2

Massachusetts Institute of Technology

Laboratory for Computer Science Progress Report 27

July 1989-
June 1990

The work reported herein was carried out by the Laboratory for Computer Science, an MIT interdepartmental research laboratory. During 1989-90, our principal financial support has come from the Defense Advanced Research Projects Agency (DARPA). DARPA has been instrumental in supporting most of our research during the past 27 years and is gratefully acknowledged here. Our overall support has come from the following organizations:

- Defense Advanced Research Projects Agency;
- National Institutes of Health;
- National Science Foundation;
- Office of Naval Research; and
- MIT controlled IBM funds under the IBM/MIT joint study contract.

Other support of a generally smaller level has come from Apple, AT&T, Bellcore, Corporation for National Research Initiatives, Digital Equipment, Nynex, Siemens, United States Army Research Office, Xerox, and the multiple industrial members of the MIT X Consortium.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MIT/LCS/PR - 27	5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION MIT Lab for Computer Science	6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research/Dept. of Navy
6c. ADDRESS *(City, State, and ZIP Code)* 545 Technology Square Cambridge, MA 02139		7b. ADDRESS *(City, State, and ZIP Code)* Information Systems Program Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA/DOD	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State, and ZIP Code)* 1400 Wilson Blvd. Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
MIT Laboratory for Computer Science Progress Report - 27

12. PERSONAL AUTHOR(S)
M.L. Dertouzos

13a. TYPE OF REPORT Technical /Progress	13b. TIME COVERED FROM 7/89 TO 6/90	14. DATE OF REPORT *(Year, Month, Day)* June 1990	15. PAGE COUNT 306

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Artificial Intelligence, Computer Architecture, Computer Science, Electrical Engineering, Network, Theory of Computers, Programming Languages

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*
Annual Report of Progress made at the MIT Labortory for Computer Science Order Contract: N00014-89J-1988.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Carol Nicolora	22b. TELEPHONE *(Include Area Code)* (617) 253-5894	22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
*U.S. Government Printing Office: 1985—507-047
Unclassified

Massachusetts Institute of Technology
Laboratory for Computer Science
545 Technology Square
Cambridge, Massachusetts 02139
617-253-5851

Progress Report

27

Contents

Introduction

ADMINISTRATION

Academic Staff

M. Dertouzos, Director
R. Rivest, Associate Director
A. Vezza, Associate Director

Administrative Staff

G. Brown, Facilities Officer
A. Djazani, Sr. Staff Accountant
W. Fitzgerald, Fiscal Officer
A. Horan, Administrative Assistant
M. Mitchell, Administrative Officer
D. Ruble, Staff Accountant
M. Sensale, Librarian
B. Tilson, Staff Accountant
P. Vancini, Personnel Officer

Support Staff

K. Adamson	R. Purvis
G. Brown	M. Rebelo
L. Cavallaro	C. Robinson
S. Costanza	D. Santoro
R. Donahue	M. Siddique
C. Ehling	R. Soble
J. Little	L. Wenger
P. Mickevich	L. Worley
C. Nicolora	M. Yohe
J. Porter	

1.1 Overview

The MIT Laboratory for Computer Science (LCS) is an interdepartmental laboratory whose principal goal is research in computer science and engineering.

Founded as Project MAC in 1963, the Laboratory developed one of the world's earliest timeshared computer systems. This early research on the Compatible Time Sharing System (CTSS) and its successor, MULTICS, made possible innovative developments such as the writing of operating systems in high level programming languages, virtual memory, tree directories, online scheduling algorithms, line and page editors, secure operating systems, concepts and techniques for access control, computer-aided design, and two of the earliest computer games—space wars and computer chess.

These early developments laid the foundation for the Laboratory's work in the 1970's on knowledge-based systems—for example, the MACSYMA program for symbolic mathematics—natural language understanding, and (with BBN) the development and use of packet networks. During this same period, the Laboratory developed theoretical results in complexity theory and linked cryptography to computer science through concepts and algorithms for public encryption (RSA). In the late 1970's, Project MAC, renamed as the Laboratory for Computer Science (LCS), embarked on research in clinical decision making, the exploration of cellular automata at the borderline between physics and computation, and on the social impact of computers. At the same time, it began two major research programs in distributed systems and languages, and in parallel systems. These led to the notion of data abstractions and the Clu language, the Argus distributed system, the dataflow principle and associated languages and architectures of parallel systems, local area ring networks, program specification, and workstation development, where the Laboratory contributed the earliest UNIX ports and compilers, and the NuBus architecture, now used in commercial computers like Apple's Macintosh II. This research has also led to the X Window System, a computer intercommunication standard, developed together with Project Athena.

The Laboratory's current research falls into four principal categories: Parallel Systems; Systems, Languages, and Networks; Intelligent Systems; and Theory. The principal technical goals and expected consequences in each of these four categories are as follows:

In *Parallel Systems*, we strive to harness the power and economy of numerous processors working on the same task. Research in the area involves the analysis and construction of various hardware architectures and programming languages that yield, over a broad set of applications, cost-performance improvements of several orders of magnitude relative to single processors. This research is expected to affect most of tomorrow's machines which we expect to be of the multiprocessor variety—not only because of potential cost-performance benefits, but also because of the natural, yet unexploited, concurrence that characterizes contemporary and prospective applications from business to sensory computing.

In *Systems, Languages, and Networks*, our objective is to provide the concepts, methods, and environments that will enable heterogeneous computers, each working on different tasks, to communicate efficiently, conveniently, and reliably with each other in order to exchange information needed and supplied by their respective programs. Such communication may involve, beyond conventional electronic mail and file transfer, the calling of programs in one environment from programs in another, perhaps different, environment; the storage and sharing of structured data among such programs; and the use of an information infrastructure consisting of common computer and communication resources. This research is also expected to have a broad impact on future systems because virtually every machine will be connected to some network.

Taken together, these two thrusts in parallel and networked machines signal our expectation that future computer systems will consist of multiprocessors interconnected by local and long haul networks, and perhaps some day by national network infrastructures as ubiquitous and as important as today's telephone and highway infrastructures.

In the *Intelligent Systems* area, our technical goals are to understand and construct programs and machines that have greater and more useful sensory and cognitive capabilities. Examples include the understanding of spoken messages, systems that can learn from practice rather than by being explicitly programmed, and programs that reason about clinical issues and help in clinical decision making. We expect tomorrow's intelligent systems to be easier to use than today's programs across a broad front of applications.

In our fourth category of research, *Theory*, we strive to understand and discover the fundamental forces, rules, and limits of computer science. Theoretical work permeates many of our research efforts in the other three areas, for example, in the pursuit of parallel algorithms and in the study of fundamental properties of idealized parallel architectures and computer networks. Theory also touches on several predominantly abstract areas, like the logic of programs, the inherent complexity of computations, and the use of cryptography and randomness to the formal characterization of knowledge. The impact of theoretical computer science upon our world is expected to continue its past record of improving our understanding of and helping us to pursue new frontiers with new models, concepts, methods, and algorithms.

1.2 Highlights of the Year

Research highlights during the reporting period were as follows:

1. Through the Laboratory's Spoken Language Systems Group, we began exploration of an international interpretive telephony effort. Users of this telephone would speak in their native tongue using a limited vocabulary of a few hundred words in a narrow domain of discourse, as in for example, appointments, visits, and travel plans that lead to meetings. Each sentence would be translated through an intermediate language (I.L.) to the language of the other party. It would also be simultaneously translated back from I.L. to the original language to ensure the system "understood" what was said. To date, we have secured informal partnerships in Europe and Japan for the purpose of carrying out this research.

2. Professor David Tennenhouse and his associates began research on computer workstations that will deal with video images, just as today's workstations handle text. Novel processing-on-the-fly methods are being explored in this area, in addition to the more traditional retrieve-process-and-store techniques. Visual images are likely to be used increasingly because people are becoming more used to them and because they can cut across linguistic barriers.

3. Professor Stephen Ward completed a prototype of the NuMesh—a "Tinkertoy" system that enables special purpose computers to be built out of general purpose, small size blocks. The resultant machines are expected to carry out special purpose processing at very high speeds.

The Laboratory's Distinguished Lecturer Series included presentations by John Hennessy, Bell Professor of Electrical Engineering and Computer Science, Stanford University; Terrence J. Sejnowski, Director, Computational Neurobiology Laboratory, Salk Institute and Professor of Biology and Physics, University of California, San Diego; Ronald L. Graham, Adjunct Director, Research, Information Sciences Division, AT&T Bell Laboratories; Robert M. Metcalfe, Ethernet Inventor and 3Com Corporation Founder; and Gordon Plotkin, Professor of Computer Science, University of Edinburgh.

Professors Leo Guibas and Mauricio Karchmer both joined the Laboratory as members of the Theory of Computation Group and Messrs. Joseph Polifroni of the Spoken Language Systems Group and Kenneth Streeter of the Information Mechanics Group became members of the research staff. Changes in the administrative staff included the departure of Mr. William Fitzgerald, who was replaced as Fiscal Officer by Ms. Azadeh Djazani, and the assignment of Ms. Carol Robinson to Personnel Officer.

The Laboratory is organized into 15 research groups, an administrative unit, and a computer service support unit. The Laboratory's membership includes a total of 400 people—110 faculty and research staff, 40 visitors, affiliates, and postdoctoral associates, 35 support staff, 160 graduate students, and 55 undergraduate students. The academic affiliation of most of the faculty and students is with the Department of Electrical Engineering and Computer Science (EECS).

The Laboratory's funding comes predominantly from the U.S. Government's Defense Advanced Research Projects Agency, which accounts for about half of the total. In addition, we are funded by and have extensive links with industrial organizations. These include partnerships for the construction of major hardware systems, consortia for the development and maintenance of standards, like the X Window system, and joint studies on research areas of common concern. Technical results of our research in 1989-90 were disseminated through publications in the technical literature, through Technical Reports, numbered 454 through 479, and Technical Memoranda, numbered 401 through 432.

Advanced Network Architecture

Academic Staff

D. Clark, Group Leader	D. Tennenhouse
J. Saltzer	

Research Staff

J. Davin	K. Sollins

Graduate Students

A. Buzacott	R. Hirschfeld
M. Frumkin	T. Shepard
E. Hashem	G. Troxel
A. Heybey	L. Zhang

Undergraduate Students

J. Bonsen	D. Martin
J. Coburn	T. Mendez
B. Gaunce	T. Ts'o
M. Lee	

Support Staff

A. Aldrich	K. J. Madera

Visitors

R. Jain	L. Wu
H. Rebstock	

2.1 Introduction

The goal of the Advanced Network Architecture project continues to be the definition of a protocol architecture that will achieve application data transfer at a gigabit or more while meeting other requirements for quality of service and media independence. The central problem of our group has been the management of bandwidth, switching capacity, and buffer resources within the network. If we are to achieve higher speeds and larger size, the tradeoffs among these resources must change, and new algorithms and approaches will be needed.

In the following sections, a number of specific projects related to this overall goal are described.

2.2 Rate-based Flow Control

Previously, we proposed *rate-based* flow control as a key concept for resource management in tomorrow's networks. Zhang's thesis [300] proposed a specific control scheme for rate-based control, including the resource allocation algorithm in the gateway and a matching control scheme in the host. The concept of a *virtual clock* to meter the traffic in the various flows is central to the scheme. Extensive simulation indicates that the scheme has great promise.

An analysis performed by Liang Wu (on sabbatical from Bellcore) related the Zhang virtual clock scheme to another scheme, the so-called *leaky bucket* scheme being proposed for resource management in telephone networks.

Helmut Rebstock, a visiting scientist from Siemens Corporation, and James Davin simulated the behavior of a novel rate adjustment algorithm proposed by David Tennenhouse. By this algorithm, packets are always generated at the maximal desired rate, but only a certain portion of the generated packets represent useful data: the rest are "empty" packets. The receiver periodically sends control information to the transmitter about the rate at which packets are actually received, and the transmitter adjusts the relative rate of empty packet transmission accordingly. Simulation showed that this algorithm successfully copes with congestion in simple networks, but it does not extend well to more complicated topologies. Tennenhouse subsequently proposed an extension of the empty packet scheme by which a congested packet switch discards empty packets first. This augmented scheme has not been simulated to date.

2.3 Protocol Performance Studies

In order to explore the contrast between rate-based and window-based flow control mechanisms, Rajiv Jain, a visiting scientist from the ITT in India, undertook to study the behavior of TCP in the presence of high bandwidth, long delay links. Jain simulated the slow-start TCP algorithm over gigabit links with transcontinental delays. Initial results suggested that although TCP afforded good link utilization in a benign environment, it degraded significantly in the presence of even modest packet loss. More conclusive results were not pursued owing to the significant effort required for each experiment.

Another approach to studying TCP behavior was continued by Timothy Shepard. Previously, a system for collecting and storing about 12 hours of the protocol headers of all the packets on one of the main Ethernets in the Laboratory had been built and is now in continuous operation. The system is used as a network debugging aid and as a source of traces to support research in the analysis of TCP packet traces.

Examination of a trace of packets collected from the network is often the only method available for diagnosing protocol performance problems in computer networks. Shepard's thesis [269] explores the use of packet traces to diagnose performance problems of the transport protocol TCP. Because manual examination of these traces can be so tedious as to preclude detailed analysis, a more effective method is developed: the primary contribution of this thesis is a graphical method for displaying the packet trace which greatly reduces the tedium of its analysis.

This graphical method is demonstrated by the examination of packet traces from typical TCP connections. The performance of two different implementations of TCP sending data across a particular network path is compared. Traces, many thousands of packets long, are used to demonstrate how effectively the graphical method simplifies examination of long, complicated traces. Because the burstiness of TCP transmitters observed in packet traces seems occasionally related to their achieved throughput, a method of quantifying this burstiness is presented and its possible relevance to understanding the performance of TCP is discussed.

To facilitate study of collected packet traces, Shepard developed an interactive, X-based tool that displays the detailed behavior of a TCP connection according to the graphical method he describes. This tool, which shows the timing relationship of the various events in the protocol transaction, permits a sophisticated analyst to diagnose and debug TCP problems at high speed. This tool has been used with great success on local MIT networks and on the ARPANET to examine a variety of performance and functional problems.

2.4 Fair Queueing in Gateways

James Davin and Andrew Heybey continued their exploration of the "Fair Share" queueing algorithm developed at Xerox PARC [89]. A series of simulations demonstrated that, in a connectionless network, a gateway which allocates outgoing link bandwidth according to a *fair queueing* algorithm can effectively regulate and share the use of a trunk, as, for example, among several government agencies jointly procuring and using a link. In its simplest form, the algorithm enforces fairness—no user may use more than its fair share of the output bandwidth. It was shown to be effective in enforcing non-uniform policies by which some users are accorded a larger share of the bandwidth than others.

Three fair queueing algorithms were evaluated by simulation: the algorithm originally described in [89], a known (but previously unexplored) variation on that algorithm that is more accommodating to bursty sources, and a novel variation on the algorithm that is distinguished by a simplified buffer management scheme. These three algorithms were evaluated for their capacity to enforce uniform and non-uniform policies in the face of network demands that ranged from light to heavy, static to dynamic, cooperative to non-conformant. The considered algorithms afforded effective policy enforcement in a variety of circumstances for which traditional first-come-first-served gateway policies failed to do so. As might be expected, none of the considered algorithms were able to correct for throughput disparities that arise from closed-loop flow control mechanisms in networks of heterogeneous delays. A paper describing this work is in preparation.

2.5 Random Drop Queue Management

An algorithm often called *random drop* has been proposed by various workers in the field as a simpler alternative to fair queueing for controlled allocation in gateways. Eman Hashem completed her study of random drop and other congestion-related phenomena [141]. Based on simulation experiments, she concludes that random drop can compensate for certain peculiar meta-stable conditions leading to unfair allocation but that random drop cannot compensate for important and fundamental causes of unfairness, such as different roundtrip times for connections across the network. Hashem also investigated a variant of the random drop scheme—*early random drop*—that aspires to congestion avoidance rather than congestion recovery. She finds that the success of early random drop is problematic insofar as it depends upon the development of effective algorithms for dynamic adjustment of the drop rate.

2.6 Next Generation Protocol Architecture

An overall research objective of the group is to synthesize, out of specific study areas, an overall protocol architecture for the networks of tomorrow.

To this end, David Tennenhouse has considered the relationship between popular network design strategies and the performance requirements of future network applications. The ATM (Asynchronous Transfer Mode) approach to broadband networking is presently being pursued within the CCITT (and elsewhere) as the unifying mechanism for the support of service integration, rate adaptation, and jitter control within the lower layers of the network architecture. Tennenhouse prepared a paper [277] concerned with the jitter (variation in delay) arising from the design of the middle and upper layers that operate within the end systems and relays of multi-service networks.

In order to augment the ongoing discussion of ATM, Tennenhouse organized the ATM Practitioner's Workshop, held January 22-24 at the MIT Endicott House Conference Center. This conference brought together over 30 individuals with a wide variety of perspectives on ATM, and realized an unusual opportunity for discussion between participants in the B-ISDN standards committees and academic researchers in the field. Although the context of the discussion was current ATM standards activity, it focused on reports of recent research results and proposals for future work.

