors that execute boxes, more detailed design of graph memory, especially concerning communication both on and off chip, and more investigation of the precise operation of links and boxes before additional work can be done on implementation. These questions are now under consideration by our group.

Conclusions

Real-time applications are of tremendous importance because of the potential for miniaturization now offered by VLSI technology. One aspect of VLSI development that seems suspect is the trend toward extremely specialized chips that are useful only in very limited applications. We view the research effort proposed here as counter to that trend, in the sense that the full utility of VLSI can be brought to bear on a very large class of applications through this versatile approach to real-time computation. The synergism between the program design aspect and the final hardware implementation has been an important part of our research.

References

[1] M. E. Brown and B. W. Weide, "Preliminary Design of a Highly Parallel Architecture for Real-Time Applications," submitted to 18th Annual Allerton Conf. on Comm., Contr., and Comp., October 1980.

[2] J. B. Dennis, "First Version of a Data-Flow Procedure Language", in Lecture Notes in Computer Science 19, Springer-Verlag, Heidelberg, 1974, pp. 362-376.

[3] J. B. Dennis and D. P. Misunas, "A Preliminary Architecture for a Basic Data-Flow Processor," SIGARCH News 3 (Proc. 2nd Annual Symposium on Computer Architecture), 4 (December 1974), pp. 126-132.

[4] J. B. DeWolf, "Requirements Specification and Preliminary Design for Real-Time Systems," Proc. COMPSAC 77, IEEE, Chicago, November 1977, pp. 17-23.

[5] K. Jackson, "Modularity in Real-Time Computer Systems," Proc. 1975 IFAC-IFIP Workshop on Real-Time Programming, ISA, Cambridge, MA, August 1975, pp. 139-146.

[6] W. T. Mao and R. T. Yeh, "An Illustration of Systematic Design of Parallel Programs for Real-Time Applications," Proc. COMPSAC 79, IEEE, Chicago, November 1979, pp. 392-397.

[7] H. A. Schutz, "On the Design of a Language for Programming Real-Time Concurrent Processes," IEEE Transactions on Software Engineering SE-5, 3, May 1979, pp. 248-255.

[8] N. Wirth, "Modula: A Language for Modular Multiprogramming," Software Practice and Experience 7, 1, Jan.-Feb. 1977, pp. 3-36.

5. RESEARCH IN PARALLEL COMPUTATION, BUS AUTOMATA, AND FINITE STATE AUTOMATA

Faculty: Jerome Rothstein

Students: John R. Mellby
Anne B. Davis
David M. Champion

The present program is an outgrowth of biologically motivated research (Rothstein [1, 2, 3, 4, 5]) which has both inspired and benefitted from related work in computer science (Rothstein [6, 7, 8, 9] , Rothstein and Weiman [10], Rothstein [11], Moshell and Rothstein [12], Rothstein [13, 14, 15], Rothstein and Davis [16]). It has produced four Ph.D. dissertations so far. The results of two have appeared in full-length papers (Weiman and Moshell [10] and [12] respectively). Mellby's dissertation was completed in the past year; a few of the results obtained are used in [15] and [16]. Davis' dissertation is currently being written up in final form; some of the results are included in [16]. It is hoped that a fifth dissertation (Champion) will ultimately develop; the research is still in a preliminary stage. This report briefly reviews earlier work, including only enough detail to make recent and current work comprehensible. The major theme is parallel speed-up of computation or data-processing using a class of devices called bus automata. A minor one is a class of finite state automata whose study was motivated by the desire to utilize modular representations of integers in fast computations by bus automata.

Survey of Earlier Bus Automaton Research

Bus automata (BAs) are arrays of automata, combined with switching arrays, in which a portion of the switching array is under local control by each automaton. The local automata communicate with others only through busses made up of links switched in by automata between those actually communicating. In effect widely separated automata can become nearest neighbors with the help of intermediate ones. The switching array is a dynamically reconfigurable global communication network permitting the array of automata to work in a distributed or parallel fashion on desired tasks. For convenience the combination consisting of a local automaton and that portion of the switching network which it controls directly is called a cell. Each cell is connected by links to a finite set of cells called its set of nearest neighbors. The links are generally unidirectional (a pair of oppositely directed links clearly gives the behavior of a bidirectional link) and classified as input or output links from the viewpoint of a given cell. Any one of a cell's output links is physically identifiable with an input link of some nearest neightor and conversely (except for external connections). Cells can be sources or destinations of messages, in which case an output or input link respectively initiates or terminates a communication bus. Cells can also connect input links directly to output links by means of internal paths called channels. Two cells neighboring a given cell which are not neighbors of each other can effectively become so as a result of switching in a channel by their common neighbor. Cascading links and channels construct busses. A cell can both receive information from a link (it can process the information within a characteristic processing time, during which a state change occurs) and let the information propagate "unnoticed" through a channel (propagation time, deter-

mined by signal velocity and cell size, can be orders of magnitude smaller than processing time). All links and channels can be operated simultaneously in their various fashions, and the roles played by the cell in each case (source, sink, switch, etc.) can be freely chosen. It is precisely the ability of the BA to involve numbers of cells, far greater than those to which inputs are fed initially, almost at once, which gives it such a great advantage over the cellular automaton (CA) for parallel computation.

Details of previous BA research are given in the literature cited. In [11] the origins of the BA concept are given, the notions of time costs for state changes (C_s) and propagation (C_p) are introduced, and several speedup results obtained. In particular, it is shown that anything a finite state automaton can do over a long interval can be done by a BA at a time cost of one state change, and the result is generalized to a speed-up theorem relative to a Turing machine: the number of state changes needed by the BA to to the same job is one plus the number of turnarounds of the finite control of the Turing machine on the tape.

The BA concept is given a specific formalization for the case of a Cartesian array of cells in [12], and applied to language recognition. The knotty problem of parallel computational complexity introduced in [11] is examined further and related to other work on complexity.

In [13] the groupoid formalism applied in [11], but invented earlier to study patterns via their generating algorithms [8], motivated development of a novel number system. Though aimed at parallel computation initially, its bizarre properties have hindered its application to ordinary calculations. However, it has interesting combinatorial and pattern aspects. These are so novel that it would be rash to predict how far further research might go or how useful it might eventually become. In many ways, this system is equivalent to the field of polynomials in one indeterminate over GF(2).

In [14] the BA is shown to afford a natural model for skill acquisition in the same sense that finite state machines serve as behavioral models. This is developed further in [15], in which the BA is taken a long way toward visual and nervous system models. Immediate recognition of patterns is accomplished, where the features can be topological, metric, logical, or a mixture, as is immediate computation of several number theoretical functions whose complexity class is NP (non-deterministic polynomial time or space in current complexity theory). In a real sense combinatorial topology becomes computational in this paper, which also shows how global coordinate transformations, conformal or more general can be carried out. The latter is applied to recognition of parabolas (and conic sections generally) in [16], which contains BA architectures to implement both the conformal method and algorithms in which geometric, rather than analytical properties are exploited. More of this below, in the discussion of Davis' dissertation.

Statistical fitting of the best straight line to data points in a plane has been accomplished immediately on the BA. Extension of the computation methods to curves in the plane or in space now seems entirely feasible for any acceptable curve-fitting or regression procedure. More of this below in the discussion of Mellby's dissertation.

In [17] the groupoid string formalism introduced in [8], and shown to be computationally universal in [11], is applied to cryptographic systems. Here both encoding and decoding algorithms can be rapidly implemented in parallel on a BA, while the computational task faced by the enemy is overwhelming to the point of being completely non-feasible. Much of the power of the method resides in the fact that complex symbolic algebraic operations are computable immediately by a BA. This last point is briefly discussed below, along with its possible relevance to problems of associative memory and relational data bases. This problem is currently under examination by D. Champion. The approach is closely related to [14] and [15], as well as [17].

The final topic is called (b,k)-automata. These are finite state automata which process strings representing numbers in base b to compute their residues mod k. They are very interesting machines many of whose properties are listed.

Recent Work on Bus Automata and (b,k)-Automata

Immediate Computation

On bus automata computation in stroke notation is immediate for many arithmetic functions, both elementary and of complexity NP [15], for little is required beyond moving strokes to appropriate locations. The data processing involved in both languages and pattern recognition, so far, is also largely of this simple nature. However, if one is interested in practical implementation of BAs on chips, say, then restriction to operations on numbers in stroke notation is a severe limitation. Fortunately, it has been found feasible to operate in binary, as well as to transform immediately between stroke and binary. In this section we only summarize how immediate multiplication in binary is accomplished; further information is in Mellby's dissertation and in joint papers of Rothstein and Mellby still in preparation.

Figure 1 is a schematic representation of the BA architecture needed to overcome the chief barrier to binary multiplication. Binary multiplication is essentially shifted addition; the part of the operation not obviously immediate is the generation of column sums and handling of carries. The key observation is that the column sum is an integer presented in stroke notation (except for the 0's which need to be "squeezed out" as in [15], Figure 3.4 and accompanying discussion). In Figure 1 the carry-in is bussed to the end of the column to be summed; each arrow-line busses a 1 from from its "tail" to its "head". The column sum then has half its strokes bussed to the carry-out cells indicated, its residue mod 2 being stored as 0 or 1 in the cell separating the cells labelled carry-in and carry-out. The carry-out is bussed to a line of cells in the plane above, serving as the carry-in to the next column sum. The "spiral-staircase" connections are set up initially, and clearly apply to all binary multiplications. The time to complete the multiplication, except for propagation time proportional to the size of the problem, is then the time to set the mod 2 cells. These are set in parallel; the time for one state-change is all that is needed, and this is immediate by definition. Immediacy of binary multiplication now follows because only a fixed finite number of state changes is required for all operations needed to multiply any two binary numbers together.

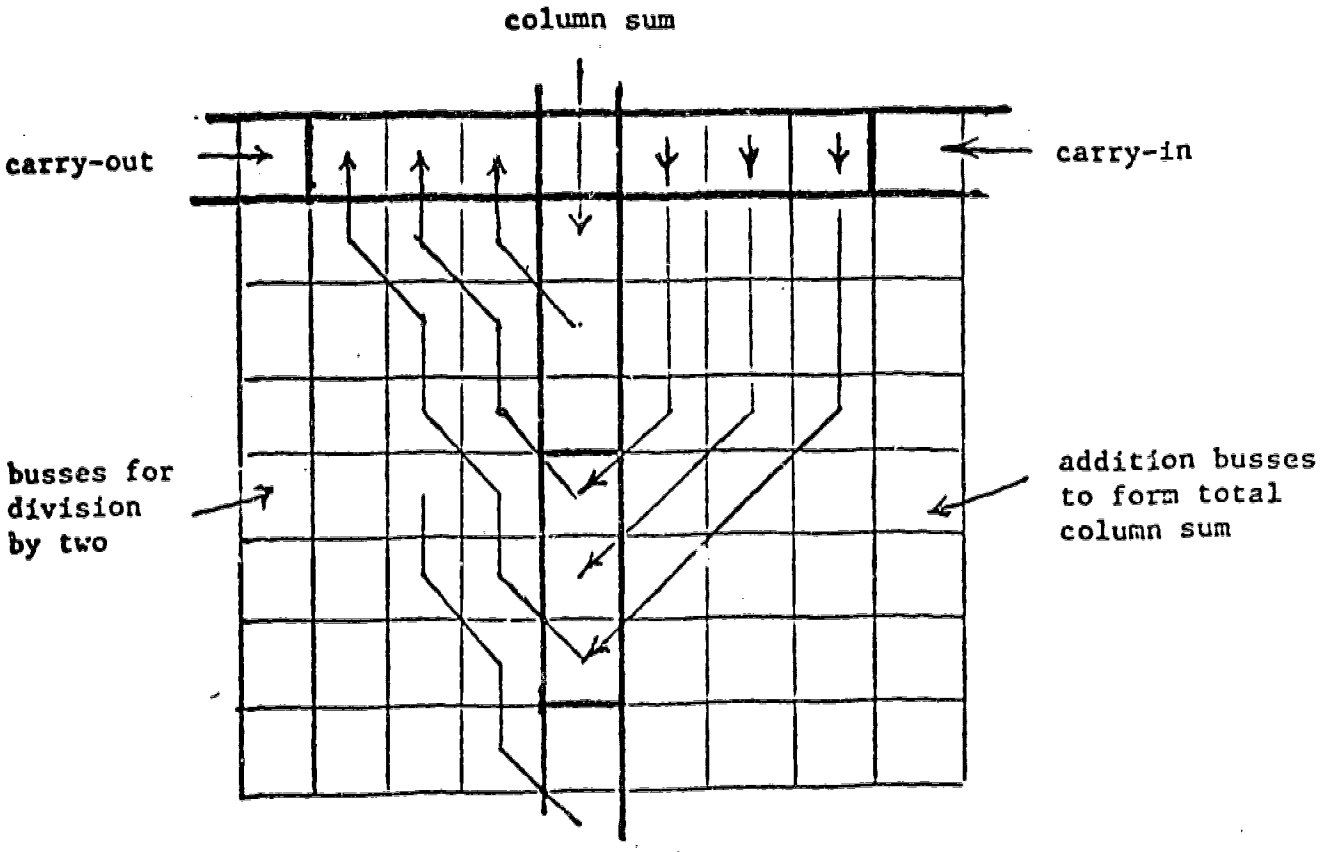


Figure 1 Plane of BA Performing Immediate Multiplication in Binary

Recognition of Conics by Bus Automata

An account of some of the work on this topic has appeared in [16], with more details and additional research to be given in the dissertation of Ms. A. B. Davis. In this summary we omit discussion of the conformal approach and of the BA architecture, explaining only the gist of the "geometric" method of conic recognition. The basic strategy is to find geometric properties which are both definitive, i.e. possessed by no configurations other than those to be recognized, and immediate, i.e. given the data describing a configuration, the processing of it to determine presence or absence of the property is immediate on the BA. One strong stimulus for this approach was the observation that setting up the coordinate system needed to apply the conformal approach to parabolas was not immediate (linear time) at first, but became so when a specific algebraic-geometric property of parabolas was exploited.