Exploration of a broad range of architectural issues was realized by the group's participation in the Internet Research Steering Group Workshop on Very High Speed Networking, held January 24-26 in Cambridge, MA. David Clark served on the program committee for this conference and chaired a working group session on protocol implementation. David Tennenhouse made a presentation on relevant results from the ATM Practitioner's Workshop.

The group was able to contribute to the evolution of national networking infrastructure by participation in a workshop on the NRI Networking Testbed held in December in Reston, Virginia.

2.7 Research Collaboration

During the current year, the group established a number of key research collaborations that will augment and advance its research agenda. Among the most important is a collaboration with Bellcore by which protocol concepts developed within the group will be demonstrated using a prototype ATM switch under development there. As a part of this effort, members of this group contributed to the design of key parts of that switch—in particular, the port controller at the input and output of the actual switch fabric.

David Tennenhouse served on a committee to discuss possible joint ventures between researchers on the MIT campus and at MIT Lincoln Laboratories to explore all-optical networking architectures.

2.8 Broadband ISDN Standards Activities

The current efforts within ANSI T1S1 to develop the standards for Broadband ISDN will, if successful, define the nature of the U.S. telecommunications infrastructure for the next several decades. Because of the importance of this effort, we attempted to contribute in ways that preserve and enhance the utility of the network for computer interconnect. David Tennenhouse attended meetings of the relevant standards committees.

Alan Buzacott also attended standards meetings and wrote an analysis of the broadband standards process [65]. He concludes that, although the emerging standards may be technically flawed in some details, the broadband ISDN standards process succeeded in introducing fundamental conceptual changes into public networks in a timely and appropriate way.

2.9 Internet Naming Services

During the year, a plan for development and deployment of a "white-pages" naming service for the Internet was prepared. International standards such as X.500 attempt to define such a service and, indeed, should be the basis of an Internet service. However, significant additional effort will be required to realize a practical, standards-based system. An RFC by Karen Sollins [273] outlines a possible program in this area.

As part of this activity, Trevor Mendez wrote an X.500 Directory User Agent, based on the X Window System, that works with the Quipu X.500 server from University College, London. To further simplify the user interface and increase functionality, he also integrated Kerberos authentication into Quipu.

2.10 Network Management

James Davin continued his efforts to develop the Simple Network Monitoring Protocol (SNMP), by participation in the relevant Internet Engineering Task Force working groups, by contributing to documentation of the protocol [68][67], and through trial implementations and document contributions. His current efforts involve the addition of authentication to SNMP [115][86][222].

2.11 Advanced Network Simulator

The interactive network simulator that supports much of the simulation work in the group was further refined and enhanced during this year. New simulator components were crafted to support a variety of experiments. David Martin developed a facility for the graphical display of relative link utilizations on links shared by multiple network users. Andrew Heybey ported the simulator to MIPS M/120 workstation hardware. James Davin and Andrew Heybey ported the simulator to the Cray 2 supercomputer, although the performance of this latter port is less than might otherwise be expected owing to the limited opportunities for vectorization in typical network simulations. A variety of bugs have been fixed, and the improvements have been released to other interested parties, for whom at least a minimal level of support is provided. The simulator is being actively used by people at Washington, Cray, Purdue, and Mitre.

2.12 Hardware Development Tools

Jonathan Coburn developed *Xil*, a digital logic description language embedded in Scheme. Xil allows digital circuits to be defined in terms of boolean logic functions and registers. It outputs configuration information for Xilinx reconfigurable logic arrays, a family of software-programmable gate arrays. Xil's Scheme embedding allows hardware descriptions to be generated algorithmically, while use of the Xilinx LCAs allows complex designs to be rapidly instantiated as single, reusable chips.

Xil is described in [76]. It is currently in use as a hardware development tool in support of other ANA research activities.

2.13 Internet Protocol Committee Participation

Because of the continuing importance of the Internet protocol suite, and because of the potential cross-fertilization between our research goals and the future needs of the Internet, members of the research group continue to participate in Internet working groups. During the year, David Clark resigned as chairman of the Internet Activities Board, a position he had held since 1981. He continues to serve on the IAB, and chairs one of its two subcommittees, the Internet Research Steering Group. Members of the group have attended Internet Engineering Task Force meetings, as well as IETF working groups and ad hoc committees as appropriate, including various meetings to discuss naming in the Internet. Of the various activities of the Internet Research Task Force, Clark contributed to the End-to-End Research Group, and both Clark and Karen Sollins participated in the Autonomous Networks Research Group.

2.14 Publications

[1] J. D. Case, J. R. Davin, M. S. Fedor, and M. L. Schoffstall. Internet Network Management Using the Simple Network Management Protocol. In *Proceedings of the 14th IEEE Conference on Local Computer Networks*, Minneapolis, MN, October 1989.

[2] J. D. Case, M. S. Fedor, M. L. Schoffstall, and J. R. Davin. *A Simple Network Management Protocol.* Request for Comments 1157, DDN Network Information Center, SRI International, May 1990.

[3] D. D. Clark and D. R. Wilson. Evolution of a model for computer integrity. In *Report of the Invitational Workshop on Data Integrity,* January 25–27, 1989, pages A.2.1–A.2.13, September 1989. NIST Special Publication 500-168.

[4] E. Hashem. *Analysis of Random Drop for Gateway Congestion Control.* Technical Report MIT/LCS/TR-465, MIT Laboratory for Computer Science, November 1989.

[5] A. T. Heybey. *Rate-based Congestion Control in Networks with Smart Links.* Technical Report MIT/LCS/TR-470, MIT Laboratory for Computer Science, January 1990.

[6] T. M. A. Lomas, Li Gong, J. H. Saltzer, and R. M. Needham. Reducing risks from poorly chosen keys. In *Proceedings of the 12th ACM Symposium on Operating Systems Principles*, pages 14–18, December 1989.

[7] K R. Sollins. *Plan for Internet Directory Services.* Request for Comments 1107, DDN Network Information Center, SRI International, July 1989.

[8] D. L. Tennenhouse. Layered multiplexing considered harmful. In H. Rudin and R. Williamson, editors, *Protocols for High Speed Networks*, Elsevier Science Publishers, 1989.

[9] D. L. Tennenhouse and I. M. Leslie. A testbed for wide area ATM research. In *Proceedings ACM SIGCOMM*, pages 182–190, Austin, TX, September 1989.

[10] L. Zhang. *A New Architecture for Packet Switching Network Protocols.* Technical Report MIT/LCS/TR-455, MIT Laboratory for Computer Science, August 1989.

Theses Completed

[1] A. Buzacott. *The Development of Broadband Telecommunications Standards.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[2] J. L. Coburn. *Xil: A Scheme Embedded Hardware Description Language for Xilinx Chip Configuration.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[3] E. Hashem. *Analysis of Random Drop for Gateway Congestion Control.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, August 1989.

[4] T. Shepard. *TCP Packet Trace Analysis.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[5] T. Ts'o. *Nox: A Private Key Encryption Server with Flexible Semantics.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[6] L. Zhang. *A New Architecture for Packet Switching Network Protocols.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, August 1989.

Talks

[1] D. Clark. Protocol performance—why networks don't go fast. Lecture given at Interop 89, San Jose, CA, October 1989.

[2] J. Saltzer. Lessons from Project Athena. Lecture given at MIT Club of Northern California, Palo Alto, CA, September 1989.

[3] D. Tennenhouse. Advanced network and workstation architectures. Lecture given MIT ILP Symposium "Telecommunications Technology and Policy for the 21st Century," Cambridge, MA, April 1990.

[4] D. Tennenhouse. Network-friendly video workstations: some architectural proposals. Lecture given at Olivetti Research, Menlo Park, CA (August 1989); DEC SRC, Palo Alto, CA, (July 1989).

[5] D. Tennenhouse. Scalable video architecture: a cross-industry model for high resolution systems. Lecture given (via teleconference) at COHRS-IEEE-NAS Workshop "In Pursuit of An Architecture of Standards for the Interoperation of High Resolution Systems Across Industries," Washington, DC, May 1990.

[6] D. Tennenhouse. The unison ATM testbed. Lecture given at AT&T Bell Laboratories, Murray Hill, NJ, July 1989.

Clinical Decision Making

Academic Staff

R. Patil	P. Szolovits, Group Leader
G. Rennels	

Collaborating Investigators

M. Criscitiello, M.D., Tufts-New England Medical Center Hospital
M. Eckman, M.D., Tufts-New England Medical Center Hospital
J. Kassirer, M.D., Tufts-New England Medical Center Hospital
S. Naimi, M.D., Tufts-New England Medical Center Hospital
S. Pauker, M.D., Tufts-New England Medical Center
W. Schwartz, M.D., Tufts-New England School of Medicine
O. Senyk, M.D. Ph.D., Tufts-New England Medical Center Hospital
F. Sonnenberg, M.D., Tufts-New England Medical Center Hospital
J. Wong, M.D., Tufts-New England Medical Center Hospital

Research Staff

J. Doyle W. Long

Graduate Students

D. Aghassi	N. Harris	T. Russ
D. Fogg	Y. Jang	T. Wu
S. Greenwald	T.-Y. Leong	A. Yeh
I. Haimowitz	R. Rubsamen	

Support Staff

A. Ellis

Visitor

H. Tanaka

3.1 An Artificial Intelligence Approach to Clinical Decision Making

3.1.1 Background and Significance

In the explosion of new knowledge, new methods, new regulations, stringent pressures to reduce costs, and higher expectations from patients for better outcomes, medicine faces a major problem of information management and utilization. During the past decade, many independent studies of this complex of problems have settled on medical informatics as the field promising to help alleviate this problem. From the GPEP report of the early 1980's, to the NLM's planning reports of the mid-1980's, to this year's recommendation at the Harvard Medical School to establish a center for medical informatics and to assure that all medical students are trained in this discipline, the need for more sophisticated computer-based applications in medicine is clearly identified. These are to be applications that in some sense "understand" the content of the information they manipulate.

The idea that one can develop computer programs that assist in making diagnostic and therapeutic decisions, or that track the ongoing state of a patient and comment on the appropriateness of therapy is hardly novel, of course. Flowchart and statistical classification models dating back to the early 1960's have played a small beneficial role in enhancing medical care, and systems of almost equal vintage that rely on a fairly complete online medical record have provided trend analysis and simple "sanity checks" for evolving patient cases. Systems based on online medical records have limited their reasoning to issues that could be adequately addressed by data that were typically available, which in most cases fails to capture much of what is clinically relevant. The scarcity of adequate background data, and the lack of modularity and internal organizational structure in the classification models, has prevented their construction for large medical domains and has seriously impaired the ability of developers to maintain them [264]. In response to these inadequacies, researchers turned in the early 1970's to artificial intelligence methods, to provide tools for building programs with more "understanding." By adopting a consultation model, where the program is to be able to ask questions of its users, the inadequacy of computer-accessible information could be overcome (though of course at a high cost in time demanded of the user). Perhaps more fundamentally, these programs pursued the hypothesis that one could overcome the lack of statistical data by substituting for it the codified expertise of human expert clinicians. This is not to say that one would simply ask people to guess statistical correlations instead of gathering data on them. Instead, the idea was to discover the reasoning and problem-solving strategies used by human experts, to de-brief them of the knowledge they use, *in the form in which they use it*, and then to build computer programs that operate according to the same principles and with the same knowledge.

3.2 State of the Art in Medical AI

Since the early 1970's, the field of medical AI has provided a number of impressive demonstrations of programs able to capture human-like expertise and to apply it in human-like ways that seem acceptable to their users. Systems such as the Present Illness Program [244] and the Digitalis Therapy Advisor [131] from our group, and MYCIN, INTERNIST-I [225], and CASNET/Glaucoma [294], provided early indications that such AI programs could overcome previous methodological limitations to provide human-like expertise in a computer. Indeed, tests of these and successor programs have several times confirmed that, within a typically narrow set of circumstances, the performance of the program was (nearly) indistinguishable from that of expert physicians [298][225][146][34].

Sadly, despite these documented successes, the practical utility of programs of this sort in medicine remains very limited—essentially only a few such programs (PUFF, ONCOCIN and a serum electrophoresis interpretation program built into an instrument by Helena Laboratories) receive any routine use [75], and then typically at medical centers closely associated with their developers. Interestingly, the techniques developed for some of these medical programs have been generalized and exported to commercial and industrial areas of application, where they have formed the basis for a significant "expert systems" revolution [107].

Why have medical AI programs not succeeded practically, when they appear to have succeeded in the laboratory? One hypothesis is that the fundamental technology of these systems is fine, but that much more engineering effort is needed to bring them to successful use. Another, which motivates us here, is that there really are fundamental deficiencies in the techniques on which these systems are based—deficiencies that prevent their functioning as well as is necessary for widespread adoption.

No doubt better engineering *will* be needed for the success of medical expert systems. Thus, work is needed on comprehensive medical record systems that contain a timely, complete and accurate view of the patient and the care he or she is getting. Outstanding user interfaces, which exploit the power of graphical output and voice input would also be a boon. Improved ancillary services to train potential users and more thoroughly integrate computer systems into the fabric of health care are also likely to be needed. In addition, work on standards for medical terminology, large scale knowledge bases organized for teaching and reference, and integration of patient care, research, library, image, and electrical signal databases into a uniformly-accessible information system is an important goal. Nevertheless, we subscribe to the second of the above hypotheses, that even significant advances toward all these laudable goals would leave fundamental gaps in our ability to build truly usable systems.

3.3 Sources of Difficulty

Among the major difficulties identified in building medical reasoning systems are the handling of multiple interacting diseases, interactions between diseases and incomplete or only partially-effective therapies, the need to take time into account in both diagnostic and therapeutic reasoning, and real time constraints on decision making. Each of these has severely stressed the basic mechanisms of even the most successful demonstrated programs, and each suggests that there is much additional need for research.

The need to deal with unanticipated interactions has meant that programs have turned from relatively simple means of associating clusters of abnormal findings with disease hypotheses to much more complex means of assembling hypotheses that represent multiple co-occurring disorders. Often, this has required new causal and probabilistic models of the interactions among disorders and the ways in which disorders manifest as abnormal findings. Thus, the knowledge base of recent medical AI programs is typically quite complex; their reasoning methods are multi-faceted and involved, their conclusions are difficult to explain because of this complexity, and the programs are hard to build, debug and maintain.

Five years ago, we suggested one approach to alleviating some of these problems, based on the adoption of a common underlying knowledge representation mechanism. Unfortunately, though we have shown some progress in this direction, the existing knowledge representation formalisms at our disposal have not been up to the task—the complexity and breadth of types of knowledge that need to be represented in medical reasoning overwhelms the abilities of existing techniques.

In addition, as the medical AI field has matured and the ambitions of specific projects have increased, it appears to take longer and longer for an interesting new idea to move from conception to demonstration in an effective program. The need to hand-tailor medical knowledge bases specific to a particular project, as well as to develop a complex set of technical capabilities, has meant that often five years may elapse from inception of an idea to its initial demonstration. This observation naturally leads to the suggestion that perhaps computer learning techniques could serve to allow the machine to be a more active partner in building new programs. Until recently, we felt that many of the learning methods explored in the AI literature were not likely to be directly applicable to our problems. Such methods fell into two camps: methods based on statistical learning gave no place to hard-earned knowledge that we already possessed, and it seemed implausible to ask automated learning techniques to rediscover all the existing knowledge of medicine. Conversely, most symbolic learning methods assumed a deterministic underlying domain, in which noise or stochastic behavior would lead either to no learning or to internal contradictions.

3.4 Artificial Intelligence and Cardiovascular Reasoning

3.4.1 Diagnosis

The heart-failure diagnosis program provides two types of diagnostic information: it determines the probability of parameter states in the physiologic model, and it generates differential diagnoses each of which fully explains the set of findings.

The model for diagnosis consists of nodes copied from the parameter states with binary values and measurement values (the findings) connected by links with probabilities determined from the knowledge base and patient input. The probabilities are combined using a "noisy-or" combination rule [245] except for worsening factors, which require another cause, and correcting factors, which decrease the probability. Thus, if causes are P, worsening factors W, correcting factors C, and primary probability is p_0, the probability of a node is:

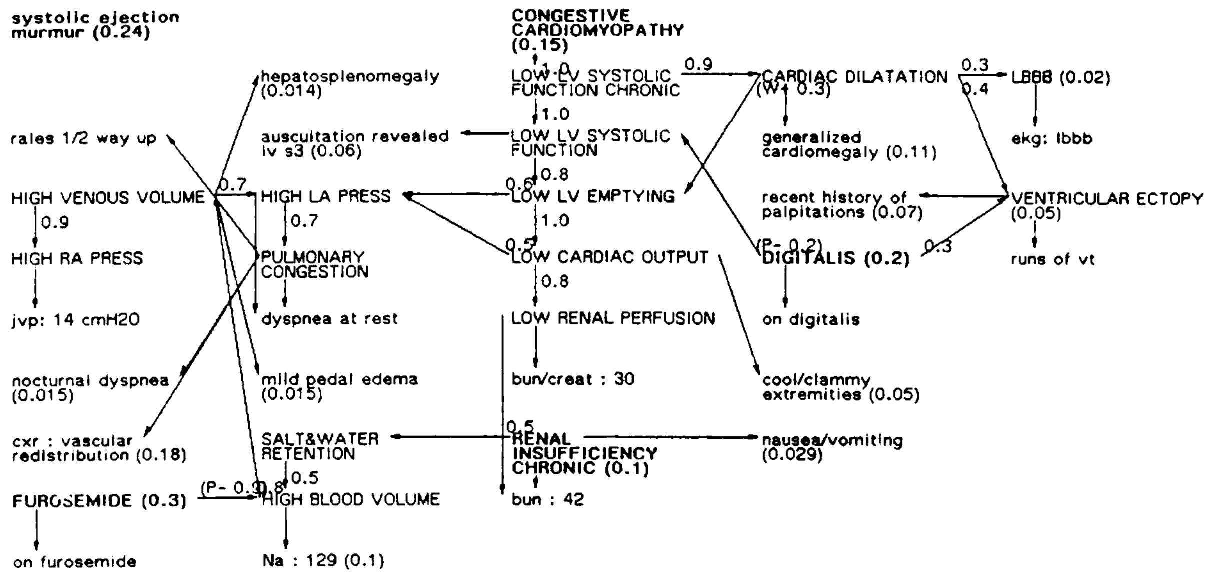
$$\exists i | i \in P \wedge (p_i = 1.0) \Rightarrow 1.0, \exists i | i \in P \Rightarrow (1 - \prod_{i \in P,W} (1 - p_i)) \prod_{i \in C} (1 - p_i), else \Rightarrow p_0$$

Similarly, each measurement value has a probability of being produced by nodes. The model is similar to those investigated by Pearl [245] as Bayesian probability networks. The difference is that this model has forward loops (excluded by Pearl) and nodes with multiple paths between them (handled only in exponential time by Pearl's methods). We investigated modifications to Pearl's algorithm and to our model. However, eliminating the forward loops in an earlier model version there were still about 40 links that would have to be cut to analyze multiple paths between nodes. Thus, Pearl's algorithm would require weighted summing of about 2^{40} solutions, which is completely infeasible. Indeed, Cooper has shown that the problem is NP-hard [79]. Thus heuristic methods are necessary to handle large networks.

After much investigation we developed a mechanism for estimating the probability of a node given evidence. It is based on the causal paths from primary nodes to the node in question. Since only about 50 of the 150 nodes in the model are primary (having a non-zero probability of existing without some other cause), it generates and stores all of the paths from these nodes to all others and computes the probabilities along those paths. The paths are generated when the model is first loaded and the probabilities are computed when the patient data is entered. A conservative approximation of the causal probability of any node is the combination of the highest probability paths from each of the primary nodes to that node, assuming the independence of the causes and the default combination rule. Explicit causal combinations (e.g., the worsening factors) are handled by revising the probabilities along the paths for the estimated probabilities of these additional factors. This mechanism has proven to be an effective way of estimating the causal probabilities of nodes. To determine the probability of a node we treat the evidence evaluation problem as locally computing the probability of the observed effects given each combination of causes. This mechanism estimates the probability of any parameter state given whatever other states or evidence is already known.

Our solution to the differential diagnosis problem is to generate complete hypotheses (causal paths from primary causes) for the findings and present the user with a list of hypotheses and their relative total probabilities for comparison. In comparing hypotheses, we discovered that the natural notion of *different* hypotheses requires that they differ in some significant node, nodes which we have labeled *diagnostic*. The algorithm is as follows: 1) check the input for definite implications, findings that require nodes to be true or false; 2) collect the abnormal findings from the input; 3) find all of the diagnostic or primary nodes that could account for each finding; 4) rank the diagnostic and primary nodes by the number of findings they account for; 5) use the better of these as seeds for finding small covering sets of primary nodes; 6) for each covering set, order the findings by the difference between the first and second highest probability path to it; 7) for each finding, the best path from the partial hypothesis is found and added to it; and 8) the hypothesis is pruned of unneeded primary nodes and extra paths that decrease the probability. Finally, the probabilities of the hypotheses are computed by multiplying the probabilities of the nodes given the other nodes in the hypothesis and they are rank ordered and presented to the user. These probabilities could be normalized by the probability of the findings but that is unnecessary as long as we are only rank ordering hypotheses. The algorithm is discussed in detail in a paper [204].

This approach to diagnosis differs considerably from others that have appeared in the litera-



Patient: Willms Session 1 @ 6/29/88 10:45:00
HISTORY: 51 year old male with nocturnal dyspnea, dyspnea at-rest, recent-history-of palpitations and nausea/vomiting and on furosemide digitalis
VITAL-SIGNS: bp: 110/90 hr: 90, monitor: runs-of-vt and sinus-rhythm, rr: 14 and T: 98.6
PHYSICAL-EXAM: appears no-acute-distress, conscious, chest revealed rales 1/2-way-up, jvp: 14 cmH20, normal pulse, auscultation revealed lv-s3 and systolic-ejection-murmur, hepatosplenomegaly, mild pedal-edema and cool/clammy extremities
LABORATORY-FINDINGS: ekg: normal-sinus and lbbb, cxr: generalized cardiomegaly and vascular-redistribution, Na: 129, k: 4.6, bun: 42, creat: 1.4, normal cbc and normal-cpk-mb

Figure 3.1: Congestive Cardiomyopathy and Renal Insufficiency Hypothesis

ture. Reggia's minimal set covering approach [252] ignores the fact that the best hypothesis may not be minimal and would not find the hypothesis in Figure 3.1. Other approaches to diagnosis based on digital circuit analysis [253][88] assume that every node is primary and every node can be measured. If every node were treated that way, a network of this size would be computationally intractable.

Our mechanism is effective for producing a meaningful set of hypotheses for the findings and it usually takes less than a minute on a Symbolics 3650 workstation. The user can compare the hypotheses, see explanations, and consider the differences. Figure 3.1 is the display of the first of five hypotheses generated for an actual patient with findings that included rales, pedal edema, high BUN, nausea, S3, and runs of VT. The display graphically presents the complete explanation for the findings and provides a textual summary of the case at the bottom of the screen. In the display, the findings are in lower case, intermediate nodes in upper case, primary nodes in bold face, primary probabilities in parentheses, causal probabilities on links and *W+* indicating worsening factors that increase the probability and *P−* indicating correcting factors that decrease it. This hypothesis accounts for the findings with congestive cardiomyopathy and renal insufficiency, while the second hypothesis accounts for the findings with congestive cardiomyopathy alone. Those hypotheses nicely capture the physician's initial dilemma: whether the high BUN was renal or prerenal. Other hypotheses included valve disease, which is an important consideration. This hypothesis illustrates several features of the algorithm: 1) it handles multiple causes; 2) it handles multiple pathways between nodes; 3) findings can be left unexplained (the murmur); and 4) iatrogenic causes (digitalis toxicity here) are handled. This kind of explanation is a rich source of information, proposing mechanisms, showing assumptions, showing where therapies might be beneficial, and providing enough information for the user to judge whether the hypothesis is really justified. (This example is discussed in more detail in [205].)

This method of generating hypotheses is heuristic and indeed it is possible to construct networks where it does not find the best answer. (Notice that only the search is heuristic, not the use of probabilities.) However, we have tested over 60 actual cases thus far as well as many created cases and have found the algorithm to be effective. On one set of 42 cases, collected while developing the algorithm, the performance was tabulated. In 31 of these the program produced reasonable hypotheses. In five the hypotheses were almost right but parts of the mechanisms were inappropriate. In the other six cases the best hypothesis was missed. There were two main reasons for these problems: 1) the program did not reason appropriately with the temporal relationships between cause and effect, and 2) it did not handle severity relations appropriately.

3.4.2 Summary

We started with a vision to build a qualitative physiologic model and develop strategies for diagnostic and therapeutic reasoning using the logical relationships between the physiologic entities and input values. From the experience gained from this system, we recognized the need for a probabilistic approach to diagnosis and a quantitative approach to therapy prediction. To fulfill these needs, we created a practical method for heuristically finding the best explanations for a set of findings in a large causal probability network and created a new method for predicting changes in a network of constraint equations based on signal flow analysis. In addition, we developed a method for using the causal model to guide case based reasoning, a statistical method for predicting behavior, a control strategy for time dependent data, and a method for attributing causes to effects over time. These methods give us the basic tools needed to develop an effective program for assisting physicians in reasoning about complex cases over single or multiple sessions.

3.4.3 Knowledge Representation and Default Reasoning

Jon Doyle continued his investigation of artificial intelligence using theories and techniques from economics and decision theory, working in conjunction with Michael Wellman (USAF) and Ramesh Patil. These investigations yielded numerous papers: Wellman and he improved their treatments of default reasoning [99], which it now appears will be published in a special issue of *Artificial Intelligence* on the best papers from the KR'89 Conference in Toronto. His paper with Patil on knowledge representation languages [97] will be published in *Artificial Intelligence* with a response article by Ron Brachman and Hector Levesque. An expansion of his paper with Elisha Sacks (Princeton) on probabilistic qualitative reasoning was presented at IJCAI [98], and has been submitted to *Computational Intelligence.* A paper on the philosophical foundations of AI will be appearing in a MIT Press collection on Philosophy and AI [92]. He will be presenting a paper on rational belief revision at the third Workshop on Nonmonotonic Reasoning in June [93]. Finally, he gave two invited talks this year, and will be giving two more in July. His invited talk at the Conference on the Dynamics of Belief (Sweden) will be appearing in a Springer volume on the Logic of Theory Change [94]. His invited talk at ISMIS'89 appeared in a proceedings volume from North-Holland [91]. He will be speaking on rational self-government at the Second International Conference on Economics and AI (Paris) in July [95], and giving an invited address on the roles of rationality in AI at AAAI'90 in August [96].

3.5 Student Progress

3.5.1 Cardiac Arrhythmia Classification (Scott Greenwald)

Automated cardiac arrhythmia detectors perform well at detecting and classifying beats in clean data, even when compared with humans. However, the performance of automated systems degrades dramatically in the presence of electrode motion noise because of falsely detected QRS-like artifacts and misclassified noise distorted beats. Current systems detect and classify beats using a very limited scope of the available information surrounding candidate beats. In contrast, human experts perform well in noise corrupted data because they use prior knowledge of general principles of electrocardiology and information gleaned from the patient's clean ECG, and because they make use of a wide *context of ECG surrounding* candidate beats. This thesis has explored the hypothesis that the use of wide contextual information within the electrocardiogram and prior knowledge of the signal source can improve beat detection and classification, and enhance artifact rejection in the field of automated arrhythmia analysis. We tested this hypothesis by developing an expert system (HOBBES) that emulates techniques used by human experts in processing ECGs. HOBBES was constructed as a post-processor to a classical arrhythmia detector (ARISTOTLE). The output of ARISTOTLE is an annotation stream which contains an entry for each detected event. Each entry contains ARISTOTLE's suggested beat label, the detection time, an estimate of the noise within the detected event, and a composite QRS morphology measure. HOBBES analyzes ARISTOTLE's annotation stream in three passes and creates a final, error reduced annotation stream as its output. On the first pass, HOBBES learns contextual information by developing a nine-dimensional feature space description of patterns of five-beat sequences seen in clean data. Each five-beat sequence is represented by the five QRS morphology features and the four (scaled) interbeat intervals within the five-beat sequence. On the second pass, HOBBES finds clearly identifiable "landmark" beats (based on morphology or beat arrival times) within the noisy data. On the final pass, HOBBES compares sequential overlapping sequences of the noisy annotation stream to hypothetical five-beat sequences based upon patterns learned in clean data. The best-fitting hypotheses are selected, and final beat labels are assigned to each event. The performance of human experts, ARISTOTLE, and HOBBES was evaluated on a database of half-hour ECG records. Ten percent of each record was selectively corrupted by adding three minutes of electrode motion noise from an independent noise database. The results indicate that although the performance of HOBBES is worse than human experts for many of the records, HOBBES has significantly enhanced ARISTOTLE's performance in processing noisy ECGs.

3.5.2 Repeated Sequences and Parameter Uncertainty in Steady-state Systems (Alex Yeh)

Mr. Yeh has been designing and implementing a program called AIS (short for Analyzer of Iterated Sequences) that, when given a state-description of a system and a sequence of actions or transformations on that state, symbolically finds some of the extreme and time-averaged effects of continually iterating that sequence. The specific effects found at present include 1) the extreme values of parameters that vary periodically with each iteration, 2) the symbolic average rate of change in parameters, and 3) an assessment of how those rates of change would be different with different values for various constants and functions (sensitivity analysis). The sequences handled by AIS are ones which have the following "constancy": the sequence always repeats the same actions in the same order and each occurrence of a given action always changes the parameters by the same amounts. Examples of such iterated sequences of actions include the ones taken by a heart in going through a beat cycle at steady-state and the actions taken by a steam engine in making one rotation of its drive shaft at steady-state. Effects to be found include the extreme pressures in an engine, the average rate at which blood enters the heart, and how increasing that entering blood's pressure affects that rate.

One motivation for finding such effects is to find what stresses a device needs to tolerate, such as the maximum pressure an engine or heart is subject to. A second motivation is that many periodic subsystems iterate at such a fast rate that the other parts of a system respond only to the behavior of such a subsystem β averaged over many iterations. Then a steady-state model for the entire system would only require a description of β's averaged behavior; β can be modeled as a constant iteration of the same sequence of parameter value changes. Examples of such subsystem and system combinations include 1) the heart and the human circulatory system, and 2) an engine and a car.

Yeh worked on three examples of using AIS. The first concerns a normal ventricle (part of the heart). AIS's results are similar to results either derived by others by hand or determined empirically from experiments. The second example is on a ventricle with a disease called mitral stenosis. The model in this example is larger and more ambiguous ("qualitative") than in the first example. The example indicates that AIS can handle fairly ambiguous models, but that the results will reflect that ambiguity. The third example changes domains and is on a steam engine. This example is like the second in that it is larger than the first. But unlike the second one, the steam engine model is a lot more precise on the forms of the functions involved, and AIS's output reflects this. He incorporated AIS's steam engine results into a simple steady-state model of a train.

3.5.3 Multi-criteria Operator Selection (Dennis Fogg)

Automated synthesis of VLSI architectures requires selecting operators to implement arithmetic operations. Two solutions to this task are presented. The first solution extends current AI techniques in parametric design by considering multiple performance criteria. The fundamental obstacle in multi-criteria design is incorporating user preferences about tradeoffs between performance criteria. We present a new framework, called Heuristically Guided Enumeration (HGE) that uses expert knowledge to guide enumeration toward optimal designs, and uses heuristics to control enumeration so better designs are created. HGE encourages the user to explore the frontier of optimal designs by generating fast approximations to the frontier. The user controls the enumeration by defining particular regions of the design space to explore and limiting the cardinality of the designs created. An implemented system enumerates one millionth of the design space yet produces near optimal results. The second solution is based on mathematical optimization methods. We formulate the operator selection task as a linear programming (LP) problem. The LP solution defines a lower bound on the optimal design, and subsequent processing to discretize the solution produces an upper bound. The LP approach is combined with Leiserson and Saxe's retiming method to simultaneously select operators and insert pipeline registers.

3.5.4 Case-based Reasoning (David Aghassi)

In routine problem solving, people reason from experience, remembering their solutions to recurrent problems rather than reconstructing them from scratch each time. The method of case-based reasoning attempts to exploit this intuitive strategy on a computer by maintaining a memory of precedents, and by solving a new case according to the solution of the most suitable precursor. Diverse applications of the method seem to suggest its viability, but a widespread lack of thorough evaluation questions this support. Indeed, while previous work implies that case-based reasoning is successful for a variety of domains, few papers identify the general relationships between performance and the domain characteristics and scaling factors that underlie it. Thus, researchers are left without an understanding of the method's scope or scale, and intuitions about human experience continue to be its primary justification.

David Aghassi's work addresses many of these open concerns in the context of heart failure diagnosis, evaluating the existing case-based reasoner CASEY with respect to a pool of 240 patients. To investigate the method's scale, he measured the effects of increasing experience on both accuracy and efficiency. He also analyzed the distribution of cases in order to quantify its intrinsic regularity, thus exposing the dependence of the system's utility on the domain and facilitating an extrapolation of this utility to other, similarly characterized applications. First, he gauged the recurrence of similar cases in varying size collections of patients; second, he measured the correlation between symptomatic similarity and diagnostic similarity; and finally, he appraised the absolute diagnostic homogeneity of the case pool.

Because cardiologists claim that most cases are variations on recurring, well understood pathophysiologic themes, he expected to justify the application and verify the presumed regularity upon which its success depends. Instead, he discovered that CASEY's accuracy does not increase with experience, while its efficiency degrades with the number of available precedents. Fundamentally, similar cases and similar diagnoses were rare among the 240 patients, and moreover, symptomatic resemblances did not guarantee diagnostic correspondence. Because of the varying combination and interaction of multiple diseases, the patients were largely heterogeneous, suggesting that the regularity described by cardiologists occurs at a more detailed level of abstraction, perhaps in the recurrence of diagnostic syndromes comprised within the cases. This more fine-grain uniformity can be exploited only by analyzing precedents, rather than by applying them in their entirety.

3.5.5 User Modeling for Medical Dialogue Systems (Ira Haimowitz)

We describe a dialogue system between an expert system and its users which combines two recent hypotheses. First, that the dialogue system should explicitly model both the person directly interacting with the dialogue system (the agent) and the person reasoned about by the expert system (the patient) in order to communicate meaningfully with both people. Second, that a dialogue system can model the domain-related beliefs, preferences and concerns of both its users and generate responses empathetic to both.