It turns out that all conics could be recognized and the various types distinguished from each other by exploiting properties of sets of conjugate diameters (doubtless other properties will also do). For a BA whose cells represent those of the Cartesian (x,y) plane it is easy to generate or recognize straight lines or segments, to cover the plane with any oblique coordinate system, or to bisect segments, all immediately. It then became possible to find a set of parallel chords, and the locus of their midpoints. If the locus is not straight the curve is not a conic. If it is a straight segment it might be a central conic (ellipse, hyperbola), if a ray, it might be a parabola. In either of those cases the locus is a diameter conjugate to the diameter of slope equal to that of the parallel chords. A second set of chords parallel to the locus is then generated; if and only if these chords generate a diameter which is a member of the first set of chords will the candidate be a central conic. For a parabola the second set of chords are lines parallel to the symmetry axis. There are many fussy questions of degenerate cases, grid finiteness, and resolving power, but in essence the situation is as stated, as they are all handled immediately (up to the limit of resolution).

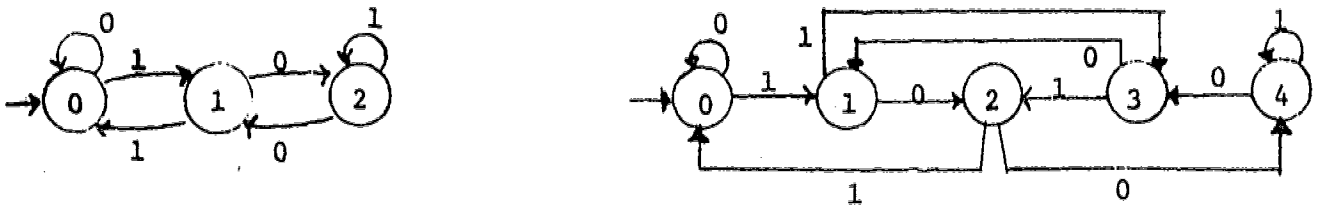
Associative Memory and Relational Data Bases

There are many close analogies between the general problems of associative memory, pattern or language recognition and relational data bases at both the conceptual level and at the level of BA implementation. For example, recognizing the context-sensitive language $\{a^n b^n c^n \mid n = 1, 2, \ldots\}$ is essentially the same as recognizing a simple pattern like a stick with three zones of equal length painted in three colors in a prescribed order. It in turn is related to searching a data base for elements classified a particular way according to three separate attributes. Again, a pattern can be viewed as a word over an alphabet of features, with specified relations between features. This has already led to the notion of immediate relations [15], and because relations can also be exhibited as predicates, to the notion of immediate predicates.

Exploratory work has begun on developing models of associative memory and relational data bases (D. Champion, supervised by J. Rothstein). A characteristic difficulty is that queries or problems are presented in terms often quite different than the organization used for data storage. In effect there is a problem of language translation (hopefully capable of being done immediately) as well as problems of language representation and processing. While solid results are still sparse, there are grounds for belief that some can be obtained in a reasonable period.

(b,k)-Automata

We define (b,k)-automata as finite state machines which, when they process strings of digits (letters) in base b (alphabet of cardinality b), will classify them according to the remainder on division by k. Two simple examples for b=2, where k is 3 or 5 are given in Figure 2. The state labels are the residues 0, 1, ... (k-1), the arrow labels are 0, 1, ... b-1. The reason for interest in this class of machines is the ease of doing calculations in the residue number system. We have proved many



(a) k=3 (b) k=5

Figure 2 (a) (2,3)-machine; (b) (2,5)-machine.

interesting theorems about these machines, of which only a few will be cited here.

If and only if b and k are relatively prime will the corresponding machine be a group machine. The case b=1 corresponds to the cyclic group of order k, while b=k-1 gives the dihedral group D_{2k}, of order 2k. The transition graphs, for the group machines, are the same as the Cayley color graphs for the corresponding groups on b generators, the relations they satisfy being the set of closed directed paths (multiple cycles permitted). With no loss in generality we can take $1 \leq b \leq k$. The orders of all (b,k)-groups divide $k\,\phi(k)$ where k is Euler's totient function (number of integers prime to and less than k).

If k is a power of b, say the nth, we have what is essentially the state transition diagram of an n-delay shift register presented as a finite state machine. The corresponding graph is both Eulerian and Hamiltonian, and for b=2 gives precisely the well-known de Bruijn graphs, as well as a simple way of obtaining complete coding chains (this last for any base).

Work on these machines is continuing. They combine automata theory, group theory, graph theory and number theory in a very interesting way. In their immediate BA versions they may well be of great importance for immediate computation as well.

References

[1] J. Rothstein, "Heuristic Application of Solid State Concepts to Molecular Phenomena of Possible Biological Interest," Proceedings of the First National Biophysics Conference, Columbus 1957, Yale University Press (New Haven, 1959), pp. 77-85.

[2] J. Rothstein, "On Fundamental Limitations of Chemical and Bionic Information Storage Systems," IEEE Trans. Mil. Electronics, Vol. MIL-7 (1963), pp. 205-208.

[3] J. Rothstein, "Excluded Volume Effects as the Basis for a Molecular Cybernetics," in Cybernetic Problems in Bionics, Gordon and Breach 1968, pp. 229-245.

[4] J. Rothstein and P. James, "Families of Chain Configurations on the Quadratic Lattice and on Narrow Lattice Channels," Jour. Applied Physics, 38, pp. 170-179, (1967).

[5] J. Rothstein, "A Class of Formal Languages for Possible Biological Application, Proc. First European Biophysics Congress, Baden, 1971, pp. 259-262.

[6] J. Rothstein, "Generalized Entropy, Boundary Conditions, and Biology," in The Maximum Entropy Formalism, edited by R. Levine and M. Tribus, M.I.T. Press (Cambridge, Mass., 1979), pp. 423-468.

[7] J. Rothstein, "Generalized Life," Cosmic Search, 1, No. 2, 38 (March 1979).

[8] J. Rothstein, "Patterns and Algorithms," Proc. 9th Symposium on Adaptive Processes, IEEE Pub. 70C58-AC, Austin, Texas, December 1970.

[9] J. Rothstein, "Generalized Orthogonal Regression in Pattern Recognition," Proc. 1977 Intern. Conf. on Cybernetics and Society, IEEE Cat. No. 77CH 1259-1 SMC, pp. 232-6.

[10] J. Rothstein and C. F. R. Weiman, "Parallel and Sequential Specification of a Context Sensitive Language for Straight Lines on Grids," Computer Graphics and Image Processing, 5, pp. 106-124 (1976).

[11] J. Rothstein, "On the Ultimate Limitations of Parallel Processing," Proc. 1976 Intern. Conf. on Parallel Processing, IEEE Cat. No. 76CH 1127-0C, pp. 202-6. Best Paper Award.

[12] J. M. Moshell and J. Rothstein, "Bus Automata and Immediate Languages," Information and Control, 40, (1979), pp. 88-121.

[13] J. Rothstein, "Toward an Arithmetic for Parallel Computation," Proc. 1977 Intern. Conf. on Parallel Processing, IEEE Cat. No. 77CH1253-4C, pp. 224-233. Most Original Paper Award.

[14] J. Rothstein, "Transitive Closure, Parallelism, and the Modeling of Skill Acquisition," Proc. 1977 Intern. Conf. on Cybernetics and Society, IEEE Cat. No. 77CH1259-1 SMC, pp. 232-6.

[15] J. Rothstein, "Topological Pattern Recognition in Parallel and Neural Models on Bus Automata," 1978 Intern. Conf. on Parallel Processing, IEEE Cat. No. 78CH1321-9C, pp. 95-107.

[16] J. Rothstein and A. B. Davis, "Parallel Recognition of Parabolic and Conic Patterns by Bus Automata, Proc. 1979 Intern. Conf. on Parallel Processing, IEEE Cat. No. 79CH1433-2C, pp. 288-297.

[17] J. Rothstein, Parallel Processable Cryptographic Methods with Unbounded Practical Security," presented at the Internation Symposium on Information Theory, Cornell University, October 1977. Available as Technical Report OSU-CISRC-TR-77-9.

ABSTRACTS OF PH.D. DISSERTATIONS 1979-80

BAKER, ALBERT L. Software Science and Program Complexity Measures, The Ohio State University, Summer Quarter 1979.

There is considerable motivation for developing a measure of actual source programs which is indicative of the difficulty incurred by a programmer in generating or maintaining a particular code segment. Such a measure is called a program complexity measure. The software science effort measure E has received considerable empirical support as this type of measure. To lend additional credence to E as a program complexity measure, software effort is evaluated using an analytical methodology which relies on well-defined program transformations. One type of transformation used embodies principles of program modularity. The behavior of E under modularizing transformations is in agreement with accepted principles pertaining to such program changes. When E is evaluated in light of transformations related to program control flow, its utility as a general measure of program complexity is questioned.

This deficiency of E in quantifying control flow complexity motivates the analytical evaluation of other measures which purport to capture this factor. The program knots measure, K, and the normal number of a program, NN, are shown to be inadequate in certain significant respects. The graph theoretic measure cyclomatic complexity, C, is shown to be applicable to program flowgraphs, and C bears up well under analytical evaluation. Thus the feasibility of synthesizing a new program complexity measure from E and C is considered. One approach to this synthesis is shown to be well defined for structured programs. This new measure, PC, is evaluated both analytically and empirically with encouraging results.

Other research efforts have indicated that combining a measure of complexity incurred in straight line code with a measure of control flow complexity might prove fruitful. This approach has been carried out with the result that the goal of an adequate program complexity measure is now closer.

FLINCHBAUGH, BRUCE E. A Computational Theory of Spatio-Temporal Aggregation for Visual Analysis of Objects in Dynamic Environments. The Ohio State University, Spring Quarter 1980.

A theory of spatio-temporal aggregation is introduced as an explanation of what it is possible to compute at an early stage in the visual analysis of objects. Spatio-temporal aggregation is defined as the process of grouping together elements in an image sequence whose motions and positions have consistent interpretations as the retinal projections of a coherent or isolated cluster of 'particles' in the physical world. The information abstracted by spatio-temporal aggregation represents early decisions about which distinct elements belong to the same physical entities or object fragments and estimations of the motions of those fragments relative to the observer. The assumptions of confluence and adjacency are made in order to constrain the infinity of possible interpretations to a computationally more manageable

domain of plausible interpretations. It is shown that confluence and adjacency lead to the derivation of specific rules for grouping which permit the appropriate aggregation of rigid and quasi-rigid objects in motion and at rest under a variety of conditions. The theory is reconciled with existing computational theories of vision so as to complement them, and to provide a useful link in the continual abstraction of visual information.

JAPPINEN, HARRY. A Perception-Based Developmental Skill Acquisition System. The Ohio State University, Summer Quarter 1979.

The traditional approach to the design of robot behavior in Artificial Intelligence can be stated as follows: for a given task a problem solver produces an ad hoc plan and then a monitor interprets and executes the plan and continuously checks its correctness. Control information for a robot's behavior is provided by symbolic representations of world states (a world model). Since the problem solver can freely manipulate the world model, no matter how complex a task, future states of the world can be projected in the planning phase, and the plan can be so organized that an optimal behavioral pattern results from it. On the other hand, unless the robot's subworld is static and strongly restricted, total reliance on a world model may result in serious efficiency problems since even the minutest details of behavior need to be explicitly determined in the plan. Furthermore, the maintaining of a consistent world model may pose harsh problems.