This dialogue system is called SERUM, standing for "System for Empathetic Responses with User Models." SERUM generates natural language responses about attribute values of domain objects via three transformations. First, the system converts properties of the agent and patient, and domain knowledge, into a pragmatic objectives like empathy. Second, SERUM converts the pragmatic objectives into surface structure cues, like object emphasis and level of technicality. Finally, SERUM converts the surface structure cues to realize text that is natural, appropriately technical, and downplaying or offsetting information that is unpleasant or undesirable to the agent or patient. SERUM is demonstrated in the medical domain of lung disease for AIDS patients, a sensitive domain where empathetic responses can be quite important.

3.5.6 Temporal Control Structure (Thomas Russ)

Thomas Russ continued the development of the Temporal Control Structure, an expert systems development shell for the construction of time dependent monitoring systems. The major improvements were improvements in the run time efficiency of the implementation, the creation of development tools for application program development, and the addition of specialized modules that provide temporal abstractions of data such as dynamic calculations of fluid and electrolyte balance. Agendas for handling asynchronous events were also added to the system.

Mr. Russ is currently extending that development and constructing a prototype application in the acute care of diabetic ketoacidosis. This prototype will provide ongoing management advice for insulin, fluid, and electrolyte therapy in the acute phase of diabetic ketoacidosis. Following development, retrospective medical record trials will be conducted during June and July.

In the past year he explored the role of hindsight in clinical decisions and implemented a demonstration program showing how such reasoning is supported by the Temporal Control Structure [257]. The implementation of hindsight was accomplished through the use of dependency-directed updating and mechanisms for passing information both forwards and backwards along the time line.

3.5.7 Symptom Clustering (Thomas Wu)

Thomas Wu's research deals with a new representation and algorithm based on symptom clustering for diagnosing multiple disorders. The symptom clustering approach partitions symptoms into causal groups, in contrast to the existing candidate generation approach, which assembles sets of disorders, or candidates. In other words, the candidate generation approach explores ways to put disease hypotheses together; the symptom clustering approach explores ways to put symptomatic evidence together.

Symptom clustering achieves efficiency by generating aggregates of candidates rather than individual candidates, and by representing them implicitly in a cartesian product form. Search criteria of parsimony, subsumption, and spanning can help narrow the symptom clustering search space. A problem-reduction search algorithm has been devised to explore this space efficiently. Experimental results on a large knowledge base indicate that symptom clustering yields a near-exponential increase in performance compared to candidate generation. For example, some complex cases that require several hours to solve using the candidate generation approach can now be solved in a matter of seconds using the symptom clustering approach.

In addition to this theoretical foundation, some preliminary work on probabilistic evaluation of symptom clusterings has been completed. Future research will investigate heuristic guides for symptom clustering algorithms, including syndromic knowledge, which seems to be important in human clinical cognition.

3.5.8 Decision Models (Tze-Yun Leong)

Characterizing the knowledge involved in decisions illuminates the representational and computational requirements for the decision-analytic approach to automated clinical decision making. This work analyzes the medical knowledge required for formulating decision models in the domain of pulmonary infections (PIs) with suspected acquired immunodeficiency syndrome (AIDS). Based on the analysis, a knowledge representation framework is proposed. The framework is evaluated by showing how it supports decision model formulation for an example case.

Aiming to support dynamically generated decision models, the knowledge characterization focuses on the structural aspects of the decision problem, such as the clinical context, the classes of evidence, hypotheses, tests, treatments, outcomes, and the behavioral relationships among them. *Concepts,* which model the objects, states, processes, and their attributes in the clinical setting, are the basic building blocks of the representation design. A language with set-theoretic and probabilistic semantics is devised to describe the concepts and their context-independent and context-dependent relationships.

3.6 Publications

[1] J. Doyle. Reasoning, representation, and rational self-government. In Z. W. Ras, editor, *Methodologies for Intelligent Systems, 4*, pages 367–380, North-Holland, 1989.

[2] J. Doyle and R. S. Patil. *Two Dogmas of Knowledge Representation: Language Restrictions, Taxonomic Classifications, and the Utility of Representation Services.* Technical Report MIT/LCS/TM-387.b, MIT Laboratory for Computer Science, September 1989.

[3] J. Doyle and E. P. Sacks. Stochastic analysis of qualitative dynamics. In N. S. Sridharan, editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1187–1192, Morgan Kaufmann, 1989.

[4] J. Doyle and M. P. Wellman. Impediments to universal preference-based default theories. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 94–102, Morgan Kaufmann, May 1989.

[5] I. Haimowitz. Modeling all dialogue system participants to generate empathetic responses. In *Proceedings of the Second International Workshop on User Modeling*, March 1990.

[6] I. Haimowitz. User modeling for medical dialogue systems. In *AAAI Spring Symposium in AI in Medicine*, pages 59–63, March 1990.

[7] I. J. Haimowitz. *Generating Empathetic Responses with Individual User Models.* Master's thesis, Cambridge University, England, 1989. M. Phil., Computer Speech and Language Processing.

[8] N. Harris. Probabilistic reasoning in the domain of genetic counseling. In *Symposium on Computer Applications in Medical Care*, 1989. Finalist, student paper competition, SCAMC 1989.

[9] N. Harris. *Probabilistic Reasoning in the Domain of Genetic Counseling.* Technical Report MIT/LCS/TR-460, MIT Laboratory for Computer Science, October 1989.

[10] T. A. Russ. Using hindsight in medical decision making. *Symposium on Computer Applications in Medical Care*, 1989. Finalist in student paper competition.

[11] T. D. Wu. Symptom clustering and syndromic knowledge in diagnostic problem solving. In *Proceedings of the 13th Symposium on Computer Applications in Medical Care*, pages 45–49, IEEE Computer Society, November 1989.

Theses Completed

[1] D. Aghassi. *Evaluating Case-based Reasoning for Heart Failure Diagnosis.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[2] S. Greenwald. *Improved Detection and Classification of Arrhythmias in Noise-corrupted Electrocardiograms Using Contextual Information.* PhD thesis, Harvard University and MIT Division of Health Sciences and Technology, May 1990.

[3] N. Harris. *Probabilistic Belief Networks for Genetic Counseling.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

Theses in Progress

[1] D. Fogg. *Artificial Intelligence and Optimization Solutions to Multi-criteria Operator Selection.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[2] T-Y. Leong. *Knowledge Representation for Supporting Decision Model Formulation in Medicine.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[3] T.A. Russ. *Reasoning with Time Dependent Data.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[4] T. Wu. *Medical Diagnostic Problem Solving Using Multiple Types of Knowledge.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[5] A. Yeh. *Automatically Summarizing Repetitive Actions and Handling Parameter Uncertainty in Systems at a Steady-state.* Phd dissertation, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

Talks

[1] N. Harris. Probabilistic belief networks for genetic counseling. Lecture given at the 13th Symposium on Computer Applications in Medical Care, Washington, DC, November 7, 1989.

[2] N. Harris. Criticizing conditional probabilities for belief networks. Lecture given at Stanford University, January 23, 1990.

[3] N. Harris. Probabilistic belief networks for genetic counseling. Lecture given at the AAAI Spring Symposium on Artificial Intelligence in Medicine, Stanford University, March 28, 1990.

[4] N. Harris. GenInfer: Probabilistic belief networks for genetic counseling. Lecture given at Laboratory for Computer Science, Massachusetts General Hospital, June 1, 1990.

[5] T. Russ. Temporal Control Structure. Lecture given at the AAAI Spring Symposium on AI in Medicine, Stanford University, March 1990.

[6] T. Wu. Symptom clustering and syndromic knowledge in diagnostic problem solving. Lecture given at the 13th Symposium on Computer Applications in Medical Care, Washington, DC, November 1989.

Computation Structures

Academic Staff

Arvind, Group Leader	G.M. Papadopoulos
J.B. Dennis	A. Vezza
R.S. Nikhil	

Research Staff

G.A. Boughton	J. Young
R.P. Johnson	

Technical Staff

J.P. Costanza	R.F. Tiberio

Graduate Students

S. Aditya	C.F. Joerg
Y. Chery	V.K. Kathail
P.S. Barth	B.C. Kuszmaul
D. Chiou	J.S. Onanian
S.A. Brobst	A. Salaam
S. Glim	M. Sharma
D.S. Henry	A. Shaw
J.E. Hicks	K.M. Steele
A.K. Iyengar	

Undergraduate Students

G. Adams	S. Parekh
A. Caro	D. Plass
J. Ferrera	D. Stetson
S. Furman	R. Tewari
R. Lustberg	C. Tse
F. Madero	

Support Staff

S.M. Hardy	F. Ugwuegbu
B. Radle	

Visitors and Adjunct Members

A. Altman	M. Heytens
Z. Ariola	R. Kreuter

4.1 Introduction

Our group is interested in general purpose parallel computation. Our approach is centered on:

- Declarative, implicitly parallel languages.
- Dataflow architectures, which are scalable because of their tolerance of increased memory latencies and support for frequent synchronization. Our vehicles for research include an abstract "Explicit Token Store" architecture (ETS), hardware prototype implementations of ETS (called Monsoon), various software emulators (GITA, MINT), and a software emulator for a new proposed architecture called P-RISC.
- Sophisticated compiling and run-time systems for Id, both for dataflow and other architectures. We have also explored the use of dataflow compiling for an experimental persistent programming language.
- Applications programs to guide the language, compiler, and architecture research.

Last year, we reported that we began negotiations with Motorola for a project to produce, as a research prototype, a complete system running Id on dataflow machines using the Monsoon processor architecture. This year, MIT-Motorola cooperation has moved into high gear. This involves extensive and daily cooperation in the design and production of the Monsoon hardware, and in the design and production of the Id programming environment. Two-node prototypes are expected by the end of summer 1990, and 16-node machines by spring 1991. To this end, a formal cooperation agreement has been signed, and Motorola has established and staffed the new Motorola Cambridge Research Center at One Kendall Square, next door to LCS.

Our main research vehicle for programming languages is Id, which has fine-grained, implicit parallelism. We have been able to formalize our incremental typing system for Id and to prove it correct. We have made much progress in developing the "manager" construct in Id, which is a disciplined way of using imperatives while retaining fine-grained, implicit parallelism and synchronization. We have continued our work in formalizing Id's operational semantics in terms of abstract reduction systems. New applications in Id include the Traveling Salesman Problem using simulated annealing, the Viterbi Search from speech recognition systems, and various sparse matrix algorithms.

We have made almost a complete transition from the TTDA (Tagged Token Dataflow Architecture) compilation schemas to new schemas that incorporate the notion of frame storage in an integral way. Frame storage is now used for extensive loop optimizations. A new back-end translates these frame-oriented dataflow graphs into code for Monsoon. We have been studying resource management for Id in great detail, including compiler-directed garbage collection, as well as numerous versions of frame and heap managers for improved concurrency.

The porting of Id World to the UNIX environment (from our original Lisp Machine environment) is complete, and has been distributed outside MIT.

The Monsoon wire-wrap prototype, which has now been running for over a year, has been invaluable for testing our ideas in resource management in Id, for measuring instruction mixes, and for designing its successor.

The second generation Monsoon processor has been designed using various ASICs. Motorola has done the board-level design and is fabricating it. The processor incorporates substantial improvements from the wire-wrap prototype in speed, functionality, and connection to the UNIX world. An I-structure board has also been designed and is being fabricated by Motorola. The Monsoon interconnection network switching chips (PaRCs) and data link chips (DLCs) have been designed and fabricated, and are undergoing testing. A 4-by-4 network board has been designed by Motorola and is being fabricated.

In other work: Vinod Kathail has completed his Ph.D. thesis on optimal interpreters for the lambda-calculus; we have continued our research on P-RISC, a synthesis of von Neumann and dataflow architectures; we are close to having a stock hardware implementation of Id; and, we are close to having a dataflow implementation of a parallel persistent language.

In addition to cooperation with Motorola, we continue to maintain strong and active contacts with several other dataflow researchers outside MIT. Members of our group have participated in the international committee that designed the new, standard, functional programming language Haskell.

4.2 Personnel and Visitors

In January 1990, Greg Papadopoulos was appointed to the MIT faculty in the Department of Electrical Engineering and Computer Science. He has been a member of our research staff since August 1988.

In December, Arthur Altman of Texas Instruments completed a year as a visitor in our group, and has transferred to Steve Ward's Computer Architecture Group.

Rudiger Kreuter from Siemens, Germany, spent the Fall of 1990 as a visitor in our group. In addition to learning about Id and dataflow, he studied the implementation of 3D graphics in Id.

As usual, we have had a steady stream of international scholars for short visits and talks.

4.3 MIT-Motorola Collaboration on Id and Monsoon

Through the concerted efforts of Albert Vezza, Associate Director of LCS, and Jerzy Skibinski, Vice President of Motorola's Microcomputer Division, a joint Research Agreement with Motorola's Computer Division of Tempe, Arizona was formalized in August 1989, although cooperation had been ongoing for seven months in anticipation of the signing.

The joint effort will result in at least three 16-node Monsoon research prototypes and at least sixteen 2-node versions. The division of labor between MIT and Motorola is as follows: MIT is responsible for the overall system, logic and chip designs, chip fabrication, a novel special tool for generating microcode from opcode specifications, the Id language, and compiler design and development. Motorola is responsible for all board-level, enclosure, supporting hardware infrastructure and I-structure logic design, development, and manufacturing. On the software side, Motorola is responsible for the Monasm assembler, dynamic linking loader, command line interpreter user interface, all host level software, and debugging tools including a Monsoon simulator.

Motorola's project is managed by Jim Richie. Their hardware work is done at their facility in Arizona, while their software work is done mostly in Cambridge at the new Motorola Cambridge Research Center (MCRC), which they have established as part of this project. The immediate focus of MCRC, which is in the Kendall Square office complex next to LCS, is close cooperation with MIT in the research and development of software for Monsoon. In the long run, MCRC is expected to take its place alongside the many fine basic research labs in the vicinity of MIT. The first employee of MCRC was Ken Traub, who completed his Ph.D. in our group in 1988. Ken was the original architect and builder of our Id compiler. As a member of MCRC, Ken is playing a leading role as overall architect of the Monsoon software system.

During the year, we have held numerous review and planning meetings with Motorola:

- August 1, 1989: Review meeting at MIT.
- September 27-28, 1989: Software and contracts meeting at MIT.
- October 19, 1989: Review meeting at MIT.
- December 7, 1989: Monsoon technical discussion meeting at MIT.
- January 25-27, 1990: Monsoon hardware and software progress review meeting at Motorola, Tempe, AZ.
- March 29-30, 1990: Review meeting at MIT.
- June 25, 1990: Monsoon hardware and software progress review meeting at Motorola, Tempe, AZ.

We are happy to report that all critical milestones to date have been met. We expect that the first 2-node prototype will be available during the third quarter of 1990, and the first 16-node prototype during the second quarter of 1991.

4.4 Other External Collaborations

Our work on Id and Monsoon has led to collaborative efforts with many research groups outside MIT.

Upon leaving MIT after finishing his Ph.D. thesis, Bob Iannucci has started the Empire Project at IBM Research, whose goal is to build a hybrid dataflow-von Neumann machine similar to the one he proposed and studied here in his thesis. We also continue to collaborate with K. Ekanadham of IBM Research, who is leading the effort to target our Id compiler for that machine. During the summer of 1989, Shail Aditya worked at IBM Research on their Id compiler.

At Sandia, a group of researchers led by Gerald Grafe is building the Epsilon dataflow machine which is similar to Monsoon in many respects. Jamey Hoch of Sandia is working on retargeting our Id compiler for Epsilon. In addition, they will be using our PaRC network switching chip in the interconnection network for their multiprocessor. Ken Steele has just left MIT to join the group at Sandia. We have participated in several meetings to discuss collaboration with the Sandia project:

- July 31, 1989: Sandia-DARPA meeting in Washington D.C.
- March 11-13, 1990: Cooperation meeting at Sandia, Albuquerque, NM.

Karl Ottenstein of Los Alamos and Bob Ballance of the University of New Mexico are working on a compiler for FORTRAN on Monsoon; they plan to use the backend of our Id compiler. Similarly, Keshav Pingali of Cornell is also investigating the implementation of imperative languages on a dataflow machine—he, too, plans to use the backend of our Id compiler in order to run his codes on Monsoon.

In a separate, but related activity, Arvind, Rishiyur Nikhil, and Jonathan Young were members of the design team for Haskell, the new, nonstrict functional programming language. The team included about 15 prominent researchers in functional programming from the U.S. and Europe. The report on Haskell was published in April 1990. It is hoped that the international research community will adopt this language as the standard for nonstrict functional languages.

On November 1-3, 1989, we held a Dataflow Workshop here at MIT. In addition to researchers from all the above groups, participants included recent graduates from our group, and researchers from Yale, Manchester University, Tera Computers, Rice University, Oregon Graduate Institute, Glasgow University, Motorola, and Hewlett Packard Labs.

On April 26-27, 1990 we held a Software Cooperation Meeting here at MIT, again attended by researchers from most of the above groups. The focus was on discussing how each of us could structure our work to maximize sharing, since many of us are interested in targeting other languages to Monsoon and in targeting Id to other machines.

One of the outcomes of the Software Cooperation Meeting was a consensus among our guests that we needed to run a workshop dedicated to furthering the understanding of the internal structure of the Id compiler. This workshop has been scheduled for June 28-29, 1990 at MIT.