In contrast, humans are capable not only of the above kind of behavior--we may call it inferred behavior--but also of reacting directly, and a part of human knowledge of the world is stored in action form. Everyday human behavior is a mixture of inferred and direct behavior. A plan determines only gross behavior while leaving control details in the care of earlier developed, task-specific action-schemes. Action-schemes receive control information directly from the trustworthy environment by using perceptions.

This work studies some aspects of direct behavior. We hope to gain insight into its advantages, disadvantages, and limitations of its use in robotics; and to determine how direct behavior might be interfaced with inferred behavior. For that end, we define a mini-language for well-formed skills and design a system which communicates with a human master, acquires new skills when needed, and executes existing skills for requested tasks. The system lacks a comprehensive world model: most of the system's 'knowledge' is in action form imbedded in its skills. Any one skill represents a hierarchical interlacing of movements and perceptions so that control information in an activated skill is provided by perceptions and/or recalling memory. The system does not initially possess any fixed level of competence, rather it develops in its ability to cope with physical tasks. Since skills strongly impose hierarchy, the system's competence gradually increases, and a need for the acquisition of new skills gradually decreases. In the implemented version of the proposed system the skill formation process is replaced by advice-taking from the human master. The system itself is capable of generalizing its skills.

We have implemented the proposed skill system and simulate its performance in a hypothetical 3-floor house in which a robot manipulates physical

objects. We hope to demonstrate in the simulation that, indeed, fairly complex behavior over a non-trivial set of tasks can be generated by a rather small collection of general skills. A future system which combines the power of planning and the efficiency of skills by using both a world model and representations of skills as operators would be an interesting advance.

KO, KER-I. Computational Complexity of Real Functions and Polynomial Time Approximation. The Ohio State University, Summer Quarter 1979.

Recursive analysis, the theory of computation of functions on real numbers, has been studied from various aspects. We investigate the computational complexity of real functions using the methods of recursive function theory. Partial recursive real functions are defined and their domains are characterized as the recursively open sets. We define the time complexity of recursive real continuous functions and show that the time complexity and the modulus of uniform continuity of a function are closely related. We study the complexity of the roots and the differentiability of polynomial time computable real functions. In particular, a polynomial time computable real function may have a root of arbitrarily high complexity and may be nowhere differentiable. The concept of nondeterministic computation is used to study the complexity of the integrals and the maximum values of real functions. These problems are shown to be new NP or PSPACE problems.

We define a strong polynomial time reducibility which preserves approximate solutions. We demonstrate its practicality by showing that a classification of NP-hard combinatorial problems according to their approximability can be made by this reducibility. Abstract properties of the reducibility are also discussed.

We show that the probabilistic algorithms, although more powerful than deterministic algorithms, are not likely able to solve NP-hard combinatorial problems in polynomial time. It is implied that the class of probabilistic polynomial time recognizable languages and the polynomial hierarchy are independent.

Finally, we investigate the average case approximation of exponential time computable sets (EXPTIME). Under a reasonable definition of the size of a set, we show the existence of a set which is not polynomial time approximable at all but which is polynomial time tt-complete in EXPTIME. On the other hand, polynomial time m-completeness guarantees an infinitely often speedup, but it does not provide any control of the probability of erroneous approximations.

KWASNY, STAN C. Treatment of Ungrammatical and Extra-Grammatical Phenomena in Natural Language Understanding Systems. The Ohio State University, Winter Quarter 1980.

This dissertation investigates several language phenomena commonly considered deviant by linguistic standards and proposes how they might be integrated into a computer model of understanding Natural Language. Techniques developed within the Augmented Transition Network (ATN) model are shown to be

adequate to relate many of these cases to well-formed counterparts. First, linguistic problems are defined and discussed and then techniques for overcoming these problems are presented. The techniques require a normative grammar be given which specifies the processing required of grammatical inputs. Our mechanisms work to relate ungrammatical inputs to appropriate grammatical ones and to process them accordingly. The mechanisms proposed include two techniques for relaxing restrictions in the grammar, a new arc type employing pattern matching, and a method for dynamically ordering the evaluation of arcs. The efficiency and structural characteristics of the original normative grammar are not altered by these mechanisms.

The simplest form of deviant sentence considered is one in which the anomaly is caused by the direct violation of a test or agreement criteria. A mechanism is developed which relaxes such tests in specified ways in order to circumvent the deviance.

Another form of deviance involves the violation of a category test. To allow for this, categories may be relaxed according to a hierarchy of categories used in the grammar. Relaxing one level in the hierarchy simply involves substituting the immediate parent category, which is a superset of the given category, along with all sibling categories to achieve a carefully controlled widening of the scope of words allowed.

More complex forms of ungrammatical inputs are successfully processed by essentially relaxing the grammar itself. In order to realize this effect, the notion of a "pattern" of ATN arcs is introduced along with several matching algorithms to allow a range of matching capabilities.

Although by itself the use of conjunction would rarely be considered ungrammatical, few Natural Language Understanding systems have been successful at including it directly in the grammar. Our belief is that it is properly treated only as an "extra-grammatical" notion. This is done by defining for a grammar external mechanisms which dynamically construct and apply arcs through the pattern mechanism.

Patterns are also shown to be important in the processing of elliptical sentences. Patterns which reflect context may be constructed dynamically and manipulated by the system in a manner which permits matching to be delayed until an appropriate point in the processing.

The power of these mechanisms is illustrated mostly through example. Arguments for their inclusion in the ATN formalism are also presented.

MELLBY, JOHN R. The Recognition of Straight Line Patterns by Bus Automatons Using Parallel Processing. The Ohio State University, Spring Quarter 1980.

Highly parallel computers are capable of solving some problems much more rapidly than sequential computers. We present algorithms on the Bus Automaton, a parallel computer. The Bus Automaton solves various computation problems immediately, i.e., in O(1) time, while they would take from O(1) to $O(n^2)$ time on sequential computers.

The primary immediate computation is the recognition of straight lines. Other problems also immediately computed are: mass transformations of data sets, arithmetic in stroke or binary notation, computation of statistical mean and variance, lexicographic sorting, and general arithmetic expression computation.

PARDO, ROBERTO. Interprocess Communication and Synchronization for Distributed Systems. The Ohio State University, Summer Quarter 1979.

A new type of computer system is emerging as a result of several factors. First, the decline in cost of hardware makes more computer systems available to more people within an organization or in different organizations. Second, organizations and users are typically distributed; rather than making them fit the structure of computer systems, systems should fit the structure of such organizations. Third, the success of computer-network technology has shown the feasibility of linking multiple computer systems, although at a very low-level. These factors are creating new computing environments, namely Distributed Computing Systems (or Distributed Systems for short). These systems are a collection of individual computer systems interconnected by a (long-haul, local, or combinations of both) communication system. The distinguishable characteristic of these systems is that besides the low-level coherence provided by the communication system (as in computer networks), many functions at higher levels (such as file systems, databases, applications) also cooperate harmoniously. This thesis focuses on two fundamental problems of the design of such systems: interprocess communication and synchronization.

In order to exploit the inherent concurrency of distributed systems, processes very often exchange messages simultaneously to multiple destinations. The first aspect of such a multi-destination interprocess communication facility deals with the underlying multi-destination routing algorithms. We develop strategies for routing multi-destination messages, mainly in local architectures.

Although routing directs messages, it does not guarantee a reliable exchange of messages. Thus, the second aspect of the exchange of multi-destination messages deals with the design of several protocols at various levels that ensure different degrees of reliability. Three levels are considered: (1) unreliable level, where the protocol implements process addressing and chooses the best multi-destination routing algorithm; (2) best-effort-to-deliver level, where the protocol attempts to deliver the message to as many of its destinations as possible, filtering duplicates and out of order messages at each destination; and (3) guarantee-to-deliver level, where the protocol ensures that every destination will receive the message, regardless of momentary unavailability due to communication or processing system failures.

The third aspect of interprocess communication relates to how programmers "see" and use the facility. We develop the notion of distributed programming and introduce the concept of "remote-operations" as an alternative to other primitives (such as send/receive or call/return) for inter-node communication.

The second fundamental problem attacked in this thesis is process synchronization in distributed systems. We examine in detail the differences between synchronization in centralized systems and distributed systems, and show the important role of message delays, efficiency, and reliability in distributed synchronization. Two distributed synchronization strategies are studied. First, an existing low-level model (Eventcounts and Sequencers) is extended and its performance is evaluated. Second, suggestions for a high-level model to be built into a distributed programming language are made. The advantages and difficulties of this latter approach are discussed.

TENG, ALBERT Y. Protocol Constructions for Communication Networks. The Ohio State University. Winter Quarter 1980.

Computer hardware components for data processing are becoming much more powerful while simultaneously becoming less expensive. To take advantage of this advance in hardware technology, many of today's data processing systems are designed as distributed computer networks. A well-defined set of rules, referred to as communication protocols, must be established to regulate the interactions between the attached host computers in a network. This research addresses various issues in constructing network communication protocols that are correct, modifiable and maintainable. Our research goal has been primarily directed towards practical techniques that enable these protocols to be implemented in an effective way and at acceptable cost. This research presents a systematic protocol construction methodology using both formal language and software engineering techniques. Our model uses a context-free grammar, called the Transmission Grammar (TG), to define, verify and implement protocols.

The TG model has been applied to various phases of protocol software development. The TG model is capable of specifying protocols which are more complicated than those modeled by the finite state automata. A significant feature of the TG model is its grammar integration operations. The arbitrary shuffle and substitution operations of protocol languages are applied to construct inherently correct protocols from validated components and/or independent parts. The step-wise TG specification allows the protocol designer to specify protocol program modules in a well-structured manner for protocol documentation. In addition, the specified grammar structure is kept so simple that automatic validation and implementation can be easily carried out.

Local validation techniques have been developed to detect simple TG structure errors, which are formally defined by grammar notations as follows: (1) left-recursion, (2) non-deterministic grammar, (3) undefined non-terminal, (4) superfluous rules and (5) improper termination. The corresponding algorithms for locating these structure errors are also outlined. These local validation techniques are first used to detect most of the simple protocol errors defined above before global validation.

Global validation techniques have also been developed to check the timing and interactions between communicating entities and to reveal protocol syntax errors. The validation automaton (VA) model is introduced to comprehensively represent the interdependency between communicating entities. Several complexity reduction rules are provided for reducing the number of states and transitions so as to alleviate the global state explosion problem. A valida-

tion checklist is also provided for systematically examining the correctness problems that may cause protocol syntax errors.

From a validated TG, proper semantics can be included in the TG for automatic software/hardware implementation of the validated protocol. The structure and architecture of a protocol "recognizer", called the General Prococol System, is also presented for automatic software implementation.

WOLF, JACOB J., III. Design and Analysis of The Distributed Double-Loop Computer Network (DDLCN). The Ohio State University. Summer Quarter 1979.

Computers are becoming much more powerful while simultaneously becoming less expensive and smaller in physical size. This proliferation of hardware has enhanced research efforts in many areas, of which computer networking and distributed processing are two such topics. This dissertation presents the design of a distributed-control, fault-tolerant double-loop computer network. Analytical and simulation models are presented and used to investigate the performance of the network under various fault-free conditions, and in the presence of faulty communication links.

The DDLCN is a double-loop computer network that uses completely distributed control to send information messages around the network. Each loop independently uses the DLCN transmission mechanism (shift-register insertion technique) to transmit messages during no-fault operation. (However, Pierce and Newhall transmission mechanisms may also be supported through firmware.) Thus, each loop is capable of supporting simultaneous transmission of variable length messages from any node to any other node on the network.

The use of two communication loops instead of one will naturally furnish a certain degree of fault-tolerance. However, with a static double-loop design, two link-faults (one on each loop) could render the entire network helpless, since either the information message or its acknowledgement would be blocked from its destination. To alleviate this problem the DDLCN incorporates tri-state control logic into the design of each Loop Interface Unit (LIU). This logic then allows any adjacent pair of interfaces to logically switch the transmission direction of the communication links between them. Thus, a link fault may be logically "tagged" onto either of the links in that pair.

Each LIU is capable of dynamically detecting a new link-fault and dynamically tagging that fault on the proper link of the newly failed link-pair so as to preserve the no-fault operation on one loop. The detection and recovery schemes are independently performed by each LIU, allowing the communication network to become dynamically fault-tolerant while still operating under totally distributed control.

Complete descriptions are presented for the detection and recovery schemes for each of the possible link-fault classes (single, single to first double, instantaneous first double, additional doubles). Basic detection and recovery schemes for LIU failures are also presented.