On February 2, 1990, we held an internal (MIT) workshop on multithreaded architectures with participants from our group and the groups led by Anant Agarwal, Steve Ward, Bill Dally, Tom Knight, as well as Bert Halstead from DEC Cambridge Research Center. The intent was to get a better understanding of each others' work, since all are exploring different kinds of multithreaded architectures.

As usual, on July 24-28, 1989, the summer dataflow course (6.83s), was taught here at MIT by Arvind and Rishiyur Nikhil. The course was attended by approximately 20 external researchers.

Also, in November 1989, Arvind and Rishiyur Nikhil taught a one-day tutorial on Id at the Supercomputing '89 Conference in Reno, Nevada.

4.5 Id: General Topics

4.5.1 Types and Incremental Type-checking

Continuing his work on the incremental type inference system for Id, Shail Aditya developed an abstract model for incremental property maintenance and applied it to show the correctness of the incremental type inference system developed for Id.

Incremental programming environments, such as Lisp, aim at providing the user the flexibility to write a sequence of definitions constituting the program, one by one and in arbitrary order, resolving global references to other definitions dynamically. They allow editing and testing parts of an incomplete program or debug those parts that are incorrect, without worrying about the status of the rest of the program. However, the Hindley/Milner static type inference system [227][84] followed in Id does not naturally lend itself to incremental compilation. Nikhil in [230] discussed the issues involved and outlined a high level mechanism to do it.

Following Nikhil's proposal, Shail Aditya devised an abstract scheme to adapt the Hindley/Milner type inference system for incremental compilation. Subtle incremental interactions were discovered between the types of a given set of definitions and their partitioning into strongly connected components (SCC), definitions that are mutually recursive with each other. Development of the necessary theoretical framework guided modifications in the scheme to handle polymorphic and mutually recursive definitions correctly. Essentially, the present scheme consists of maintaining an upper and a lower type bound for each top level identifier along with its current SCC. Inconsistencies arising due to the declared type falling out of the expected range, or because of a change in its SCC, are detected and the affected definitions are flagged for recompilation. The goal is to show an exact correspondence between the types inferred in the incremental scheme with those inferred when a complete and correct program is given, while at the same time performing minimal recompilation work due to an incremental change in the program. The detailed proofs of the correspondence are due to appear in Shail Aditya's forthcoming master's thesis. Future work in this direction will be to optimize the space and time requirements of the incremental bookkeeping done by the compiler.

4.5.2 Managers

Paul Barth and Rishiyur Nikhil continued their work on managers. Managers add nondeterminism to Id, an important property for state-sensitive computation, as required by operating systems, databases, and I/O. Although nondeterminism is a powerful feature, it introduces a new class of programming errors: irreproducible results. We are addressing this at the language level by encapsulating managers in abstract type definitions. A manager is an abstract type, consisting of an updatable state and a set of operations that access and update the state. Each operation computes a new state from the old state. Mutual exclusion is provided for the state so that concurrent operations do not interfere. This encapsulation allows state invariants to be proved by proving them for each operation.

Several challenging efficiency concerns are now being addressed. For space efficiency, managers with a complex state should mutate the state rather than copy it. For parallelism, such managers should provide fine-grained mutual exclusion, so that independent operations can proceed concurrently. These concerns raise syntactic and semantic issues in the design and implementation of managers. A new syntax has been proposed that addresses these issues while maintaining a clean abstraction.

Manager applications have been written for graph algorithms, sorting, memory management, union-find, and parallel priority queues. The traveling salesman problem was coded using a simulated annealing algorithm, using managers for both path mutation and random number generation. A detailed study of potential parallelism and synchronization bottlenecks was performed on several variations of the algorithm.

4.5.3 Sequentialized Code Execution

We are currently experimenting with writing resource managers and operating systems in Id. These kinds of programs, which make use of imperative side-effects, must have explicit sequentialization of reads and writes.

James Hicks has extended the Id language and compiler for sequential constructs. The new syntax sequentializes the execution of groups of bindings in let blocks or loops. To sequentialize a group of bindings, use '&' instead of ';' to separate the bindings, as shown in this example:

```
{ x0 = e0;
  x1 = e1
& x2 = e2;
  x3 = e3
in
  e4 }
```

This ensures that the evaluation of e2 and e3 does not begin until all computation has ceased in the previous two bindings. Note, however, that e0 and e1 may execute in parallel, and that e2 and e3 may execute in parallel—sequentialization is only inserted between binding groups separated by '&'.

Parentheses may be used to group bindings and enforce more arbitrary synchronization graphs. Here is an example:

```
{ x0 = e0;
 ( x1 = e1 &
   x2 = e2 )
 in e3 }
```

In this example, e0 may execute in parallel—with e1 and e2, but e2 may not begin execution until expression e1 terminates.

4.5.4 Formalization of Id's Operational Semantics

Zena Ariola and Arvind have continued their work on giving precise operational semantics for Id. The approach consists of translating Id into a simpler and smaller kernel language, and giving semantics of the kernel language in terms of an Abstract Reduction System (ARS). In order to prove the correctness of compiler optimizations, a notion of program equality is needed. Such a notion is easier to define for an ARS than an interpreter. P-TAC, an earlier attempt to define such a language, and ARS were reported last year [13].

P-TAC was a simple and a low level language that allowed us to capture most aspects of the current implementation of Id on the dataflow machines. However, it was so far from the source language that the translation procedure (from Id to P-TAC) became a serious impediment in understanding the operational semantics of Id. Even though it allowed many program optimizations to be described in terms of source-to-source P-TAC program transformations, it ruled out certain other program optimizations because the information to perform them was essentially lost in the translation process.

Kid, our current kernel language, is essentially a de-sugared version of Id, which is Id without comprehensions, general union types and pattern matching, and nondeterministic features such as managers. Both array and list comprehensions can be expressed in terms of other Id features such as loops and "open" lists. Though such a translation is not simple, it can be understood in its own right. Similar remarks apply to complex pattern matching. The stage at which type-checking should be performed is still an open question. Given the type definitions, type checking can be done at the Kid level though it may be profitable to do so at an earlier stage.

An ARS for Kid, which includes nested function definitions and loops, has been defined. Many loop optimizations and partial evaluation have been expressed as Kid source-to-source transformations. The work on formalizing the printable output and termination of Id programs is underway.

4.6 Id: Compiler and Run-time Systems for Monsoon

4.6.1 New Compilation Schemas for Dealing With Frames

James Hicks implemented the code generator for the bounded loop schema for the Monsoon backend. The TTDA bounded loop schema is much different from the Monsoon bounded loop schema because each iteration executes in a different context on Monsoon, while in TTDA only the iteration number within a context changed. This necessitates a change in the `D` operator that routes tokens from one iteration to the context, or activation frame, of the next iteration. Another change is that the synchronization that allows only k iterations to execute at once must be performed using locks and two-phase transactions—this synchronization was performed with bit-vectors and special instructions in Gita.

The new bounded loop schema consists of three parts: setup, iteration, and cleanup. The setup portion consists of a 1-bounded, or sequential, loop that allocates a ring of activation frames and fills in the loop constants in each frame. The iteration portion consists of the actual loop body plus the glue necessary for synchronization and to route tokens to the next iteration or to the outputs of the loop. The cleanup portion of the loop clears and deallocates each iteration context. We have taken much care so that the setup, iteration, and cleanup portions of the loop may be overlapped to reduce the latency incurred by loops that execute few iterations. The setup portion allows the iterations to begin as soon as it has one activation frame setup. When the loop predicate evaluates to false, the cleanup portion of the code is triggered with the continuation of the next context in the ring, which is guaranteed to be inactive at that point. The cleanup code starts with that context, and continues around the ring with the proper synchronization to ensure that it does not overrun the loop body.

4.6.2 Staging the Instruction Set Development for Monsoon

The macroinstruction set of Monsoon is a "soft" interface, such that an opcode is successively decoded through the pipeline by downloadable lookup tables. The decode tables are set up by a host processor whenever Monsoon is cold-booted. In Monsoon, an opcode encodes the effective address mode, the matching mode (e.g., join, constant, imperative), the ALU operation, and the number and disposition of result tokens. For example, the double precision floating point subtract operation FSUB consumes 32 opcodes for all of its variants of one vs. two outputs, constant vs. dyadic matching, etc.

The software Monsoon interpreter, MINT, is also indirectly driven by the decode tables. A preprocessing program, MUD, takes the decode tables as input and, for each opcode, produces a C (originally Lisp) subroutine which is later compiled and linked into MINT.

In order to manage the "bring-up" and validation of the compiler, the software which generates the decode tables and MINT, we have partitioned the instruction set into three subsets to be developed in stages. The first set, *IS0*, has approximately 60 opcodes (out of a possible 2048) and represents a very minimal instruction set. *IS0* supports frame and I-structure allocation, but no deallocation and only single deferred readers. Exceptions are not supported, nor is the run-time type system (Id is statically typed).

The next stage, *IS1*, brings the total to a few hundred opcodes and is capable of supporting the entire Id language, including closures, accumulators, managers, storage reclamation, and multiple deferred reads against I-structures.

The final stage, *IS2*, encompasses the whole instruction set, including experimental extensions for temporary registers and threading. As of June 1990, the *IS0* set has been certified by a series of tests on our gate-level simulator of a Monsoon processing element, and an *IS0* version of the compiler and MINT has executed a simple successive over-relaxation 2D wavefront problem.

4.6.3 New Backend for Monsoon

This past year Andrew Shaw has implemented a new backend for the ID compiler to transform ID program graphs into Monsoon machine language. Prior to this, we had been generating code for the Monsoon wire-wrap prototype using an interim backend that was a modification of the original TTDA (Tagged Token Dataflow Architecture) backend. The new backend uses the same data structures as the middle end of the compiler, and several new optimizations have been implemented, along with the standard peephole optimizations that were in place with the old TTDA backend. For example, the calling convention constrains the entry points of procedures to lie in consecutive address locations; one of the new modules can relax the conservative selection of these instructions. Since the Monsoon architecture has some assembly constraints on the layout of machine code, new algorithms were designed to enforce these constraints upon the final output code. For example, two-output instructions are constrained to have one of their destination instructions in the following instruction slot; a new bipartite-matching algorithm was implemented to find a near-optimum selection of successor-constrained instructions.

In addition, an interim loader was implemented to interface with the new Monsoon Interpreter, since the full loader has not yet been implemented.

4.6.4 Compiler-directed Storage Reclamation for Id

We have been experimenting with structure-storage management over the past year. Informal studies have shown that functional languages typically "cons" at four times the rate of Lisp programs. This high rate of storage allocation means that functional programs have a great dependency on garbage collection. Unfortunately, garbage collection can be very expensive, especially on a parallel machine.

We have introduced a pragma, **@Release**, into Id to annotate structures that are temporary and that should be deallocated. When the Id compiler sees an **@Release** annotation on a structure, it inserts code to deallocate the structure upon termination of the nearest enclosing conditional branch, procedure body, or loop iteration.

There is one further optimization the compiler can perform with **@Releases**. If a structure is allocated in a loop, and deallocated in the same or next iteration, then the compiler can lift the allocate and deallocate out of the loop to reduce the overhead of calls to the storage manager. Outside the loop, k copies of the structure will be allocated, where k is the loop bound. These will be used by the iterations of the loop. After the loop terminates, all k structures will be deallocated.

The **@Release** pragma has been used extensively by Olaf Lubeck in his Id implementations of the Gamteb photon transport benchmark and the Particle-in-Cell (PIC) code. In October 1989, Olaf Lubeck, James Hicks and Paul Johnson got Gamteb to run on the Monsoon prototype. This version of Gamteb was annotated so that it did not leak any storage—all structures that became garbage were deallocated. The largest problem run on the prototype started with 40,000 particles. It allocated 300,000 9-tuples, 200,000 3-tuples, and 270,000 activation frames of size 512. When it completed, only 616 words of storage were still allocated—and that contained the answer. This is quite impressive considering that the prototype only has 128K words of memory, and only half of that is used for the heap. This work has shown that explicit structure-storage management is useful; it allows us to run programs that could not be run otherwise.

James Hicks is working on compiler analysis for the verification and automatic insertion of **@Releases** in his Ph.D. research. The goal of this work is to have the compiler analyze programs to determine the lifetime of structures, and to insert code to deallocate structures that are no longer needed. The compiler performs lifetime analysis by using abstract interpretation—it interprets the program over an abstracted value domain at compile time in order to determine which expressions in the program allocate structures, and where those structures may be used.

In scientific codes, which tend to have very regular control and data flow, most (if not all) of the structures that become garbage should be detected and deallocated by the compiler. Hicks has performed some experiments with simple program analyzers that support this belief. His analyzer detected 23 of the 25 `@Releases` inserted into Gamteb by Lubeck. The analyzer also was run on Simple; in this case the compiler inserted deallocates that at run time deallocated 85% of the garbage created by four iterations of Simple on a 10 by 10 grid. By the end of fall 1990, the compiler should be able to insert code to deallocate all of the garbage created by this program.

4.6.5 Run-time Systems for Id

Jonathan Young has led a major effort on developing the Id Run-time System (RTS). The RTS is designed to be a flexible interface to the low level primitives needed to execute Id programs. The RTS is composed of three main parts: the allocation and deallocation of *contexts*, both fixed- and variable-size, for procedure invocations; the allocation and deallocation of *aggregates* for data structures; and the management of input from and output to the outside world. Most of our work this year has focused on the first two parts.

As a stopgap measure until we have a simulator capable of executing SVC trap instructions, we have coded primitive heap and context allocators for the new Monsoon machine which maintain two pointers to the beginning and end of the heap. Since they do not reuse storage, when the pointers cross, the machine dies. These allocators are generated inline by the compiler, and have allowed the rest of the software development to proceed.

We expect that as IS1 (instruction set level 1, including SVC instructions) becomes operational on the simulator, we will be able to test and debug the full functionality of the two-phase operations and the exception mechanism. Once this happens, the Id RTS can begin to execute via SVC instructions, although most of the RTS handlers will simply make a procedure call at this point. When registers are finally added to the simulator (IS2), we expect that the RTS will become much more efficient.

There are several problems to be addressed in order to manage storage efficiently on a Monsoon multiprocessor. In particular, we must avoid network traffic—most of the manager code must be completely *local.* We also wish to ensure that the critical sections are short and the reuse of memory is as high as possible.

On the new Monsoon machine, each processor will allocate contexts locally, and using the exception mechanism, each thread must be able to allocate a context independent of the other threads in the pipeline. We have designed and implemented (but not debugged) a scheme which, on average, achieves this behavior by caching a small number (16) of contexts with each thread while linking the rest into a processor-global context free list.

Under this scheme, each allocation (and deallocation) of fixed-sized contexts will take approximately six instructions (exception, load cache pointer, return fetched context to caller, increment pointer, and store pointer) normally and 20 instructions for the exceptional case that the free (or empty) list has over- or underflowed. Since this happens statically once every 16 operations, the amortized cost averages out to no more than seven instructions.

On the new architecture, we aim to solve several new problems when allocating objects from the global heap. First, the heap will be *interleaved* across multiple nodes in the system. While no work is needed to achieve this interleaving, the heap manager will need to create pointers which take full advantage of the hardware interleaving mechanism.

Second, multiple processors will be handling multiple simultaneous heap requests. Even though the heap is remote, we desire to handle the majority of all requests locally. This requires some PE-local heap data structures. Finally, we wish to avoid interference between different threads executing in parallel on the same PE.

The heap manager on the Monsoon multiprocessor is a hybrid of two schemes. Each processor will run a *local allocator*, a version of quick-fit which utilizes the assumption that if an object is deallocated, it is likely that another object of the same size will be allocated. When the local allocator runs out of memory, however, it will ask the *global allocator* for more storage, pre-allocating a large block of memory for future requests. While allocating storage on the global heap does require network traffic, this should be tolerable because it is so infrequent.

Arun Iyengar has also become actively involved in designing the Id storage managers.

4.7 Applications

Paul Barth and Stephen Brobst implemented a number of approaches to parallel simulated annealing for the traveling salesman problem. Manager extensions to the Id programming language were used to facilitate the implementation of critical sections while performing update-in-place operations on the adjacency matrix of cities in the algorithm. A number of paradigms for the use of managers in implementing the algorithm were explored: Compare and Swap, Canonical Ordering, Master Lock, and Locking with Back-Off. It was found that fine-grained parallelism was exposed very naturally in the model of execution provided by dataflow. However, it was found that there is significant coarse-grain sensitivity within the program to the particular algorithm implemented for managing critical sections. It was important to not over-serialize execution of loops corresponding to different temperatures in the simulated annealing algorithm. However, it was also critical to consider the contention resulting from too many parallel loop iterations.

This contention resulted from two sources. One source was the contention for the current seed value of the random number generator. This problem was addressed by initiating multiple, parallel random number generators. The other source of contention was on the cities to be swapped. In general, to swap two cities, it is required to grab locks on six cities (the two to be swapped as well as the two neighbors for each city). The methods of managing this locking had a large impact on performance. As expected, any use of global locks introduced a major synchronization bottleneck for large instances of the problem. By optimizing the "fast path" of execution through the locking primitives, we were able to reduce, but by no means eliminate, the impact upon the critical path of our computation.