To investigate the performance of the DDLCN, an analytical model and simulation model are used to predict link (transmitter) utilizations on the network. This analysis allows message flow patterns to be derived for the network under any total message arrival rate and any link-fault situation. This makes it possible to predict possible traffic bottlenecks for non-homogeneous network configurations (a few dominant nodes).

The simulation model predicts all of the parameters calculated by the analytical model plus mean message transmission and queueing times. It is also used to investigate numerous dynamic message routing techniques in an effort to develop an optimal routing strategy.

In conclusion, a dynamically fault-tolerant computer network is presented that operates under totally distributed control. Detection and recovery schemes are presented for both link and interface faults. Analytical and simulation models are developed that predict message flow patterns and expected message transmission times. The simulation model is also used to investigate several dynamic message routing algorithms.

WONG, PATRICK M. K. A Methodology for the Definition of Data Base Workloads: An Extension to the IPSS Methodology. The Ohio State University. Autumn Quarter 1979.

Information system performance depends in large part on the data base workload. The data base workload is generally not considered in detail in the system design and evaluation process due to lack of systematic characterization and analysis procedures. In this research, a methodology for characterizing the data base workload for performance accessment purpose has been developed. As part of this methodology, procedures were developed for characterizing a) the distribution of data base attributes within the information system's data base in an implementation independent manner, and b) the structure of the user generated queries which constitute the system's input workload. Two transformation algorithms employing these characterizations complete the methodology. The first algorithm transforms the general prepositional based user query into one or more normalized Boolean expressions. The second algorithm transforms the Boolean expressions into data base responses. The latter is defined as the system's data base workload for a user query.

Discrete event simulation constructs have been developed based on methodological model. These simulation constructs have been incorporated into the Information Processing System Simulator's Modeling and Executional Facilities. These extensions provide a powerful addition to IPSS. With them it is easy to describe queries in terms of Boolean expressions and to determine system responses using only a few simulation language statements.

The IPSS language extensions have been implemented, tested and verified. It was found that the methodology is applicable to both conventional information storage and retrieval systems and relational data base management systems.

RESEARCH AND DEVELOPMENT AWARDS

Equipment Grant

Title:	Equipment for Research Programs in Database Computers
Principal Investigator:	David K. Hsiao
Investigator and Director:	Douglas S. Kerr
Sponsor:	External Research Program, Digital Equipment Corporation
Date:	August 1, 1980
Amount:	$222,155

Abstract:

A multi-mini computer system consisting of one VAX 11/780 and two PDP-11/44s interconnected with parallel transfer buses will be installed in the Laboratory for Database Systems Research. The VAX 11/780 is configured with two 67-mbyte disks, one tape, four terminals, one console, one line printer and 1.5-mbyte primary memory. One PDP-11/44 has two 67-mbyte disks, one tape, two terminals and 256-kbyte primary memory, while the other PDP-11/44 has one 67-mbyte disk and 256-kbyte primary memory.

Two research programs are to be conducted on the lab equipment. The first program, funded separately by the Office of Naval Research, is to seek an architecture providing high-performance for great-capacity database management utilizing multiple mini-computers operating in parallel with replicated software. The second program, which has not yet been funded, will be to emulate the architecture of a database computer, known as DBC, on the lab equipment for performance and design studies.

Graduate Training Grant

Title:	Graduate Training Program in Biomedical Computing and Information Processing
Program Directors:	A.E. Petrarca, Associate Professor, Department of Computer and Information Science Gregory L. Trzebiatowski, Associate Dean, College of Medicine
Sponsor:	National Library of Medicine (NIH Grant LM 07023)
Duration:	7/1/80-6/30/82
Amount:	$148,249 (1980-81); $145,452 (1981-82)

Abstract:

In order to meet the needs for specialists in biomedical computing, an interdisciplinary Graduate Training Program in Biomedical Computing and Information Processing was established through a joint effor of the School of Allied Medical Professions of the College of Medicine and the De-

partment of Computer and Information Science. Students who are interested in the study and application of computer and information science to health care, medical education, and biomedical research may pursue, through one of the participating departments, the graduate degrees of Master of Science (via Allied Medical Professions of Computer and Information Science) and Doctor of Philosophy (via Computer and Information Science or appropriate Ph.D. granting departments in the College of Medicine).

Research Grants

Title: Analyzing Program Methodologies Using Software Science

Principal Investigator: Stuart H. Zweben
Sponsor: U.S. Army Research Office (DAAG29-80-K-0061)
Duration: 8/1/80-7/31/83
Amount: $146,579

Abstract: The area of applicability of various Software Science metrics to COBOL will be extended by conducting controlled experiments and by developing appropriate software tools to automate the analysis of data. The extent to which the metrics might be appropriate in evaluating the quality of computer software will also be investigated.

Title: The Distributed Loop Computer Network

Principal Investigator: Ming T. Liu
Sponsor: National Science Foundation (MCS77-23496)
Duration: 6/1/78-5/31/80, Extended to 9/30/80
Amount: $71,104

Abstract: The Distributed Loop Computer Network (DLCN) is envisioned as a powerful distributed computing system which interconnects min.-midi computers, terminals and other peripheral devices through careful integration of hardware, software and a loop communication network. The prime aim of the proposed research is to investigate the feasibility that DLCN can provide efficient, inexpensive, reliable and flexible service for a localized user community.

Title: Extension and Application of a Theory of Information Flow and Analysis: The Use of Information in a Decision-Making Environment

Principal Investigator: Clinton R. Foulk
Sponsor: National Science Foundation, Div. of Information Science and Technology (DSI-76-21949)
Duration: 8/1/79-7/31/81
Amount: $121,513

Abstract: This research program builds on previous research which has been underway at Ohio State University for the last few years. We now plan to extend our research in three different but complimentary directions: 1) We are extending the basic theoretical work; 2) We are gathering additional data with the use of a flexible, sophisticated, simulation model in order to establish new relationships and important parameters; and 3) We are developing, designing and carrying out experiments involving human subjects in order to obtain real data about use of information by decision-makers.

Title: Properties of Axiomatic Data Specification

Principal Investigator: Daniel J. Moore
Sponsor: National Science Foundation (MCS78-02615)
Duration: 6/1/78-5/31/80, Extended to 11/30/80
Amount $75,563

Abstract: Properties of the process of axiomatic specification of abstract data types are to be studied. Primitive data types and the use of those primitive types in constructing user-defined data types will be investigated. The possibility of automatic completeness checking and implementation by compiler will be evaluated.

Title: Research in Data Security, Data Secure Systems, and Database Computers

Principal Investigator: David K. Hsiao
Investigators: Douglas S. Kerr and Richard R. Underwood
Sponsor: Office of Naval Research (N00014-67-A-0232; N00014-75-C-0573)
Duration: 3/1/73-2/29/80, Extended to 9/30/81
Amount: $771,620

Abstract: Present research focuses on the design of an architecture providing high-performance for great-capacity database management utilizing multiple mini-computer systems operating in parallel with replicated software.

Title: Theoretical Foundations of Software Technology

Principal Investigators: B. Chandrasekaran, L. J. White, H. W. Buttelmann
Sponsor: U.S. Air Force Office of Scientific Research (F49620-79-C-0152)
Duration: 7/1/79-6/30/81
Amount: $143,762

Abstract: This research will develop basic theoretical models and results in the areas of software and programming language structure and design, with the purpose of producing knowledge that will enable development of more reliable and transportable software. The current focus is on three areas: semi-automatic program testing, where an interactive prototype system is being developed; automatic program synthesis from algorithms stated in English; and computability and complexity issues in formal syntax and semantics, language definition, translation, and translator generation.

Title: Toward More Complicated Computer Imagery

Principal Investigator: Franklin C. Crow
Sponsor: National Science Foundation (MCS-7920977)
Duration: 1/15/80-1/31/81
Amount: $79,342

Abstract: Initial efforts will focus on the design of data structures to support efficient rendition of the same object at many different levels of detail. Subsequent work will focus on algorithms for the display of such objects and designed for distributed execution. Finally, the algorithms will be implemented and image sequences produced on a multicomputer facility.

APPENDIX A

CURRENT STATUS AND CAPSULE HISTORY OF DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

	SEPT '74	SEPT '75	SEPT '76	SEPT '77	SEPT '78	SEPT '79	SEPT '80
A. Staff							
1. Full Time	20	21	22	20	21	27	24
2. Part Time	12	12	12	13	12	14	12
B. Graduate Students	198	201	182	197	198	200	200 (est)
C. Undergraduate Students	475	450	470	470	470	470	470 (est)
D. Course Enrollment (Autumn Quarter)	1925	2098	2290	2308	2568	2928	3100 (est)

	'74-75	'75-76	'76-77	'77-78	'79-79	'79-80
Students Taught	6876	7241	7615	7528	8447	9420
Baccalaureate Degrees Awarded	109	103	118	125	126	124
M.S. Degrees Awarded	58	64	70	54	59	56
Ph.D. Degrees Awarded	7	13	5	8	7	10
Ph.D. Degrees Awarded- Total	23	36	41	49	56	66
Applications for Graduate Study	355	325	333	335	479	509
Number of Graduate Students Supported	81	77	81	92	72	70

APPENDIX B

COMPUTER AND INFORMATION SCIENCE COURSE LISTING
BY NUMBER AND TITLE

100 Computers in Society

201 Elementary Digital Computer Programming

211 Computer Programming for Problem Solving (Effective Autumn 1980)

212 Computer Data Processing

221 Programming and Algorithms I

222 Programming and Algorithms II

294 Group Studies

313 Introduction to File Design

321 Introduction to File Processing

380 File Design and Analysis

411 Design of On-Line Systems

489 Professional Practice in Industry (Effective Summer 1980)

493 Individual Studies (Effective Autumn 1980)

505 Fundamental Concepts of Computer and Information Science

511 Computer Systems and Programming for Administrative Sciences

541 Survey of Numerical Methods

542 Introduction to Computing in the Humanities

543 Intermediate Digital Computer Programming

548 Computer Science for High School Teachers

551 Database Systems

555 Survey of Programming Languages

557 Minicomputer Programming Systems

594 Group Studies (Effective Autumn 1980)

607 Mathematical Foundations of Computer and Information Science I

610 Principles of Man-Machine Interactions

640 Numerical Analysis

641 Computer Systems Programming I

642 Numerical Linear Algebra

643 Linear Optimization Techniques in Information Processing

644 Systems Programming (Withdrawn Autumn 1980)

660 Introduction to Operating Systems (Effective Autumn 1980)

675 Introduction to Computer Architecture (Effective Autumn 1980)

676 Minicomputer and Microcomputer Systems

677 Computer Networks

680 Data Structures

693 Individual Studies

694 Group Studies

694J Data Models and Database Systems (Number becomes 788.06D Winter 1981)

694K Current Topics in Computer and Information Science (Withdrawn Autumn 1980)

694L Biomedical Information Processing

694M Software Engineering (Number becomes 757 Winter 1981)

694O Introduction to Operating Systems (Number becomes 760 Autumn 1980)

694P Programming Languages (Number becomes 755 Autumn 1980)

694S Introduction to Operating Systems: Laboratory (Number becomes 761 Autumn 1980)

694U Elements of Computer Systems Programming

707 Mathematical Foundations of Computer and Information Science II

712 Man-Machine Interface

720 Introduction to Linguistic Analysis

726 Theory of Finite Automata

727 Turing Machines and Computability

728 Topics in Theory of Computing

730 Basic Concepts in Artificial Intelligence

735 Statistical Methods in Pattern Recognition

740 Computer Systems Programming II (Withdrawn Autumn 1980)

741 Comparative Operating Systems

745 Numerical Solution of Ordinary Differential Equations

746 Advanced Numerical Analysis

750 Modern Methods of Information Storage and Retrieval

751 Fundamentals of Document-Handling Information Systems

752 Techniques for Simulation of Information Systems

753 Theory of Indexing

755 Programming Languages

756 Compiler Design & Implementation

757 Software Engineering (Effective Autumn 1980)

760 Operating Systems (Effective Autumn 1980)

761 Introduction to Operating Systems: Laboratory (Effective Autumn 1980)

765 Management Information Systems

775 Computer Architecture (Effective Autumn 1980)

780 File Structures

781 Aspects of Computer Graphics Systems

788 Intermediate Studies in Computer and Information Science

788.01 - Theory of Information

788.02 - Information Storage & Retrieval

788.03 - Theory of Automata

788.04 - Artificial Intelligence

788.04A - Topics in Artificial Intelligence

788.05 - Pattern Recognition

788.06 - Computer Systems Programming

788.06A - Computer Center Organization and Management

788.06B - Computer Performance Organization

788.07 - Programming Languages

788.07A - Intermediate Studies in Programming Languages

788.08 - Computer Organization

788.09 - Numerical Analysis

788.09A - Numerical Solution of Ordinary Differential Equations

788.10 - Man-Machine Interaction

788.11 - Formal Languages

788.12 - Management Information Systems

788.12A - Seminar in Management Information Systems

788.13 - Biological Information Processing

788.14 - Socio-Psychological Aspects of Information Processing

793 Individual Studies

797 Interdeparemtnal Seminar

[illegible]mation Theory in Physical [illegible]e

806 Cellular Automata & Models of Complex Systems

812 Computer & Information Science Research Methods

820 Computation Linguistics

835 Special Topics in Pattern Recognition

845 Numerical Solution of Partial Differential Equations

850 Theory of Information Retrieval I

852 Design and Analysis of Information Systems Simulations

855 Advanced Topics in Programming Languages

880 Advanced Theory of Computability

888 Advanced Studies in Computer & Information Science

888.01 - Theory of Information

888.02 - Information Storage & Retrieval

888.03 - Theory of Automata

888.03B - Cellular Automata and Models of Complex Systems

888.04 - Artificial Intelligence

888.05 - Pattern Recognition

888.06 - Computer Systems Programming

888.06A - Data Models and Database Systems

888.07 - Programming Languages

888.08 - Computer Organization

888.09 - Numerical Analysis

888.10 - Man-Machine Interaction

888.11 Formal Language

888.12 - Management Information Systems

888.13 - Biological Information Systems

888.14 - Socio-Psychological Aspects of Information Processing

889 Advanced Seminar in Computer and Information Science

894 Group Studies

899 Interdepartmental Seminar

999 Research

APPENDIX C

COMPUTER AND INFORMATION SCIENCE FACULTY

Professors

Lee J. White, Ph.D., (University); Chairperson of the Department of Computer and Information Science. Algorithm analysis and complexity, data structures, software engineering, program testing. Joint appointment with Electrical Engineering.

Kenneth J. Breeding, Ph.D., (University of Illinois). Computer organization and switching theory. Joint appointment with Electrical Engineering.

Balakrishnan Chandrasekaran, Ph.D., (University of Pennsylvania); artificial intelligence, pattern recognition, interactive graphics, finite memory decision theory.

Charles A. Csuri, M. A., (The Ohio State University). Advancement of computer graphics technology in software and hardware (language algorithms, data generation of inputs), use of computer technology in telecommunications. Joint appointment with Art.

Tse-yun Feng, Ph.D., (University of Michigan). Parallel processors and processing, interconnection networks, computer architecture. Appointment Autumn 1980.

Richard I. Hang M. S., (The Ohio State University). Computer graphics, engineering application of computers. Joint appointment with Engineering Graphics.

David K. Hsiao, Ph.D., (University of Pennsylvania). Systems programming, computer architecture, database management systems, access control and privacy protection of data, and database computers.

Clyde H. Kearns, M. S., (The Ohio State University). Computer graphics, engineering application of computers. Joint appointment with Engineering Graphics.

Robert S. Larue, P. E., M. S., (University of Idaho). Computer graphics, engineering application of computers. Joint appointment with Engineering Graphics.

Ming-Tsan Liu, Ph.D., (University of Pennsylvania). Computer architecture and organization, computer communications and networking; parallel and distributed processing; mini/micro computer systems.

Robert B. McGhee, Ph.D., (University of Southern California). Robotics, switching theory, logical design. Joint appointment with Electrical Engineering.

Roy F. Reeves, Ph. D., (Iowa State University). Director, Instruction and Research Computer Center, and Professor of Mathematics. Numerical analysis, programming, and computer center management.

Jerome Rothstein, A.M., (Columbia University). Information and entropy, foundations of physics, methodology, biocybernetics, automata theory, formal languages, cellular automata, parallel processing. Joint appointment with Biophysics.

Charles Saltzer, Ph.D., (Brown University). Coding theory, numerical analysis, automata theory. Joint appointment with Mathematics.

Associate Professors

H. William Buttelmann, Ph.D., (University of North Carolina). Formal language theory, computational linguistics, language processing, programming languages.

Ronald L. Ernst, Ph.D., (University of Wisconsin). Man-computer interaction, decision systems, and general theory of human performance. Joint appointment with Psychology.

Clinton R. Foulk, Ph.D., (University of Illinois). Systems programming, computers in education.

Douglas S. Kerr, Ph.D., (Purdue University). Programming, database systems and numerical analysis.

James C. Kinard, Ph.D., (Stanford University). Accounting, management information systems, managerial decision making. Joint appointment with Accounting.

Sandra A. Mamrak, Ph.D., (University of Illinois). Computer system performance evaluation, computer networks, systems programming.

William F. Ogden, Ph.D., (Stanford University). Language theory, computer security, complexity theory, methodology for the design, implementation and verification of computer systems. Appointment Autumn 1980.

Anthyony E. Petrarca, Ph.D., (University of New Hampshire). Automatic indexing, chemical structural information processing, automated search systems, other aspects of information storage and retrieval, biomedical information processing.

Stuart H. Zweben, Ph.D., (Purdue University). Programming languages, programming methodology, analysis of algorithms, data structures, systems programming.

Adjunct Associate Professors

James R. Randels, Ph.D., (The Ohio State University). Assistant Director, University Systems Computer Center. Computer operating systems and utilities, telecommunications applications, subroutine libraries, programming languages.

James E. Rush, Ph.D., (University of Missouri). Director of R&D, OCLC. Indexing theory, automated language processing, organization of information, parallel processing, structured programming, program testing, and programming management.

Ronald L. Wigington, Ph.D., (University of Kansas). Director of R&D, Chemical Abstracts Service. Computer and information system design.

Assistant Professors

Bruce W. Ballard, Ph. D., (Duke University). Programming languages, natural languages, and program synthesis.

Ramamoorthi Bhaskar, Ph. D., (Carnegie-Mellon University). Accounting, artificial intelligence, cognitive psychology, managerial decision-making. Joint appointment with Accounting.

Frank C. Crow, Ph. D., (University of Utah). Computer graphics. Computer-aided design, multiprocessor and special purpose computer architecture.

Subrata Dasgupta, Ph. D. (University of Alberta). Microprogramming, computer architecture, concurrent programming.

Dennis Leinbaugh, Ph. D., (University of Iowa). Operating systems; hard-real time and process synchronization. Appointed Summer 1980.

Timothy J. Long, Ph. D., (Purdue University). Complexity theory, theory of computation, and algorithm analysis.

Sanjay Mittal, Ph. D., (The Ohio State University). Artificial intelligence, knowledge-based systems, data base modeling. Appointment Autumn 1980.

Daniel J. Moore, Ph. D., (University of Kansas). Complexity theory, recursion theory, semantics of simulation systems, formal theories of data abstraction.

Kamesh Ramakrishna, Ph. D., (Carnegie-Mellon University). User-Computer interaction, computational complexity, cognitive models of learning and instruction in complex task domains. Appointment Autumn 1980.

Jayashree Ramanathan, Ph. D., (Rice University). Programming languages, computer systems, and software engineering.

Richard R. Underwood, Ph. D., (Stanford University). Numerical linear algebra, solution of large sparse systems of equations, eigenvalue analysis, linear least squares problems, numerical solution of partial differential equations.

Bruce W. Weide, Ph. D., (Carnegie-Mellon University). Analysis of algorithms, computational complexity, data structures, combinatorics, computer architecture, parallel and distributed computing, real-time programming.

Adjunct Assistant Professor

Richard E. Parent, Ph. D., (The Ohio State University). Associate Director, Computer Graphics Research Group. Artificial intelligence, computer architecture, computer graphics.

Visiting Faculty 1979-80

Toby S. Berk, Ph. D., (Purdue University). Computer graphics, operating systems, programming methodology and languages. Appointment Autumn 1980.

K. V. S. Bhat, Ph. D., (University of Hawaii). Analysis and design of algorithms, assembler and programming languages, computer applications, data structures and databases, on-line computer systems design.

David J. H. Brown, Ph. D., (Witwatersrand). Artificial intelligence, information retrieval.

Bruce Flinchbaugh, Ph. D., (The Ohio State University). Artificial intelligence, computer vision. Appointment Autumn 1980.

P. Govinda Reddy, Ph. D., (Indian Institute of Technology, New Delhi). File organization, data security, distributed data base systems. Appointment Autumn 1980.

R. M. K. Sinha, Ph. D., (Indian Institute of Technology, Kharagpur). Pattern analysis, natural language processing, design of Indian language peripherals and computer aided printing. Appointment Autumn 1980.

Charles J. Shubra, Jr., M. S. (Pennsylvania State University). Database management, management information systems, programming methodology.

Neelamegam Soundararajan, Ph. D., (Bombay University). Theory of computation, semantics of programming languages, semantics of parallel processing.

Lynn Ziegler, Ph. D., (University of Michigan). Language design, computational complexity, approximation theory. Appointment Autumn 1980.

Administrative and Professional Staff

Ernest Staveley, B. S., (U. S. Naval Post-Graduate School). Administrative Associate, and Assistant Director of CIS Research Center.

Celianna Taylor, B.S.L.S., (Graduate School of Library Science, Case-Western Reserve University). Senior Research Asoociate and Associate Professor of Library Administration. Database design (natural language data), information (natural language) system design, development and implementation.

Faculty appointments, leaves of absence, and resignations

Toby S. Berk was appointed Visiting Associate Professor of Computer and Information Science, on leave from Florida International University. He received his Ph.D. in 1972 from Purdue University. His areas of interest are computer graphics, operating systems, programming methodology and languages, computer science education, computers and society.

K.V.S. Bhat resigned effective September 30, 1980. He will be Associate Professor at the University of Iowa.

Subrata Dasgupta resigned effective September 30, 1980. He will be Associate Professor at the University of Alberta, Canada.

Ronald L. Ernst was granted a leave of absence for the 1980-81 academic year. He will be Visiting Associate Professor in the Department of Computer Science, North Carolina State University.

Tse-yun Feng was appointed Professor of Computer and Information Science. He received his Ph. D. in 1967 from the University of Michigan and comes from Wright State University. Dr. Feng is an internationally renowned leader in associative processors and parallel processing. He is the founder of the Annual International Conference on Parallel Processing which had its beginning in 1972. Recent research is concerned with multistate interconnection networks.

Bruce E. Flinchbaugh was appointed Visiting Assistant Professor. He received his Ph. D. from The Ohio State University in 1980. Summer of 1980 was spent at M.I.T. on a project in the Department of Psychology and Artificial Intelligence Laboratory. His areas of interest are artificial intelligence and computer vision.

Dennis Leinbaugh was appointed Assistant Professor of Computer and Information Science and comes from the University of Nebraska. He received his Ph. D. from the University of Iowa in 1975. His areas of interest are operating systems: hard-real-time and process synchronization.

Howell H. W. Mei resigned effective September 30, 1980 and will continue his work with the Director of Quality Assurance, U. S. Army Computer Systems Command at Fort Belvoir, Virginia.

Sanjay Mittal was appointed Assistant Professor of Computer and Information Science. He received his Ph. D. from the Department in August, 1980 and his areas of interest are in artificial intelligence, knowledge-based systems, data base modeling. Appointment Autumn 1980.

Daniel J. Moore resigned effective June 30, 1980 and has accepted a position at Bell Telephone Laboratories at Holmdel, New Jersey.

William F. Ogden was appointed Associate Professor of Computer and Information Science and comes from The University of Michigan. He received his Ph. D. from Stanford University in 1969. His areas of interest are language theory, complexity theory, computer security, methodology for the design, implementation and verification of computer systems.

Kamesh Ramakrishna was appointed Assistant Professor of Computer and Information Science and comes from Carnegie-Mellon University where he has recently received his Ph. D. His areas of interest are user-computer interaction, computational complexity, cognitive models of learning and instruction in complex task domains.

P. Govinda Reddy was appointed Visiting Professor of Computer and Information Science beginning Autumn Quarter 1980. He received his Ph. D. in applied mathematics from the Indian Institute of Technology, New Delhi, in 1967, and the D.I.C. in computer science from the Imperial College, London in 1970. He is currently Chairman of the Department of Computer Science, Indian Institute of Technology, New Delhi. His areas of interest are file organization, data security, distributed data base systems.

Lawrence L. Rose resigned effective June 30, 1980 and will continue as a staff scientist at Battelle Memorial Research Institute in Columbus, Ohio; he will continue to be associated with the Department as Adjunct Associate Professor.

R.M.K. Sinha was appointed Visiting Assistant Professor of Computer and Information Science beginning Autumn Quarter 1980. He received his Ph.D. from the Indian Institute of Technology, Kharagpur and is currently Assistant Professor in the Department of Electrical Engineering and Computer Science Program, Indian Institute of Technology, Kanpur.