Amin Salaam and Rishiyur Nikhil have been studying an implementation of the Viterbi Search in Id. Viterbi Search is a key component of speech recognition systems. The inputs to the search are an Acoustic-Phonetic network (APnet) and a dictionary. The APNet is a graph generated by an earlier phase that performs signal processing on the acoustic signal of the speech utterance. Each arc in the graph represents a time interval in the utterance, and is labeled by a list of probabilities, indicating how well the signal in that time interval matched each of all possible phonemes. The dictionary is also a graph structured around words. Each word is represented by a sequence of arcs corresponding to phonemes, and words that can legally appear in sequence are also connected by arcs. Dictionary arcs are also labeled with probabilities indicating possible omission of insertion in an actual utterance. The Viterbi Search algorithm matches paths in the APnet to paths in the dictionary, finally emitting the "most likely" sequence of words in the utterance. Because of this complex graph structure, the algorithm has not been effectively parallelized to date. We have been studying existing code (in C and Scheme) for this algorithm since March 1990 in order to extract the fundamental aspects of the algorithm. We plan to produce a parallel implementation in Id and to run it on Monsoon. We expect that this will shed much light on dataflow implementations of graph algorithms in general.

Other applications written by Rishiyur Nikhil and Arvind in Id include: a simulator for a very abstract model of an Explicit Token Store (ETS) dataflow machine (of which Monsoon is a concrete example), and various versions of LU-decomposition for dense and sparse matrices.

4.8 Id World, the Id Programming Environment

Development of Id World, the Id programming and experimentation environment, continues with a focus on advances which benefit both the old TTDA/Gita system and the software system which will support Monsoon. Paul Johnson has worked on improvements in Id World on UNIX workstations and software system construction tools. Id Mode for Gnu Emacs provides source code indentation and compilation of Id programs. With the assistance of Hicks, simulator statistics graphs and overlays of multiple graphs are available under the X Window System. Id World Version 4.3, which was released in April, provides these improvements and Id compiler pragmas for explicit structure storage management. As of Version 4.1, Id World runs on UNIX workstations under Common Lisp. Version 4.3 has been tested under Allegro Common Lisp and Lucid Common Lisp running on Mips, Motorola, and Sun workstations. We were unsuccessful in running Id World under Austin-Kyoto Common Lisp (AKCL) due to deficiencies in language support for error handling. Improvements in software system construction tools include: handling file system specifics—translation of program filenames and logical pathname support in the Defprogram facility, startup initializations and banner customization for Lucid and Allegro disk images, consistent versions of internal software—generation, and use of generic patch files for our development systems.

4.9 Monsoon Hardware Development

4.9.1 Monsoon Wire-wrap Prototype Processing Element

The Id compiler now produces Monsoon object code for all of the Id language. The only restriction to running an Id program on the prototype is the size of memory—the prototype has only 128K words of data memory, of which half is used for I-structure storage and half for activation frame memory.

We have a run-time system, written in Id and Monsoon assembly code by Young and Hicks, that manages the allocation and deallocation of activation frames and heap storage on the Monsoon processor. Activation frames are managed on a free list, and heap storage is managed by either the buddy or first-fit system. Implementation of the storage managers has been very difficult because concurrent calls to the storage manager can be interleaved on an instruction-by-instruction basis.

The prototype is currently used to measure performance and instruction mixes of applications under the Monsoon instruction set architecture with the run-time system described above. These measurements have shown that Id can be run efficiently on Monsoon. The dynamic instruction mixes collected while running Id applications on Monsoon have shown that the architecture is well matched to the language, and not too much overhead is introduced to support instruction-level parallelism. However, some of the architectural restrictions imposed by the implementation of the wire-wrap processor, such as the restriction of one explicit destination per instruction and 10-bit offsets for addressing of frame locations, have caused excessive difficulty in compilation and have introduced excessive overhead in the object code. The processor has been redesigned to mitigate these problems.

These performance measurements have also shown that better algorithms must be used for storage management so that concurrent invocations of the storage manager will not cause excessive sequentialization of the program. This sequentialization will be magnified on multiple-processor Monsoon systems unless better algorithms are used. Young and Armando Fox have designed and implemented a storage manager that uses multiple local free lists and local heaps to allow concurrent access to the heap.

4.9.2 The Monsoon Processing Element (Second Generation)

We have made significant progress in the design and verification of a second generation Monsoon processing element (PE). This board-level design was transferred to the Motorola Microcomputer Division, where it was successfully layed-out and routed. It is now in the process of being manufactured. We also designed and fabricated two application-specific integrated circuits (ASICs): a byte-slice of the datapath and tag/pointer ALU. Greg Papadopoulos was responsible for the overall PE and ASIC architectures. Jack Costanza and Ralph Tiberio executed the detailed designs and simulations.

The original Monsoon wire-wrap prototype processor was made operational in 1988, executing its first compiled Id program in October of that year. The new PE is largely compatible with the original prototype, employing an eight-stage pipeline, 64-bit datapaths (plus eight bits of type), and 32-bit instructions. The new PE differs in several important respects:

- **Interprocessor Network:** The first prototype could operate only as a uniprocessor because it lacked an interprocessor network. The new PE integrates the network into the processor by employing an on board PaRC and Datalink chips, as well as input and output FIFO buffers.

- **VME Interface:** The new PE is hosted on a 9U form-factor VME bus. The VME bus interface implements diagnostic functions (access to processor scan state, single stepping, breakpointing), VME and PE interrupts, and high speed I/O through a dual ported frame store.

- **Exception Handlers:** The new PE now provides a way to efficiently transfer control to state-preserving exception handlers. Exceptions can be induced unconditionally (an SVC), elicited by a standard ALU conditions (like overflow) or by operand type inconsistencies. In fact, we will use the SVC mechanism to provide dynamic linking to the resource management system, including frame and heap allocation/deallocation.

- **Temporary Registers:** The new PE permits state to be communicated between an instruction and its successor through a small set of temporary registers associated with the first stage of the ALU. There are eights sets of temporaries with three 72-bit registers per set, one set for each "logical thread" being interleaved by the eight stage pipeline. It is expected that temporaries will measurably improve the dynamic efficiency of compiled code.

Experience generating code for the wire-wrap prototype has suggested a number of minor enhancements to the processor datapath. For example, it is now possible to use the current tag as one of the arguments to the ALU—optimizing procedure linkage and **case** statements. We have also attempted to make the design more manufacturable by providing complete scan coverage of all internal processor state and parity protection for instruction memory, frame store, and the token queues. Figure 4.1 gives the block diagram of the processing element datapath. The processor is designed to run on a 100 nanosecond cycle time, yielding a sustained processing rate of 10 million tokens per second. Compiled code presently exhibits a dynamic average of 1.4 tokens per instruction. Thus, the processor pipe should deliver approximately 8 million instructions per second, any fraction of which may be floating point operations. The first set of processors will be equipped with 256K words (32 bits) of instruction memory, 256K words (72 bits) of frame store memory and 64K words (144 bits) of token queue memory. Both the instruction and frame store memories are upgradable to 1MWord.

The processing element detailed design was captured and extensively simulated on our Apollo/Mentor Graphics design systems. Both arrays were implemented in LSI Logic's 10K series of 1.5 micron channeless arrays, and packaged in 144 pin fine-pitch quad flat packs. The DATAPATH array comprises a little over 10,000 gates and implements a 9-bit slice of the datapath pipeline registers, temporaries, breakpoint registers, form token multiplexors, VME interface, and all static RAM parity generation and check. Each processor uses eight DATAPATH options. The PIU gate array is an ALU function unit specialized for tag and pointer manipulation. The PIU array comprises approximately 6,500 gates. Both arrays have been successfully prototyped in volumes sufficient for an initial build of five processing elements.

Design verification emphasized full-board gate-level simulation and timing verification, including all of the gate arrays. Simulation tests generally took the form of small handcoded dataflow graphs designed to test various aspects of the instruction processing mechanism. One or more initial tokens would be introduced into the pipeline, and then the simulation

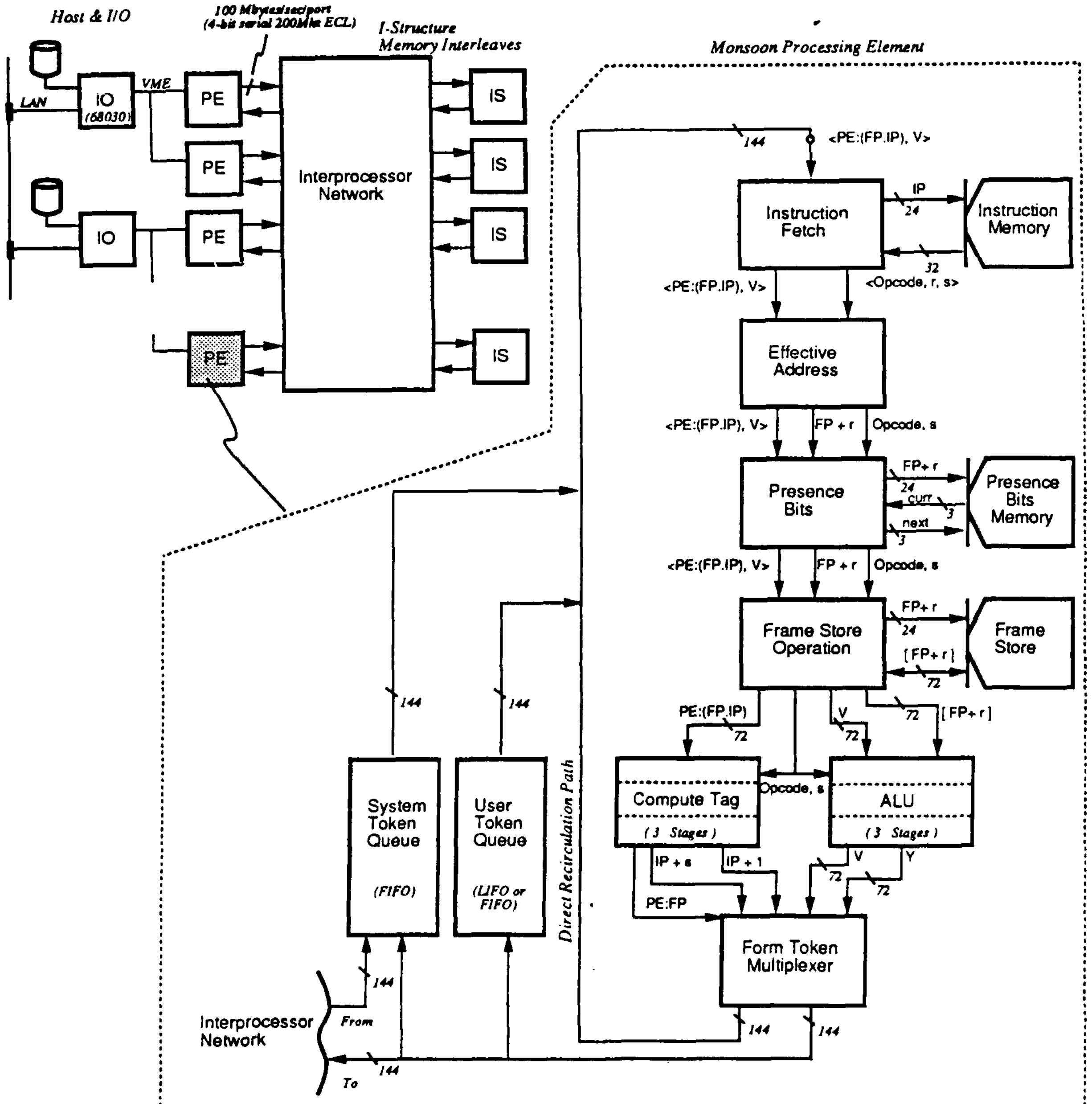


Figure 4.1: Block Diagram of Monsoon Processing Element Datapath

was allowed to "free-run" using the processing of the tokens themselves to generate the various test vectors. The simulated design was transferred to Motorola in the Fall of 1989. A 12-layer surface-mount circuit board was successfully routed by Motorola in February 1990, and assembly began on the first processor prototypes in May 1990. Simulation and verification of the processor still remains an area of intense activity. We have generated a set of code fragments with reference timing traces to aid in the hardware debugging and certification process.

4.9.3 The Interconnection Network for Monsoon

Andy Boughton, Christopher Joerg, Juan Ferrera, and Robert Lustberg have continued development of the network for Monsoon. This network is packet-switched and supports a bandwidth of 800 MBits/sec/port. The two primary components of the network are the Packet-switched Routing Chip (PaRC) and the Data Link Chip (DLC). These components were discussed in some detail in last year's progress report.

PaRC is a CMOS gate array designed by Chris Joerg that forms the basis of the Monsoon network. PaRC has 4 input ports and 4 output ports, each of which is 16 bits wide and has a maximum throughput of 800 MBits per second. Each input port has 4 buffers, each of which can hold one packet. PaRC has a sophisticated buffering and scheduling strategy which will allow an output port to transmit a packet whenever possible. PaRC uses a CRC code to detect errors on received packets. PaRC also allows a processor to get a fast acknowledgment that its message has been received. The mechanism for this is able to provide the acknowledgment without further burdening the network.

Chris Joerg has finished the design of PaRC and the generation of a complete set of test vectors. PaRC has 33,000 used gates and is capable of operating at 50 MHz. It has a low latency (100ns in light traffic), while making effective use of its bandwidth (90% utilization in heavy traffic). The set of test vectors allows the vendor to check for defects on newly fabricated chips.

PaRC has been fabricated in LSI Logic's 1.5 micron compacted array series, and working chips have been received. One of these chips has been placed on a simple test fixture constructed by Juan Ferrera. This setup was used to verify the timing of selected output signals.

The DLC is an ECL gate array that interfaces 16-bit wide PaRC ports to 4-bit wide interboard cables. Each DLC contains one data link transmitter and one data link receiver. Each of the 4 bits of the interboard data path is differentially driven at 200 MBit/sec.

Andy Boughton has completed the design of DLC and the generation of a set of test vectors. DLC has been fabricated in Motorola's Mosaic II ECL array series and working chips have been received.

Juan Ferrera and Robert Lustberg have designed and constructed a test fixture for the DLC chip. The fixture uses two DLCs to transmit test patterns over a 40' datalink cable. Tests with this fixture indicate that DLCs can be used to reliably interconnect network boards in different racks.

The first use of PaRC and DLC will be in the initial version of the Monsoon processor board. These boards are currently under construction. Each of these boards uses one PaRC chip and one DLC chip. These boards will allow PaRC and DLC to be more thoroughly tested in an operational environment.

Our industrial partner, Motorola, designed 4 input/4 output network boards using a PaRC and 4 DLCs. These boards which should be available early in 1991, will be used in the construction of 16-node Monsoon systems.

4.9.4 The I-structure Memory Board

Ken Steele completed his Master's thesis entitled *Implementation of an I-Structure Memory Controller* in February 1990. The design was kept simple to reduce cost and design time. In particular, the board does not perform local management of deferred continuation lists. Instead, the compiler allocates storage *in the frame* for one cell of a deferred list. When an I-fetch instruction is executed against an empty I-structure location, the I-structure board automatically responds with a token whose continuation is a small modification of the normal return continuation. The effect of this modified continuation is to thread the deferred list through all the frames from which the deferred I-fetches were issued.

Motorola Microcomputer Division in Tempe, Arizona is building an I-structure controller based on Steele's thesis design. Fabrication of the hardware is currently underway.

4.9.5 Caching for Monsoon

This past year, Derek Chiou has been investigating the possibility of caching on Monsoon. The prototype Monsoon uses static RAM for all of its memory at the present time. It would be desirable to use slower, cheaper dynamic RAM for future iterations of the processor. If the processor is to run at competitive speeds, using slower RAMs will require caching of some sort. Software has been developed that will produce an instruction trace from either MINT or the Monsoon prototype. Chiou also wrote a cache simulator which takes an instruction trace and collects cache data. Most of the serious data collection has been done for very small caches—generally around 32 words of fully associative cache. Results have been reasonably promising, with hit rates ranging from 33% to 84%. These results are very preliminary, however.

4.9.6 Completion of MINT, a Monsoon Simulator

Last year we reported the construction of MINT (Monsoon Interpreter), a simulator of the Monsoon instruction set. MINT has a variety of uses: debugging microcode for Monsoon, development and testing of Monsoon object code generators in the compiler, gathering of more detailed statistics than Monsoon itself is capable of gathering, etc.

This year, Andrew Shaw has extended MINT to simulate multiple Monsoon processor execution. The extension has a network simulation that models latency, but not contention. It is expected that contention will not be a significant problem as the performance of the Monsoon network is very high, and the references will be well distributed, as data is interleaved across processors. A multiple-processor run time system was implemented to distribute processes and to handle resource manager requests. Several experiments were run that indicate that Gita simulation was indeed an accurate predictor of performance in Monsoon. In addition, the capacity of the network was deemed sufficient to handle processors' memory requests and that the limiting factor in parallel execution is likely to be serialization induced by requests to the resource manager. In addition to multiple-processor simulation, an I-structure board simulation was added to MINT.

4.10 Other Activities

4.10.1 Optimal Interpreters for the Lambda-Calculus

Vinod Kathail completed his doctoral dissertation [156] in which he developed a new interpreter for the λ-calculus that is *optimal* in the theoretical sense defined by J.-J. Lévy [195], and gave proofs of its correctness and optimality.

The interpreter is based on a new graph representation for λ-expressions that permits sharing of not only subexpressions but also *contexts*, i.e., parts of an expression that are not complete subexpressions. This is in contrast to the commonly used representations of expressions which permit sharing of only subexpressions.

The interpreter is presented as a graph reduction system along with a normalizing strategy for applying the reduction rules. The set of rules includes a graph version of the β-rule of the λ-calculus as well as certain other rules, some of which are similar to the rules for handling environments in an environment-based interpreter for the λ-calculus. Some of the nice features of the interpreter are as follows: first, all the reduction rules are local constant-time operations on graphs; second, the reduction strategy for applying the rules is quite simple; and finally, the input to the interpreter as well as the output of the interpreter are "clean" representations of λ-terms. They do not contain various new types of nodes used by the interpreter. A version of the interpreter has been implemented on Lisp Machines.

To prove the correctness of the interpreter, the thesis develops two calculi, called λ_{fc} calculus and λ_f calculus. λ_{fc} calculus is essentially the term version of the graph reduction system underlying the interpreter. λ_f calculus is obtained from λ_{fc} calculus by removing certain types of terms and reduction rules that are not very useful for terms. The thesis shows the correspondence between the graph reduction system underlying the interpreter and λ_{fc} calculus, as well as correspondence between the two calculii and De Bruijn notation [87]. Although λ_f calculus was motivated by the interpreter, it may be of general interest because of the way it simulates changing of De Bruijn numbers.

The thesis also strengthens an earlier result of Barendregt, et al., that states that if λ-expressions are represented as trees, then there is no *recursive* (one-step) reduction strategy that is optimal. The extension proved in the thesis provides some justification for the basic assumption underlying the optimality criterion, i.e., the number of β-contractions performed in reducing an expression is a good measure of the cost of reducing the expression.

4.10.2 P-RISC

Madhu Sharma is investigating the design of a processor that is a concrete implementation of the P-RISC architecture. Most multithreaded architectures incur a large context-switching cost. The cost may be incurred either in *hardware*—when register space is provided for a large number of contexts, or in *time*—when contexts have to be swapped in and out of processor register files. The proposed design virtually eliminates both elements of context-switching cost. It caches contexts in multiple register sets in the processor, but manages to mask the context swapping cost using an additional port to the register file and deep, deterministic, instruction lookahead mechanism. The design is claimed to be only marginally more expensive than commercial RISC processors such as the SPARC.

A detailed simulator for the architecture has been developed. Statistics gathered for small handcoded programs run on the simulator indicate that the architecture does manage to mask context-switching cost and performs well under low or high parallelism.

We are now developing compiling techniques for the architecture, with two approaches pursued. The first is a *dataflow-graph* driven approach, wherein we start with a dataflow graph, sequentialize threads of the computation to obtain larger threads (whenever there is no gain in executing the threads in parallel), and arrive at a "control-flow graph", which is translated into P-RISC code. The second approach is the conventional "control-based" approach and will be used for compiling imperative languages.

4.10.3 Compiling Id for von Neumann Machines

Bradley Kuszmaul has been working on retargeting the Id compiler for stock hardware, such as conventional UNIX workstations. Starting with the existing dataflow graphs produced by the compiler, he translates these into parallel control-flow graphs based on the P-RISC abstract machine model. The major effort is then in analysis and transformations on the control-flow graph, including strictness analysis, subscript analysis, identification of threads, transformations to lengthen threads and reduce synchronizations, and peephole optimizations. Finally, these graphs are used to generate object code in the T language (T is a dialect of Scheme). The existing T compiler already has a very sophisticated code generator for a variety of stock machines (including register allocation, closure optimization, etc).

We expect to release a version of Id World using this new compiler by June 30, 1990. This implementation should substantially increase the availability of Id World to researchers who may not have access to Lisp machines or Monsoon dataflow machines. It should shed light on the differences in implementation requirements between nonstrict, lenient languages like Id, and nonstrict, lazy languages like Miranda.

4.10.4 Parallel Persistent Languages

Michael Heytens and Rishiyur Nikhil have made significant progress in their project to design and implement a parallel persistent language. The aim is to produce a system in which (a) the the user can declare, create and manipulate objects of arbitrary structured types; and (b) all such objects are automatically persistent. Such a system can be viewed as a synthesis of programming languages and databases.

Because of the rich object structure of the language, and because the structure can change significantly over time, high performance cannot be achieved using conventional database methods (detailed planning of data layouts on disks and scheduling of disk activity). Our approach is to use parallelism.

We have designed a kernel database language that is greatly inspired by Id, i.e., having fine-grained, implicit parallelism. In addition, in update transactions, each field can only be redefined once, as in I-structures; this allows update transactions also to be run in parallel. We have implemented a compiler that translates transactions into dataflow graphs and then into P-RISC code that is augmented with manager calls for disk I/O. As in Id and Monsoon, another objective is to mask disk latencies using parallelism.

We have designed a segmented, paged, distributed virtual heap for the persistent system. Pages of the virtual heap reside in files in each processing element, and are fetched on demand into page frames in each processing element. The protocols for page faults and flushing on transaction-commit have been designed, and the filespace occupied depends only on the heapspace in use (not the entire virtual heap address space). The files are partitioned across processing elements and are partitioned by type, allowing fast traversals of collections of objects of a given type. The files may also be indexed, allowing fast direct access to individual objects.

A prototype is being implemented, consisting of an ensemble of P-RISC emulators running on a network of Sun workstations. It is currently running a subset of the language, and we expect to support the full language by June 30, 1990. After this, we plan extensive evaluations, running numerous published and new benchmarks, and porting the system to a real multicomputer with parallel disks.

4.10.5 Bachelor's Theses and UROP Projects

Armando Fox (supervised by Jonathan Young) investigated possibilities for more efficient run-time storage managers. A number of traditional schemes were analyzed, with particular attention to projected performance and synchronization requirements for a dataflow architecture. A prototype of a storage manager was implemented which addressed the problems of global resource contention and long critical sections, and its performance was compared to the existing first-fit manager. Although substantial improvements were observed in a variety of cases, overhead associated with clearing out storage for reuse still accounted for a significant fraction of the deallocation latency. Increasing the efficiency of this operation and possibly implementing some sort of garbage collector remain topics for future investigation.

David Plass (supervised by Jonathan Young) implemented a parser generator which produces LALR shift-reduce tables in the dataflow language Id, for use in an Id parser. The algorithm employed avoids the creation of the LR(1) kernels by calculating LR(0) kernels and later adds LALR lookahead information. In addition, a general purpose parser shell was implemented which can be used in conjunction with output by a compiler to parse source language inputs. This work will help in our effort to write the Id compiler in Id.

Alejandro Caro (supervised by Jonathan Young) designed and implemented a symbolic debugger for the Id programming language on the Monsoon processor. The debugger allows the user to trace function calls and returns, to examine local variable bindings and loop variable bindings, and to examine the state of the machine in detail. Furthermore, the debugger allows the user to invoke these functions at the *source code level*, relieving the user from having to learn the intricacies of the processor and compilation schemes.

Glen Adams, (supervised by Greg Papadopoulos), extended the Lisp MINT simulator to model I-structure memory. An I-structure module simulator is capable of handling I-store, I-fetch, I-put, and I-take requests, as well as ordinary reads and writes. This was interfaced to the MINT system so that it is suitably initialized and driven by the queuing system. The design is extensible to model multiple I-structure units.

Mike Flaster (supervised by Rishiyur Nikhil) implemented a compiler for a small, nonstrict language that uses dependence analysis to convert a nonstrict program into a strict one. First, the compiler performs conventional def-use analysis, as well as subscript analysis for arrays in loops, using the Banerjee-Wolfe and GCD tests, augmented with some symbolic subscript-expression reduction. Then, the compiler performs loop-splitting, loop-reversal, loop-distribution, scalar expansion, induction-variable analysis, etc. to ensure that all dependencies are forward dependencies. The program can now be run with the parallelism that is best suited to the resources of a given machine.

UROP student Doug Stetson (supervised by Jonathan Young) implemented a compiler which translates a small subset of C into an Id program graph, the intermediate language of the Id compiler. The subset included assignments, conditionals and loops, but not procedure calls or pointers. The sequential semantics of C was enforced by artificial dataflow edges.

4.11 Publications

[1] Arvind and R. S. Nikhil. *A Dataflow Approach to General-purpose Parallel Computing.* Computation Structures Group Memo 302, MIT Laboratory for Computer Science, July 1989. Prepared for the proceedings commemorating the 25th Anniversary of Project MAC.

[2] Arvind, R. S. Nikhil, and K. Pingali. I-Structures: data structures for parallel computing. *ACM Transactions on Programming Languages and Systems*, 11(4), October 1989. Also Computation Structures Group Memo 269, March 1989.

[3] Arvind and R. S. Nikhil. Executing a program on the MIT tagged-token dataflow architecture. *IEEE Transactions on Computers*, 39(3), March 1990. Also Computation Structures Group Memo 271, June 1988.

[4] P. Hudak and P. Wadler, editors. *Report on the Programming Language Haskell, A Non-strict Purely Functional Language* (Version 1.0). Technology Report, Yale University, Department of Computer Science, YALEU/DCS/RR777, April 1990. Authors include Arvind, R. Nikhil, and J. Young.

[5] A. K. Iyengar. Parallel characteristics of sequence alignment algorithms. In *Proceedings of Supercomputing '89*, pages 304–313 Reno, NV, November 13-17, 1989. Also Computation Structures Group Memo 304, October 1989.

[6] B. C. Kuszmaul, and J. Fried. NAP (no ALU processor): the great communicator. *Journal of Parallel and Distributed Computing*, 8 (2):169-179, February 1990.

[7] R. S. Nikhil. New languages with implicit parallelism and heap storage are needed for parallel programming. In *Opportunities and Constraints of Parallel Computing*, pages 93–95 J.L.C. Sanz, editor, Springer-Verlag 1989. Position paper presented at a workshop, IBM Almaden, December 5-6, 1988.

[8] R.S. Nikhil and Arvind. *Id: a Language With Implicit Parallelism.* Computation Structures Group Memo 305, MIT Laboratory for Computer Science, February 1990.

[9] R.S. Nikhil and Arvind. *An Abstract Description of a Monsoon-like ETS Interpreter*, Computation Structures Group Memo 308, MIT Laboratory for Computer Science, April 1990.

[10] G.M. Papadopoulos. Program development and performance monitoring on the Monsoon dataflow multiprocessor. In *Instrumentation for Future Parallel Computing Systems*, pages 91–110, Frontier Series. Also Computation Structures Group Memo 303, October 1989.

[11] G.M. Papadopoulos and D.E. Culler. Monsoon: an explicit token store architecture. In *Proceedings of the 17th International Symposium on Computer Architecture*, Seattle, WA, May 1990. Also Computation Structures Group Memo 306, March 1990.

[12] J. Young. *Tests for Monsoon Instruction Subset 0.* Computation Structures Group Memo 307, MIT Laboratory for Computer Science, April 1990.

Theses Completed

[1] G. Adams. *Adding I-structure Simulation to MINT.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science. May 1990.

[2] A. Caro. *An Id Debugger.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science. May 1990.

[3] M. Flaster. *The Use of Dependence Analysis to Improve Compilation of Nonstrict Languages.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science. May 1990.

[4] A. Fox. *Parallel Storage Allocation in Id.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science. May 1990.

[5] D.B. Plass. *PIDGEN: a Parser Generator in Id.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science. May 1990.

[6] C. Joerg. *Design and Implementation of a Packet Switched Routing Chip.* Master's thesis, MIT Department of Electrical Engineering and Computer Science. May 1990.

[7] K.M. Steele. *Implementation of an I-structure Memory Controller.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, January 1990.

[8] G.L Wang. *A Synchronization Network Used for Debugging and Diagnostics on the Monsoon Multiprocessor System.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[9] V.K. Kathail. *Optimal Evaluators for Lambda-calculus Based Functional Languages.* PhD thesis, MIT Department of Electrical Engineering and Computer Science. May 1990.

Theses in Progress

[1] S. Aditya. *An Incremental Type Inference System for the Programming Language Id.* S.M. thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[2] D.S. Henry. *Analysis of Real-time Constraints in Stream Machine.* S.M. thesis, MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[3] J.E. Hicks, Jr. *Compiler Directed Storage Reclamation by Object Lifetime Analysis.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, Expected May 1991.

[4] B.C. Kuszmaul. *Compiling Data-flow Programs for Control-flow Computers.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected December 1990.

Talks

[1] Arvind. P-TAC: A parallel intermediate language. Lecture given at IFIP Conference on Functional Programming Languages and Computer Architecture, Imperial College, London, United Kingdom, September 12, 1989.

[2] Arvind. Project dataflow. Lecture given at MIT to DARPA visitors, October 26, 1989 and December 5, 1989.

[3] Arvind. Dataflow architectures and languages. Lecture given at Distinguished Lecturers Series, University of Southern California (November); Indian Institute of Science, Bangalore, India (December); Indian Institute of Technology, Madras, India (December); Indian Statistical Institute, Calcutta, India (December), 1989.

[4] Arvind. Supercomputing, parallelism and dataflow. Lecture given at Supercomputing '89 Conference, Teraflop Computer Panel, Reno, NV, Luncheon for Professor Emeriti at MIT, November 1989.

[5] Arvind. Multithreaded architectures: an internal workshop. Lecture given at MIT Laboratory for Computer Science, February 2, 1990.

[6] Arvind. Declarative programming and implicit parallelism. Lecture given at Parallel Computing Workshop, Ohio State University, Columbus, OH, March 21, 1990.

[7] Arvind. Architecture challenges to realizing massive parallelism. Lecture given at Parallel Computing Workshop, Ohio State University, Columbus, OH, March 23, 1990.

[8] Arvind. Explicitly programmed parallelism versus automatically generated parallelism. Panel discussion at Parallel Computing Workshop, Ohio State University, Columbus, OH, March 22, 1990.

[9] Arvind. Does programming parallel machines have to be painful? Lecture given at MIT-EECS Colloquium Series, April 9, 1990.

[10] Arvind. Advances in dataflow architectures: the Monsoon project. Lecture given at Activities in Computing and Artificial Intelligence, Workshop at MIT for NEC Corporation and Laboratory for Computer Science (May 7), School of Computer Science, Systems Seminar Series, McGill University, Montreal, Canada (April 20), IBM, Austin, Texas, (June 5), Lawrence Livermore Laboratory Sisal Workshop at Asilomar State Park, Pacific Grove, CA (June 7), IBM Research, Almaden, CA (June 8), 1990

[11] Arvind. Computer architecture: what it can do and what it cannot do for artificial intelligence. Lecture given at Conference on Knowledge Based Computer Systems: KBCS '89, Bombay, India, December 11-13, 1989.

[12] Arvind. Measuring fine-grain parallelism. Lecture given at Workshop on Advanced Programming Environments for Scientific Computation, Yale University, New Haven, CT, June 11, 1990.

[13] Arvind. The evolution of dataflow architectures from TTDA to P-RISC. Lecture given at Massive Parallelism: Hardware, Programming and Applications, Amalfi, Italy, October 9, 1989.

[14] S.A Brobst. Software pipelining and VLIW architectures. Lecture given at Boston University, Boston, MA, December 6, 1989.

[15] V.K. Kathail. An optimal interpreter for the λ-calculus. Lecture given at MIT Dataflow Workshop, MIT, Cambridge MA (November 1989); Bellaire Research Center, Shell Development Company, Houston, TX (March 1990); University of Chicago, Chicago, IL (April 1990); Cornell University, Ithaca NY (May 1990).

[16] V.K. Kathail. A λ-calculus interpreter that implements Lévy's optimal reductions. Lecture given at Semantics of Graph Reduction Workshop, Ecole Normale Supérieure, Paris, France, INRIA, Rocquencourt, France, April 1990.

[17] A.K. Iyengar. Lecture given at Parallel characteristics of sequence alignment algorithms. Supercomputing '89, Reno, NV, November 13-17, 1989.

[18] R.S. Nikhil. Compiling for P-RISC. Lecture given at Second Workshop on Languages and Compilers for Parallel Computing, University of Illinois, August 1989.

[19] R.S. Nikhil. The parallel programming language Id and its compilation for parallel machines. Lecture given at Workshop on Massive Parallelism, Amalfi, October 10, 1989.

[20] R.S. Nikhil. Compiling the parallel language Id for P-RISC, a dataflow RISC machine. Lecture given at McGill University, Montreal, October 27, 1989.

[21] R.S. Nikhil. Progress with P-RISC. Lecture given at Dataflow Workshop, MIT, November 3, 1989.

[22] R.S. Nikhil. LU decomposition in Id. Lecture # 3, Tutorial on "Programming in Id," Supercomputing '89, Reno, NV, November 13, 1989.

[23] R.S. Nikhil. The parallel language Id and its compilation for P-RISC, a dataflow architecture. Lecture given at University of Pennsylvania, Philadelphia, PA (December 1989); Boeing Computer Services (Helicopter support), Philadelphia, PA (December 1989); NECUSE Workshop on Parallel Computing, Amherst College, Amherst, (January 1990), (NECUSE - Northeast Consortium on Undergraduate Science Education).