Lynn Ziegler was appointed Visiting Instructor for the 1980-81 academic year. He received a Ph.D. in mathematics from the University of Michigan in 1977 and is entering the graduate program in computer and information science at OSU. His areas of interest are language design, computational complexity, and approximation theory.

APPENDIX D

COMPUTER AND INFORMATION SCIENCE SEMINAR SERIES

October 11, 1979 "Bus Automata", Jerome Rothstein, Professor of Computer and Information Science and Professor, Department of Computer and Information Science, and Professor, Sensory Biophysics, The Ohio State University.

October 17, 1979 "Future's Research and the Computing Industry", Mr. Hank E. Koehn, Vice President Future Research Division, Security Pacific National Bank, Los Angeles, California.

October 25, 1979 "Very Fast Information Update and Retrieval Using Cells", Bruce W. Weide, Assistant Professor, Department of Computer and Information Science, The Ohio State University.

October 29, 1979 "An Experimental Study of Natural Language Programming", Alan W. Biermann, Associate Professor, Duke University, Durham, North Carolina.

November 8, 1979 "Overview of MDX - A System for Medical Diagnosis", Sanjay Mittal, Doctoral Candidate, Department of Computer and Information Science, The Ohio State University.

November 15, 1979 "Visual Computations for Analyzing Moving Objects", Bruce E. Flinchbaugh, Doctoral Candidate, Department of Computer and Information Science, The Ohio State University.

January 10, 1980 "An Organizational Information Processing Extension to the Information Economics Model", Jim Kinard, Associate Professor, Faculty of Accounting, College of Administrative Science, The Ohio State University.

January 17, 1980 "Everything You Always Wanted to Know about Reducibilities (.... but were afraid to ask)", Timothy J. Long, Visiting Assistant Professor, Department of Computer and Information Science, The Ohio State University.

January 24, 1980 "A Semantic Preserving Transformer for Assertion Based Languages", R. Krishnaswamy, Assistant Professor, Department of Computer Science, Iowa State University.

January 31, 1980 "A Data Link Encryption System", Frank H. Myers, Supervisor, Signaling Link Design Group, Bell Laboratories, Columbus.

February 5, 1980 "A Tree Machine for Searching Problems", Jon Louis Bentley, Departments of Computer Science and Mathematics, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 15213.

February 6, 1980 "A Methodology for Database Design", Ramez El-Masri, Computer Science Department, Stanford University.

February 7, 1980 "Knowledge-Driven Search in Large Dynamic Domains", Walter Reitman, Computer Science and Psychology, University of Michigan.

February 21, 1980 "Symbolic Evaluation in Program Testing", Lori A. Clarke, Assistant Professor, Department of Computer and Information Science University of Massachusetts, Amherst.

February 25, 1980 "A Reference String Sampling Method", W. James Wittneben, Assistant Professor, Department of Computer Science, Iowa State University.

February 26, 1980 "Natural Language Acquisition by Procedurally Extended Grammars", Mark A. Jones, University of Kansas.

February 28, 1980 "An Integrated Testing, Verification and Documentation System", Leon J. Osterweil, Associate Professor, Department of Computer Science, University of Colorado at Boulder.

March 5, 1980 "Data Base Encryption with Subkeys", David L. Wells, Department of Computer Science, University of Wisconsin - Milwaukee.

March 6, 1980 "Can Machines be Selfish?", Andrew Oldenquist, Professor, Philosophy Department, The Ohio State University.

March 7, 1980 "An Adaptive Main Memory Access Strategy", Hussein G. Badr, Computer Science Department, Pennsylvania State University.

March 10, 1980 "Guaranteed Response Times in a Hard-Real-Time Environment", Dennis Leinbaugh, Computer Science Department, University of Nebraska.

March 11, 1980 "Parallel Processing: Languages, Schedules, and Performance Results", Paul F. Reynolds, University of Texas at Austin.

March 13, 1980 "Algorithm Analysis to Reduce Interconnection Complexity on Limited Interconnection Networks and VLSI Architecture", Robert H. Kuhn, Assistant Professor, Department of Computer Science, University of Illinois at Urbana-Champaign.

March 17, 1980 "Organization and Retrieval from Long-Term Episodic Memory", Janet Kolodner, Computer Science Department, Yale University.

March 19, 1980 "Processor-Rich Computer Architectures: The Relational Data Base Example", Roger Schultz, Computer Science Department, Iowa State University.

April 2, 1980 "Guaranteed Component-Wise Error Bounds for Approximate Solutions of Nonlinear Two-Point Boundary Value Problems Using an Improved Kantorovich-Like Theorem", Ying-Da Lee, University of Wisconsin.

April 9, 1980 "UNIX and the Programmer's Workbench", Dr. John R. Mashey, Supervisor, LCAMOS Design Group, Bell Laboratories.

April 11, 1980 "Fault Diagnosis of Multistage Interconnection Networks", Dr. Tse-Yun Feng, Wright State University.

April 15, 1980 "Constructing ZOG Networks: A Study of Schematization as an Aid to Organizing Knowledge", Kamesh Ramakrishna, Carnegie-Mellon University, Pittsburgh, PA.

April 24, 1980 "Knowledge Structuring", Mark S. Fox, Carnegie-Mellon University, Pittsburgh, PA.

May 2, 1980 "Formal Specifications: A Critical Component of Hierarchical Programs", William F. Ogden, University of Michigan.

May 8, 1980 "Automatic Database System Conversion Under Schema Transformation", Ben Shneiderman, Associate Professor, Department of Computer Science, University of Maryland.

May 22, 1980 "TRIAD", Jayashree Ramanathan, Assistant Professor, Department of Computer and Information Science, The Ohio State University.

May 29, 1980 "Base-Modular Automata and Their Applications", Jerome Rothstein, Professor, Department of Computer and Information Science, and Professor, Sensory Biophysics, The Ohio State University.

APPENDIX E

PUBLICATIONS OF THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE STAFF

BALLARD, B. W.; BIERMANN, A. W. Toward Natural Language Computation. In: American Journal of Computational Linguistics, April-June 1980, pp. 71-86.

BROWN, D. C.; KWASNY, S. C., CHANDRASEKARAN, B.; SONDHEIMER, N. K. An Experimental Graphics System with Natural Language Input. In: Computers and Graphics, Vol. 4, pp. 13-22, 1979.

CHANDRASEKARAN, B.; GOMEZ, F.; MITTAL, S.; SMITH, J. An Approach to Medical Diagnosis Based on Conceptual Structures. In: PRoceedings of the VI International Joint Conference on Artificial Intelligence, Tokyo, August 20-23, 1979.

CHANDRASEKARAN, B.; MITTAL, S.; SMITH, J. W. RADEX--Toward a Computer-Based Radiology Consultant. In: Pattern Recognition in Practice, Gelsema and Kanal (Editors), North Holland-Publishers, 1980.

CHANDRASEKARAN, B. Guest Editorial: Special Collection on Program Testing. In: IEEE Transactions on Software Engineering, Vol. SE-6, No. 3, May 1980.

CHOU, C.; LIU, M. T. A Concurrency Control Mechanism and Crash Recovery for a Distributed Database System (DLDBS). In: Distributed Data Bases, C. Delobel and W. Litwin (Editors), North-Holland Publishers, 1980.

HSIAO, D. K.; KERR, D. S.; MADNICK, S. E. Privacy and Security of Data Communications and Data Bases. In: Issues in Data Base Management, H. Weber and A. I. Wasserman (Editors), North-Holland Publishers, 1979, pp. 223-248.

HSIAO, D. K.; KERR, D. S.; NEE, C. Database Access Control in the Presence of Context Dependent Protection Requirements. In: IEEE Transactions on Software Engineering, Vol. SE-5, No. 4, July 1979, pp. 349-358.

HSIAO, D. K.; KERR, D. S.; MADNICK, S. E. Computer Security (part of the "ACM Monograph Series"), Academic Press, August 1979, 299 pages.

HSIAO, D. K. Data Base Computers. In: Advances in Computers, Vol. 19. Academic Press, Inc., 1980.

BANERJEE, J.; HSIAO, D. K.; NG, F. K. Database Transformation, Query Translation. and Performance Analysis of a New Database Computer in Supporting Hierarchical Database Management. In: IEEE Transactions on Software Engineering, Vol. SE-6, No. 1, January 1980, pp. 91-109.

LIU, M. T.; PARDO, R.; TSAY, D.; WOLF, J. J.; WEIDE, B. W.; CHOU, C. System Design of the Distributed Double-Loop Computer Network (DDLCN). In: Proceedings of the First International Conference on Distributed Computing Systems, Huntsville, Alabama, October 1-4, 1979.

TSAY, D.; LIU, M. T. Interface Design for the Distributed Double-Loop Computer Network (DDLCN). In: Conference Record of the National Telecommunications Conference, Washington, D. C., November 27-29, 1979. Volume 3. IEEE Cat. No. 79CH1514-9.

TENG, A. Y.; LIU, M. T. The Transmission Grammar Model for Protocol Construction. In: Proceedings, Trends and Applications: 1980, Computer Network Protocols. National Bureau of Standards, Gaithersburg, MD, May 29, 1980.

LIU, M. T.; MAMRAK, S. A.; RAMANATHAN, J. The Distributed Double-Loop Computer Network (DDLCN). In Proceedings of the ACM '80 Annual Conference, Nashville, TN, October 27-29, 1980.

MAMRAK, S. A.; AMER, P. D. Comparing Interactive Computer Services: Theoretical, Technical and Economic Feasibility. In: AFIPS Conference Proceedings, National Computer Conference, New York City, Vol. 48, June 1979, pp. 781-7.

MAMRAK, S. A. Computer Selection: To Measure or Not to Measure. In: National Bureau of Standards Special Publication 500-52, 15th Meeting Computer Performance Evaluation Users Group, San Diego, October 1979, pp. 37-52.

MAMRAK, S. A.; ABRAMS, M. D. A Taxonomy for Valid Test Workload Generation. In: Computer, December 1979, pp. 60-65.

MITTAL, S.; CHANDRASEKARAN, B.; SMITH, J. Overview of MDX - A System for Medical Diagnosis. In Proceedings III Annual Symposium on Computer Applications in Medical Care, Washington, D. C., October 14-17, 1979.

MITTAL, S.; CHANDRASEKARAN, B. Conceptual Representation of Patient Data Bases. In: Proceedings of the 13th Hawaii International Conference on Systems Science, January 3-4, 1980. This paper has been selected as one of the best papers at the Conference and will also appear in the Journal of Medical Systems.

MITTAL, S.; CHANDRASEKARAN, B. Organizing Data Bases Involving Temporal Information. In: Proceedings of the 1980 International Conference on Cybernetics and Society.

PARDO, R.; LIU, M. T. Multi-Destination Protocols for Distributed Systems. In: Proceedings of the Computer Networking Symposium, National Bureau of Standards, Gaithersburg, MD, December 12, 1979.

KENNEDY, K.; RAMANATHAN, J. A Deterministic Attribute Grammar Evaluator Based on Dynamic Sequencing. In: ACM Transactions on Programming Languages and Systems, Vol. 1, No. 1, July 1979, pp. 142-160.

ROTHSTEIN, J.; DAVIS, A. Parallel Recognition of Parabolic and Conic Patterns by Bus Automata. In: Proceedings of the 1979 International Conference on Parallel Processing, IEEE Cat. No. 79CH1433-2C, pp. 288-297.

ROTHSTEIN, J. Review of Frontiers in Visual Science, eds. S.J. Cool and E. Smith, III. In: Applied Optics, 18, 3080 (1979).

ROTHSTEIN, J. Review of a collection of conference preprints presented at the Seventh Symposium on Photo-Electronic Images Devices. In: Applied Optics, Vol. 19, No. 2, January 15, 1980, p. 300.

ROTHSTEIN, J. Review of Electronic Displays, by S. Sherr. In: Applied Optics, 19, 1669 (1980).

WANG, P.; LIU, M. T. Parallel Processing of High-Level Language Programs. In: Proceedings of the 1979 International Conference on Parallel Processing, Bellaire, Michigan, August 21-24, 1979.

WANG, P.; LIU, M. T. A Multi-Microprocessor System for Parallel Computations. In: Proceedings of the ACM Second Annual Symposium on Small Systems, Dallas, TX, October 1-3, 1979.

WANG, P.; LIU, M. T. The Architecture of a Parallel Execution High-Level Language Computer. In: Proceedings of the International Workshop on High-Level Language Computer Architecture, May 26-28, 1980.

WEIDE, B. W. Very Fast Information Update and Retrieval Using Cells. In: Proceedings of the 17th Annual Allerton Conference on Communication, Control, Computing, University of Illinois, October 1979.

WHITE, L. J.; COHEN, E. I. A Domain Strategy for Computer Program Testing. In: IEEE Transactions on Software Engineering, Vol. SE-6, No. 3, pp. 247-257, May 1980.

WHITE, L. J.; COHEN, E. I.; CHANDRASEKARAN, B. Discussion of 'A Survey of Program Testing Issues' by John B. Goodenough. Discussant item in: Recent Directions in Software Technology, MIT Press, 1979.

WOLF, J. J.; WEIDE, B. W.; LIU, M. T. Analysis and Simulation of the Distributed Double-Loop Computer Network (DDLCN). In: Proceedings of the Computer Networking Symposium, NBS, Gaithersburg, MD, December 12, 1979.

WU, S. B.; LIU, M. T. A Generalized Cluster Structure for Large Multi-Microcomputer Systems. In: Proceedings of the 1979 International Conference on Parallel Processing, Bellaire, Michigan, August 21-24, 1979.

WU, S. B.; LIU, M. T. Optimal Interconnection Design for Large Multi-Microcomputer Systems. In: Proceedings of the Workshop on Interconnection Networks for Parallel and Distributed Processing. April 21-22, 1980.

YOVITS, M. C. Advances in Computers (Editor). Academic Press, October 1979.

ZWEBEN, S. H. Heads I Win, Tails You Lose. A solicited letter In: the "Surveyor's Forum", ACM Computing Surveys, Vol. 11, No. 3, September 1979, pp. 277-278.

RAIHA, K.; ZWEBEN, S. H. An Optimal Insertion Algorithm for One-Sided Height-Balanced Binary Search Trees. In: Communications of the ACM, Vol. 22, No. 9, September 1979, pp. 508-512.

ZWEBEN, S. H. Contributing author to Advances in Software Science (Author: M. Halstead). In: Advances in Computers, Vol. 18. M. C. Yovits (Editor), Academic Press, New York, 1979, pp. 119-172.

ZWEBEN, S. H.; BAKER, A. L. A Comparison of Measures of Control Flow Complexity. In: Proceedings of the 1979 Computer Software and Applications Conference, Chicago, November 1979, pp. 695-701.

ZWEBEN, S. H.; FUNG, K. C. Exploring Software Science Relations in COBOL and APL. In: Proceedings of 1979 Computer Software and Applications Conference, Chicago, November, 1979, pp. 702-707.

APPENDIX F

RECENT TECHNICAL REPORTS

1978

HSIAO, D. K.; KRISHNAMURTHI, K. Simulation studies of the database computer (DBC). February 1978. 39 pp. (OSU-CISRC-TR-78-1) (AD-A056 048/2GI).

WHITE, L. J.; TENG, F. C.; KUO, H.; COLEMAN, D. An error analysis of the domain testing strategy. December 1978. 93 pp. (OSU-CISRC-TR-78-2).

BAKER, A. L.; ZWEBEN, S. H. The use of software science in evaluating modularity concepts. July 1978. 30 pp. (OSU-CISRC-TR-78-3).

WHITE, L. J.; COHEN, E. I.; CHANDRASEKARAN, B. A domain strategy for computer program testing. August 1978. 69 pp. (OSU-CISRC-TR-78-4) (AD-A067 552/0GA)

YOVITS, M. C.; ROSE, L. L. Information flow and analysis: Theory, simulation, and examples. Part I. Basic theoretical and conceptual development. Part II. Simulation, examples and results. September 1978. 77 pp. (OSU-CISRC-TR-78-5) (PB-293 458/6GA).

DELUTIS, T. G. The Information Processing System Simulator (IPSS). "Language syntax and semantics for the IPSS modeling facility" Volume I. 481 pp. (OSU-CISRC-TR-78-6).

ROSE, L. L.; CARR, G. G. Modeling the SSA Processes. December 1978. 51 pp. (OSU-CISRC-TR-78-7).

1979

BANERJEE, J.; HSIAO, D. K. Parallel bitonic record sort- an effective algorithm for the realization of a post processor. March 1979. 22 pp. (OSU-CISRC-TR-79-1) (AD-A068 661/8GA).

BANERJEE, J.; HSIAO, D. K.; MENON, J. The clustering and security mechanisms of a database computer (DBC). April 1979. 112 pp. (OSU-CISRC-TR-79-2) (AD-A068 815/0GA).

DELUTIS, T. G.; CHANDLER, J. S. The Information Processing System Simulator (IPSS). "Language syntax and semantics for the IPSS execution facility - Version 1" Volume I. 309 pp. (OSU-CISRC-TR-79-3).

DELUTIS, T. G.; CHANDLER, J. S.; BROWNSMITH, J. D.; WONG, P.; JOHNSTON, K. The Information Processing System Simulator (IPSS) "Language syntax and semantics for the IPSS Modeling Facility" Volume II. 1979. 325 pp. (OSU-CISRC-TR-79-4).

ROSE, L. L.; O'CONNOR, T. IDAS: Interactive design and analysis for simulation. 1979. 82 pp. (OSU-CISRC-TR-79-5).

HSIAO, D. K.; MENON, J. The post processing functions of a database computer. July 1979. 34 pp. (OSU-CISRC-TR-79-6).

CHANDLER, J. S. A multiple goal program model for the analysis of SSA district office service processing. August 1979. 33 pp. (OSU-CISRC-TR-79-7).

DELUTIS, T. G.; BROWNSMITH, J. D.; CHANDLER, J. S.; WONG, P. M. K. Methodologies for the performance evaluation of information processing systems. September 1979. 201 pp. (OSU-CISRC-TR-79-8).

1980

BLATTNER, M.; RAMANATHAN, J. TRIAD: A New Approach to Programming Methodology. January 1980. 48 pp. (OSU-CISRC-TR-80-1).

MAMRAK, S. A.; RAMANATHAN, J. A Programming/Operating System for a Distributed Computer System. February 1980. 28 pp. (OSU-CISRC-TR-80-2).

HSIAO, D. K.; MENON, J. Design and Analysis of Update Mechanisms of a Database Computer (DBC). June 1980. 127 pp. (OSU-CISRC-TR-80-3).

YOVITS, M. C.; FOULK, C. R.; ROSE, L. L. Information Flow and Analysis: Theory, Simulation, and Experiments. December 1979. 83 pp. (OSU-CISRC-TR-80-4).

FLINCHBAUGH, B. E.; CHANDRASEKARAN, B. A Theory of Spatio-Temporal Aggregation for Vision. April 1980. 43 pp. (OSU-CISRC-TR-80-5).

ZEIL, S. J.; WHITE, L. J. Sufficient Test Sets for Path Analysis Testing Strategies. July 1980. 29 pp. (OSU-CISRC-TR-80-6).

HSIAO, D. K.; MENON, M. J. Parallel Record-Sorting Methods for Hardware Realization. July 1980. 42 pp. (OSU-CISRC-TR-80-7).

APPENDIX G

ACTIVITIES OF THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE STAFF

B. Chandrasekaran presented "An Approach to Medical Diagnosis Based on Conceptual Structures" at the VI International Joint Conference on Artificial Laboratory, Tokyo, Japan, August 20-24, 1979. Co-authors are F. Gomez, S. Mittal, and J. Smith.

B. Chandrasekaran presented "Conceptual Structures for Knowledge Representation and Distribution" at the IEEE Systems, Man & Cybernetics Society Conference on Cybernetics and Society, Denver, Colorado, October 8, 1979. Co-author was F. Gomez. Dr. Chandrasekaran was also the invited organizer and Chairman of the session on "Natural and Social Systems as Metaphors for Distributed Processing" at the same conference.

B. E. Flinchbaugh presented an invited discussion of his current research on "Implications of Temporality for Computer Vision" at the M.I.T. Artificial Intelligence Laboratory, Cambridge, MA, December 12, 1979.

C. R. Foulk presented "Information Flow and Analysis" at the 1980 ACM Computer Science Conference in Kansas City, Missouri, February 13, 1980. Co-author is M. C. Yovits.

A. W. Haley, Jr. presented "Integration of the Independently Tested Modules into a Path Oriented Testing Strategy" at the 1980 ACM Computer Science Conference in Kansas City, Missouri, February 12, 1980.

A. W. Haley, P. Maurath, S. Zeil (graduate students in Computer and Information Science) and E. Swarthout (a senior Electrical Engineering major) representing the OSU Chapter of the Association for Computing Machinery (ACM) placed third (of 23 teams) in the National Scholastic Programming Contest held at the 1980 Computer Science Conference in Kansas City, Missouri, Prior to the contest, the team placed first in the ACM East-Central Regional Programming Contest.

D. K. Hsiao presented a one day seminar on data security and database computers at Information Technology, Inc. in Washington, D.C., July 25, 1979.

D. K. Hsiao delivered an invited talk on database computers at the University of Alberta, Canada, on August 20, 1979. He was also invited to serve as an External Ph.D. Examiner for a Ph.D. examination held at the Department of Computer Science.

D. K.Hsiao presented "Database Computers" in the Database Seminar sponsored by IBM and Sogasta, Ubino, Italy, August 27 - September 3, 1979.

D. K. Hsiao presented "Relational Database Management Systems", Continuing Education, The Ohio State University, Columbus, Ohio, September 20 & 21, 1979.

D. K. Hsiao presented "Recent Advances in Database Computer Technology", Tutorial Session, The 5th International Conference on Very Large Data Bases, Rio de Janerio, Brasil, September 28, 1979.

D. K. Hsiao presented "Database Computer Research" at the following locations:

The Prime Computer Company, October 12, 1979.

Joint EE and CIS Colloquim, University of Maryland, College Park, Maryland, October 18-19, 1979.

Digital Equipment Corporation, Maynard, Maryland, October 26, 1979.

University of Paris, Paris, France, February 19, 1980.

IRIA, Le Chesnay, France, February 20, 1980.

IBM Santa Teresa Lab., San Jose, California, March 21, 1980.

Database Management Group, Digital Equipment Corporation, Merrimack, N. H., June 6, 1980.

D. K. Hsiao presented "Computer Security", Continuing Education, The Ohio State University, Columbus, Ohio, November 15 and 16, 1979 with Professor D. S. Kerr.

D. K. Hsiao presented "Database Machines", Continuing Education, The Ohio State University, Columbus, Ohio, December 17-18, 1979.

D. K. Hsiao presented "Database Machines Are Coming" at the following locations:

IEEE Distinguished Visitors Series, Wright State University, Dayton, Ohio, December 27, 1979.

University of Tronheim, Tronheim, Norway, February 14, 1980.

Professional Seminar, National Computer Conference, Anaheim, CA, May 22, 1980.

D. K. Hsiao presented "Database Computers", the Executive Office of the President, Washington, D.C., January 31, 1980.

D. K. Hsiao presented "Prototyping a Database Computer, DBC", the Chr. Michelsens Institute, Bergen, Norway, February 13, 1980.

D. K. Hsiao presented "Access Control Function of DBMS:, NBS, Bethesda, Maryland, February 28, 1980.

D. K. Hsiao presented "DBC-A Database Computer for Very Large Databases", Bell Laboratories, Murray Hill, N. J., March 25, 1980.

D. K. Hsiao presented "Database Computer and Security Research" at the following locations:

Institute of Mathematics and Institute of Computing Technology, Academia Sinica, Peking, China, April 16-18, 1980.

Hwazong Institute of Technology, Wuhan, China, April 21-25, 1980.

D. K. Hsiao presented "DBC - A Database Computer for Very Large Database Management", IBM Scientific Center, Cambridge, MA, June 10, 1980.

M. T. Liu presented an invited talk on "System Design of the Distributed Double-Loop Computer Network (DDLCN)" to the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, July 10. He also presented the same talk to the Bell Laboratories, Holmdel, New Jersey, July 11, 1980.

M. T. Liu presented two papers entitled "Parallel Processing of High-Level Language Programs", and "A Generalized Cluster Structure for Large Multi-Microcomputer Systems" at the 1979 International Conference on Parallel Processing, Bellaire, Michigan, August 21-24, 1979. The papers have been published in the Conference Proceedings, pp. 17-25, and pp. 74-75, respectively. Co-authors are P. S. Wang and S. B. Wu, respectively.

M. T. Liu presented a paper entitled "System Design of the Distributed Double-Loop Computer Network (DDLCN) at the First International Conference on Distributed Computing Systems, Huntsville, Alabama, October 1-5, 1979. The paper was published in the Conference Proceedings, pp. 95-105. Co-authors are R. Pardo, D. Tsay, J. J. Wolf, B. W. Weide, and C. Chou. Dr. Liu also chaired a session on "Distributed Data Bases Processing and Control" at the Conference.

M. T. Liu presented an invited talk on "Distributed Double-Loop Computer Networks" at the Ford Motor Company Symposium on Distributed Computer Networks, Dearborn, Michigan, October 18, 1979. He was also a member of a panel discussion session on "Local Computer Networking" at the Symposium.

M. T. Liu and B. W. Weide presented "Analysis and Simulation of the Distributed Double-Loop Computer Network (DDLCN)" at the 1979 Computer Networking Symposium, Gaithersburg, MD, December 12, 1979. The paper was co-authored by J. J. Wolf and was published in the conference Proceedings, pp. 82-89.

M. T. Liu presented "Multi-Destination Protocols for Distributed Systems", at the 1979 Computer Networking Symposium, Gaithersburg, MD, December 12, 1979. The paper was co-authored by R. Pardo and was published in the conference Proceedings, pp. 176-185.

M. T. Liu presented an invited talk entitled "Distributed Processing and Local Networking" at the University of Pittsburgh, Pittsburgh, PA, on Feb. 28, 1980.

M. T. Liu presented "Optimal Interconnection Design for Large Multi-Microcomputer Systems" at the Workshop on Interconnection Networks for Parallel and Distributed Processing, Purdue University, April 21-22, 1980. The paper appeared in the conference Proceedings, pp. 101-102, and was co-authored by S. B. Wu.

M. T. Liu presented "The Transmission Grammar Model for Protocol Construction" at the NBS/IEEE Trends and Application 1980: Computer Network Protocols Conference at Gaithersburg, Maryland, May 29, 1980. The paper was published in the conference Proceedings, pp. 110-120. Co-author is A. Y. Teng.

S. A. Mamrak chaired a session on "The Feasibility of Measurement in Interactive Computer Service Selection" at the National Computer Conference, New York City, June 1979.

S. Mittal presented "An Overview of MDX - A Medical Diagnosis System" at the IEEE Conference on Computer Applications in Medicine, Silver Spring, MD, October 15, 1979. The full paper appears in the Proceedings of the Conference. Co-authors were B. Chandrasekaran and J. Smith.

S. Mittal presented an invited talk, "Design of a Distributed Medical Diagnosis System", at the Computer Science Lab, Texas Instruments, Dallas, Dec. 5, 1979.

D. Moore presented "Network Communications via High Level Objects" at the 1980 ACM Computer Science Conference in Kansas City, Missouri, February 12, 1980.

J. Rothstein and A. Davis presented "Parallel Recognition of Parabolic and Conic Patterns by Bus Automata", at the 1979 International Conference on Parallel Processing in Bellaire, Michigan, August 21-24, 1979.

J. Rothstein attended the Statistical Mechanics Meeting at Rutgers University, Brunswick, NJ, on December 13-14, 1979. He presented two papers: 1) The Jacobian in Continuous Information and Continuous Entropy, and its Connection with a Group of Thermodynamically Irreversible Transformations", and 2) A New Speculative Foundation for the Identity of Cosmological and Thermodynamical Arrows of Time".

B. W. Weide presented "Very Fast Information Update and Retrieval Using Cells" at the 17th Annual Allerton Conference on Communication, Control, and Computing, University of Illinois, Monticello, Illinois, Oct. 10-12, 1979.

L. J. White presented an invited colloquium entitled "New Research Areas in Computer and Information Science" to the Department of Electrical Engineering, Ohio University, Athens, on April 1, 1980.

L. J. White participated in a Panel Discussion entitled "Data Processing Education in the Central Ohio Area" sponsored by the Central Ohio Chapter of the Association for Computing Machinery (COCACM), held at BancOhio, Columbus, May 7, 1980.

S. B. Wu presented "Interconnection Design and Resource Assignment for Large Multi-microcomputer Systems" at the 1980 ACM Computer Science Conference in Kansas City, Missouri, February 13, 1980.

S. J. Zeil and L. J. White presented "Sufficiency of Path-Oriented Predicate Tests" at the 1980 ACM Computer Science Conference in Kansas City, Missouri, February 12, 1980.

S. H. Zweben presented "Design Methodologies in Software Engineering" as part of the Distinguished Seminar Series at OCLC, Inc., Columbus, August 14, 1979.

S. H. Zweben was a co-leader of the Regional Chapters Workshop of the Association for Computing Machinery, Hartford, Connecticut, September 15, 1979.

S. H. Zweben was workshop leader at the Chapters Workshop held in conjunction with the 1979 Annual Conference of the Association for Computing Machinery, Detroit, Oct. 28, 1979.

S. H. Zweben presented "An Approach to Computer Program Testing" to the Akron Chapter of the Association for Computing Machinery, Akron, Ohio, on December 12, and to the California State College (Pennsylvania) Student Chapter of the Association for Computing Machinery, California, PA, on December 13, 1979.

S. H. Zweben presented "Measuring Software Development" to the Westchester-Fairfield Chapter of the Association for Computing Machinery, Port Chester, NY, December 18, 1979. Dr. Zweben also presented "An Approach to Computer Program Testing" to the Atlanta Chapter of the Association for Computing Machinery, Atlanta, GA, on January 16, 1980.

S. H. Zweben presented "An Approach to Computer Program Testing" to the Ohio State University Student Chapter of the Association for Computing Machinery (ACM), Columbus, on April 23, 1980.

S. H. Zweben presented "Principles of Software Development" at a seminar on Microprocessors: The Mighty Midgets, sponsored by the Office of Continuing Education, The Ohio State University, May 13, 1980.

S. H. Zweben was an invited panelist on "Panel on Chapter Continuity" at the Association for Computing Machinery Chapters Workshop, held in conjunction with the 1980 National Computer Conference, Anaheim, California, May 18, 1980.

S. H. Zweben presented "Analyzing Software Quality" at a seminar sponsored by the Association for Computing Machinery Central Ohio Chapter, Columbus, June 3, 1980.

S. H. Zweben was an invited participant at the Association for Computing Machinery Southeast Region Chapters' Workshop, Atlanta, Georgia, June 7, 1980.

S. H. Zweben presented "An Approach to Computer Program Testing" at the Central Ohio Chapter of the Association for Computing Machinery, Columbus, June 11, 1980.

S. H. Zweben has been selected for the 1979-80 Lectureship Program of the Association for Computing Machinery (ACM).

S. H. Zweben has been elected to the Committee on Chapters of the Association for Computing Machinery for a three-year term.

S. H. Zweben received a Recognition of Service Award from the Association for Computing Machinery (ACM) at the ACM Central Ohio Chapter's June 1980 meeting in Columbus.

APPENDIX H

DOCTORATES AWARDED

1971-72

CAMERON, JAMES S. Automatic Document Pseudoclassification and Retrieval by Word Frequency Techniques

EKONG, VICTOR J. Rate of Convergence of Hermité Interpolation Based on the Roots of Certain Jacobi Polynomials

GORDON, ROBERT The Organization and Control of a Slave Memory Hierarchy

LANDRY, B. CLOVIS A Theory of Indexing: Indexing Theory as a Model for Information Storage and Retrieval

1972-73

DEFANTI, THOMAS A. The Graphics Symbiosis System - an Interactive Mini-Computer Animation Graphics Language Designed for Habitability and Extensibility

GELPERIN, DAVID H. Clause Deletion in Resolution Theorem Proving

HARRIS, DAVID R. GOLDA: A Graphical On-Line System for Data Analysis

LAY, W. MICHAEL The Double-KWIC Coordinate Indexing Technique: Theory, Design, and Implementation

MATHIS, BETTY ANN Techniques for the Evaluation and Improvement of Computer-Produced Abstracts

WEIMAN, CARL F. R. Pattern Recognition by Retina-Like Devices

WHITTEMORE, BRUCE J. A Generalized Decision Model for the Analysis of Information

YOUNG, CAROL E. Development of Language Analysis Procedures with Application to Automatic Indexing

1973-74

CHAN, PAUL SUI-YUEN An Investigation of Symmetric Radix for Computer Arithmetic

GILLENSON, MARK L. The Interactive Generation of Facial Images on a CRT Using a Heuristic Strategy

HEPLER, STEPHEN PHILIP Use of Probabilistic Automata as Models of Human Performance

WANG, PAUL TIING RENN Bandwidth Minimization, Reducibility Decomposition, and Triangulation of Sparse Matrices

1974-75

BEUG, JAMES L. Human Extrapolation of Strings Generated by Ordered Cyclic Finite State Grammars

DOHERTY, MICHAEL E. A Heuristic for Minimum Set Covers Using Plausability Ordered Searches

FOURNIER, SERGE The Architecture of a Grammar-Programmable High-Level Language Machine

LONGE, OLUWUMI An Index of Smoothness for Computer Program Flowgraphs

MCCAULEY, EDWIN JOHN A Model for Data Secure Systems

PETRY, FREDERICK E. Program Inference from Example Computations Represented by Memory Snapshot Traces

SU, HUI-YANG Pagination of Programs for Virtual Memory Systems

1975-76

BAUM, RICHARD I. The Architectural Design of a Secure Data Base Management System

DASARATHY, BALAKRISHNAN Some Maximum, Location and Pattern Separation Problems: Theory and Algorithms

HARTSON, H. REX Languages for Specifying Protection Requirements in Data Base Systems - A Semantic Model

JUELICH, OTTO C. Compilation of Sequential Programs for Parallel Execution

KALMEY, DONALD L. Comparative Studies Towards the Performance Evaluation of Software for Solving Systems for Nonlinear Equations

KAR, GAUTAM A Distance Measure for Automatic Sequential Document Classification System

MOSHELL, JACK MICHAEL Parallel Recognition of Formal Languages by Cellular Automata

MUFTIC, SEAD Design and Operations of a Secure Computer System

PYSTER, ARTHUR B. Formal Translation of Phrase-Structured Languages

REAMES, CECIL C. System Design of the Distributed Loop Computer Network

RUSSO, PHILLIP M. Cellular Networks and Algorithms for Parallel Processing of Non-numeric Data Encountered in Information Storage and Retrieval Applications

SANTHANAM, VISWANATHAN Prefix Encoding with Arbitrary Cost Code Symbols

SRIHARI, SARGUR N. Comparative Evaluation of Stored-Pattern Classifiers for Radar Aircraft Identification

1976-77

CHENG, TU-TING Design Consideration for Distributed Data Bases in Computer Networks

GUDES, EHUD An Application of Cryptography to Data Base Security

ISAACS, DOV Computer Operating System Facilities for the Automatic Control and Activity Scheduling of Computer-Based Management Systems

KRISHNASWAMY, RAMACHANDRAN Methodology and Generation of Language Translators

LEGGETT, ERNEST W., JR. Tools and Techniques for Classifying NP-Hard Problems

1977-78

BABIC, GOJKO Performance Analysis of the Distributed Loop Network

CHANDLER, JOHN S. A Multi-Stage Multi-Criteria Approach to Information System Design

COHEN, DAVID Design of Event Driven Protection Mechanisms

COHEN, EDWARD I. A Finite Domain-Testing Strategy for Computer Program Testing

KANNON, KRISHNAMURTHI The Design and Performance of a Database Computer

LAKSHMANAN, K. B. Decision Making with Finite Memory Devices

MARIK, DELORES A. Grammatical Inference of Regular and Context-Free Language

PARENT, RICHARD E. Computer Graphics Sculptors' Studio - An Approach to Three-Dimensional Data Generation

1978-79

AMER, PAUL D. Experimental Design for Computer Comparison and Selection

BANERJEE, JAYANTA Performance Analysis and Design Methodology for Implementing Database Systems on New Database Machines

BROWNSMITH, JOSEPH D. A Methodology for the Performance Evaluation of Database Systems

DICKEY, FREDERICK J. Translations Between Programming Languages

LEE, MARY JANE An Analysis and Evaluation of Structure Decision Systems

NATARAJAN, K. S. A Graph-Theoretic Approach to Optimal File Allocation in Distributed Computer Networks

WANG, JIN-TUU Design of a Mixed Voice/Data Transmission System for Computer Communication

1979-80

BAKER, ALBERT L. Software Science and Program Complexity Measures.

FLINCHBAUGH, BRUCE E. A Computational Theory of Spatio-Temporal Aggregation for Visual Analysis of Objects in Dynamic Environments.

'APPINEN, HARRY. A Perception-Based Developmental Skill Acquisition System.

KO, KER-I. Computational Complexity of Real Functions and Polynomial Time Approximation.

KWASNY, STAN C. Treatment of U grammatical and Extra-Grammatical Phenomena in Natural Language Understanding Systems.

MELLBY, JOHN ROLF. The Recognition of Straight Line Patterns by Bus Automatons Using Parallel Processing.

PARDO, ROBERTO. Interprocess Communication and Synchronization.

TENG, ALBERT Y. Protocol Constructions for Communication Networks.

WOLF, JACOB J., III. Design and Analysis of the Distributed Double-Loop Computer Network (DDLCN).

WONG, PATRICK M. K. A Methodology for the Definition of Data Base Workloads: An Extension to the IPSS Methodology.