[24] R.S. Nikhil. Another way to compile a non-strict language. Lecture given at Meeting of the IFIPS Working Group 2.8 on Functional Languages, Rome, Italy, March 26-29, 1990.

[25] R.S. Nikhil. Id update. Lecture given at Monsoon Software Cooperation Meeting, MIT, April 26, 1990.

[26] G.M. Papadopoulos. Scan path design. Lecture given at Motorola Microcomputer Division, Tempe, AZ, October 24, 1989.

[27] G.M. Papadopoulos. Dispelling the dataflow myth there is still a program counter...lots of program counters! Lecture given at MIT Seminar, October 30, 1989.

[28] G.M. Papadopoulos. Monsoon as a multithreaded multiprocessor. Lecture given at MIT Dataflow Workshop, November 2, 1989.

[29] G.M. Papadopoulos. The Olympic Games information system. Lecture given at Olympic Games Candidacy Committee, Athens, Greece, March 9, 1990.

[30] G. M. Papadopoulos. Monsoon: an explicit token store architecture. Lecture given at International Symposium on Computer Architecture, Seattle, WA, May 29, 1990.

[31] J. Young. Compiling Id for a real dataflow machine. Lecture given at Compass, Cambridge, MA, February 23, 1990.

Computer Architecture Group

Academic Staff

A. Agarwal	W. Dally
T. Knight	S. Ward

Research Staff

R. Jenez	G. Pratt
D. Kranz	J. Wolfe
M. Minsky	

Graduate Students

K. Abdalla	C. Metcalf
A. Ayers	J. Nguyen
C. Barclay	J. Morrison
D. Chaiken	D. Nussbaum
K. Ishii	J. Olsen
K. *Johnson*	J. Pezaris
J. Kubiatowicz	E. Puckett
K. Kurihara	C. Selvidge
B. H. Lim	A. Sherstinsky
G. Maa	R. Tessier
K. Mackenzie	S. Trowbridge

Undergraduate Students

D. Berger	N. Osgood
R. Dujari	I. Shen
B. Handley	S. Smoot
T. Nguyen	J. Vidal

Visitor

A. Altman

Support Staff

C. Winterton

5.1 Introduction

The 1989-90 academic year saw the birth of the Computer Architecture Group, an inter-laboratory conglomerate whose research projects and goals subsume those of several previously independent single-faculty groups. The CAG is still in its formative stages, and continues several projects previously reported elsewhere (including those of the former Real Time Systems Group and the Alewife project). Our goal over the next several years is to combine gratuitously different research projects to yield a coherent overall approach to architecture-related research.

Following sections report progress in several LCS projects which have continued in the new CAG venue.

5.2 Alewife

Agarwal, Kranz, and others continued their work on the Alewife machine project. The goal of the Alewife experiment is to demonstrate that a parallel computer system can be made both scalable and easily programmable. *Scalability* will be achieved through an architecture that allows the exploitation of locality. That is, for programs that display communication locality, scalable machines can offer proportionally better performance with more processing nodes. A program running on a parallel machine displays communication locality if the probability of communication with various nodes decreases with physical distance. *Programmability* will be achieved through automatic management of locality by a combination of hardware and software mechanisms.

As the experimental vehicle for this research, we are designing and implementing the parallel computer system, Alewife. Users will be able to write a large class of parallel applications on this machine without worrying about partitioning and placement of data or processes, while achieving speedups comparable to those available from carefully handcrafted programs.

The opposing goals of scalability and programmability are hard to achieve simultaneously. In conventional shared memory machines, all memory accesses incur the same cost. Thus, they can be programmed relatively easily, but bus or network bandwidth limitations hamper their scalability. Conventional message-passing multicomputers scale because they allow the exploitation of locality through the use of distributed local memory and direct networks, but the user has to explicitly partition and place data and processes, which makes it hard to program such machines. Perhaps it is this problem that prompted the following remark by Ken Kennedy in *Computing Research News:* "Contemporary parallel machines are architecturally diverse and have reasonably primitive programming systems that expose the details of the machine architecture to the user."

The Alewife machine has a distributed shared memory organization with a cost-effective, mesh network. Such an architecture allows the exploitation of locality, and is scalable. In our system, the hardware and software systems share the responsibility of enhancing locality to reduce bandwidth demands on the network, resulting in a system that is also easy to program.

Automatic locality management in Alewife is achieved by *reducing latency* of memory requests when possible, and by *latency tolerance* otherwise.

Several components in the Alewife system cooperate in automatic minimization of latency. Hardware-managed distributed caches significantly reduce the frequency of remote communications by automatically copying frequently-used data locally. However, maintaining cache coherence in scalable machines is a hard problem. We developed a novel method of enforcing distributed cache coherence called LimitLESS directories (Limited directories Locally Extended through Software Support). The LimitLESS directory works by maintaining a constant (two to four) number of pointers to shared copies of data in hardware, while trapping the processor when a directory overflow occurs and extending the specific directory entry into local memory. This simple scheme performs as well as the unscalable full map scheme, but its hardware overhead remains constant with machine size. The LimitLESS directory works well because the fixed set of pointers suffices for the common case of limited sharing, and because our processor's rapid context switching allows efficient trap handling for the rare cases.

The software run-time system of Alewife allows process and data partitioning, placement, and migration for improving locality. A new scheme, called *lazy futures*, partitions tasks at run-time, maximizing locality without compromising parallelism. A distributed tree scheduler dynamically assigns tasks to processors trying to balance the performance gains due to locality with the need for load balancing. The compiler assists by clever data and process decomposition and scheduling, and through hints to the run-time system.

When the system cannot obviate a remote memory request and is forced to incur the latency of the communication network, the Alewife processors attempt to tolerate this latency by rapidly scheduling a runnable process in place of the stalled task. Alewife can tolerate synchronization latencies as well through the same context switching mechanism. We designed a new processor architecture, called APRIL, that can rapidly switch between processes. (In our first-round implementation the switch will take 11 cycles.). The fast switching is achieved by caching a few (four in our implementation) process context frames on the processor to eliminate the overhead of unloading and restoring the process registers.

To assess the extent to which scalability and programmability can be achieved through automatic management of locality, we are building the Alewife machine and its software system and implementing several large symbolic and numeric applications. Alewife's rich performance instrumentation will provide an accurate evaluation of our ideas with real parallel applications on the initial 64 node machine, and experimentally observed communication locality profiles will also help us to forecast the extent to which such schemes scale to much larger systems.

Some of the salient developments in the Alewife machine project over the last year are outlined below.

5.2.1 The Alewife Machine Hardware Organization

The architecture of the Alewife machine has been defined and we obtained detailed performance estimates for a variety of applications through simulations. Figure 5.1 depicts the Alewife machine as a mesh connection of a set of processing nodes. (Figure 5.3 depicts the

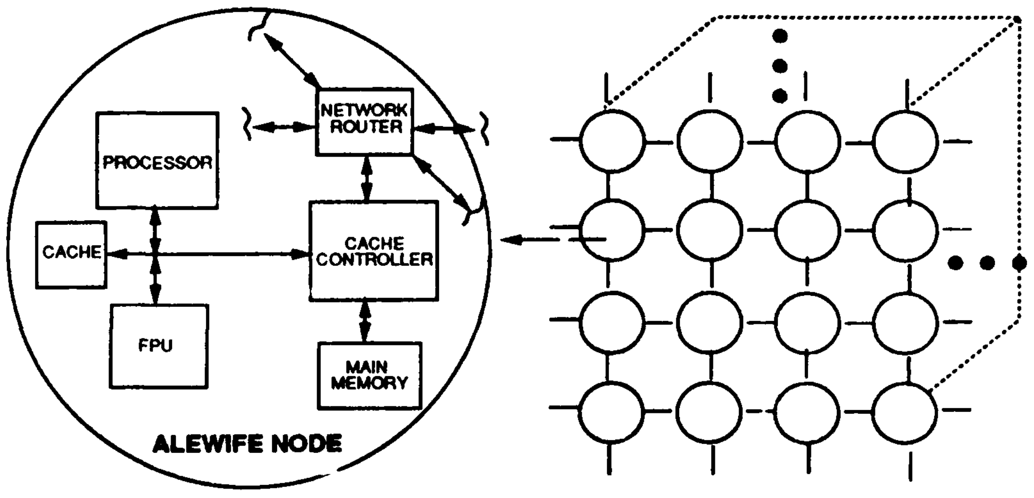


Figure 5.1: The Alewife Machine Hardware Organization

structure of the distributed LimitLESS directory.) Each Alewife node consists of a processor called APRIL, a cache, a portion of globally-shared distributed memory, a cache-memory-network controller, a floating-point coprocessor, and the network switch. Our current proposal is to build a modest 64-node (8 × 8) experimental system. Measurements from this machine will indicate the scalability potential of our ideas for a future larger-scale effort based on a three dimensional network called NuMesh (described in a later section).

5.2.2 ASIM: A Simulation System for Alewife

A simulator for the Alewife machine has been operational since June 1989. The simulator accurately models the processor, cache and memory, and the interconnection network. A compiler and run-time system are operational and produce code for the Alewife processor. ASIM simulates roughly 10,000 processor instructions per second on our SPARCserver 330.

ASIM has recently been augmented with many new features such as a floating-point coprocessor, support for special processor mechanisms including remote process invocation, and full-empty bit synchronization with support for arrays of full-empty bit data. ASIM also implements other directory coherence protocols and network structures to enable architectural comparisons. Several parallel applications have been written, compiled, and run on the ASIM simulator. The simulator has been heavily instrumented and yields a wide range of useful statistics including parallelism profiles, communication locality histograms, cache and

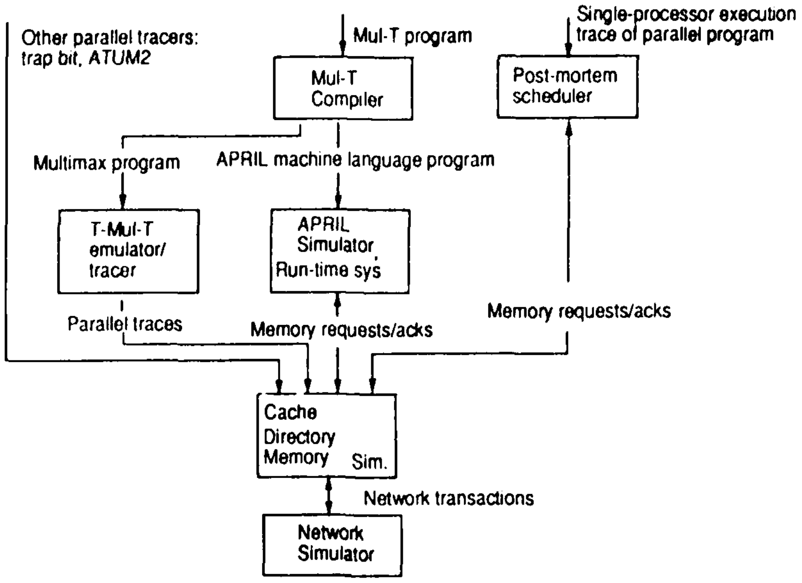


Figure 5.2: The Alewife Simulator System

network statistics, processor utilization, process length distributions, run lengths between synchronizations, etc. These execution profiles provide feedback to the programmer and help in parallel program optimization. The structure of ASIM is depicted in Figure 5.2.

5.2.3 Locality Management through Caches

We investigated the use of caches in providing efficient coherent shared memory. Caches copy frequently-used data in a fast local memory and can obviate repeat requests to remote memory. Furthermore, their operation is transparent to the programmer. David Chaiken has designed the cache coherence protocol for the LimitLESS directory scheme. Figure 5.3 depicts a LimitLESS directory node with two pointers implemented in hardware. The overflow pointers for datum X are stored in the local chunk of main memory. When the overflow pointers must be accessed the controller traps the processor, and the processor then proceeds to emulate a full-map protocol.

Chaiken and Kiyoshi estimated the performance of the LimitLESS scheme through simulations and showed that LimitLESS directories perform almost as well as the non-scalable full map directories. For example, for weather forecasting code, LimitLESS was only about 6% worse (with a 50 cycle processor overhead for software handling of a remote request that overflowed the directory and trapped the processor) than the full map protocol. Perhaps more importantly, this scheme allows us to experiment with the required amount of hardware support for cache coherence.

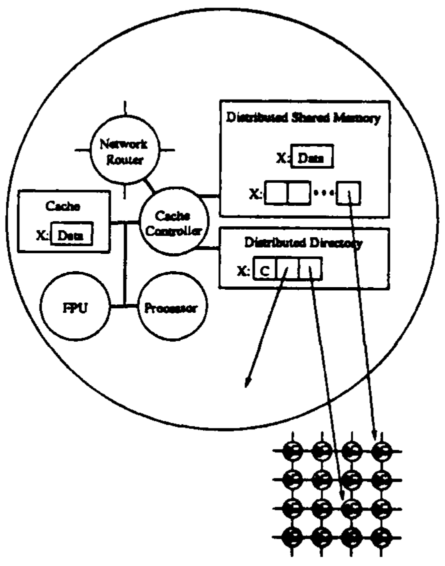


Figure 5.3: The LimitLESS Directory Scheme

5.2.4 The APRIL Processor

Processor architectures of the future must change to reflect the special needs of multiprocessors. Processors in multiprocessing environments must be able to tolerate long latencies arising from synchronizations and remote memory operations that defy locality optimizations. Additionally, LimitLESS directories and other functionality emulated through trap software require efficient trap handling. Lim, Kubiatowicz, and others have designed a new processor architecture called APRIL that meets these requirements. APRIL is a multithreaded VLSI processor with high single-thread performance. A multithreaded processor mitigates the negative effects of long communication and synchronization delays in multiprocessors by overlapping these delays with computation from other processes. High single thread performance ensures reasonable behavior when the application lacks parallelism.

The left half of Figure 5.4 depicts the user-visible processor state comprising four sets of general purpose registers, and four sets of Program Counter (PC) chains and Processor State Registers (PSR). The PC chain represents the instruction addresses corresponding to a thread, and the PSR holds various pieces of process-specific state. Each register set, together with a single PC-chain and PSR, is conceptually grouped into a single entity called a *task frame*. Four such task frames are implemented in the first version of APRIL. Task switching happens in 11 cycles in this implementation. Only one task frame is active at a given time and is designated by a current frame pointer (FP). All register accesses are made to the active register set and instructions are fetched using the active PC-chain.

Kubiatowicz developed a simple memory reference based interface between the APRIL processor and the Alewife cache/memory controller. Using a control word associated with each memory reference, various types of synchronization or communication types are synthesized by the controller. This interface allows a simple implementation of the processor and the

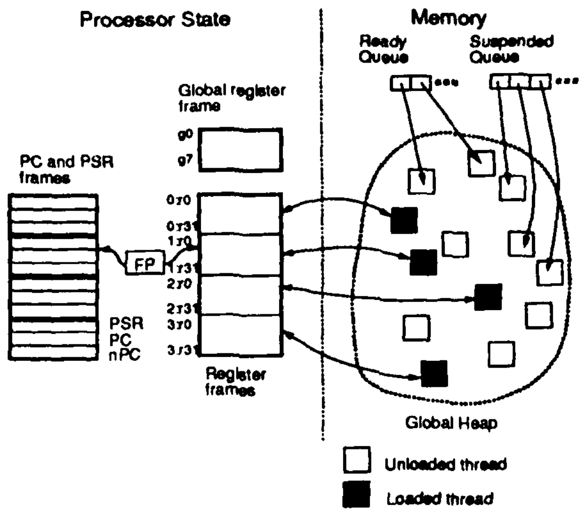


Figure 5.4: The APRIL Processor Architecture

controller by enabling graceful trapping into software for handling complicated rare conditions for which dedicated hardware cannot be justified. The interface has been implemented in the Alewife simulator.

The implementation of the APRIL processor explores a new form of collaboration between industry and university. We are modifying LSI Logic's SPARC processor to avoid duplicating the engineering effort in designing components not directly relevant to our research. Besides other obvious benefits such as industry support, this approach affords the opportunity to ride the industrial technology curve as new technologies evolve, and allows rapid transfer of new ideas from university to industry, and the ability to impact evolving industry standards. The implementation, called MIT-SPARCLE, is a joint effort with LSI Logic and SUN Microsystems. LSI Logic's standard cell and gate array SPARCs allow high level modifications of the processor. Currently our modifications have been incorporated into the SPARC gate-array design at LSI and we are now moving into the design verification phase.

We also evaluated the performance of multithreaded processors in large scale multiprocessors using an analytical model. For processor parameters derived from APRIL's SPARC-based implementation, the study showed that multithreaded processors such as APRIL can achieve over 80% efficiency with just three threads with a 10 cycle memory delay in a 3D mesh network with base average latency of 55 cycles.

5.2.5 The Interconnection Network

The Alewife controller uses a simple message-based interface with the network. Various forms of shared memory coherence models are maintained by the controller via messages to other nodes. We plan to use the Caltech Mesh Routing Chips for our switch nodes in the two dimensional 64 processor system. A much more aggressive three dimensional interconnect, called NuMesh, is being developed in the Computer Architecture Group in a collaborative effort of Steve Ward, Tom Knight, Bill Dally, and Anant Agarwal. A future larger scale implementation of Alewife will use this network.

5.2.6 The Run-time System

Nussbaum is working on a run-time system that optimizes locality of memory referencing through clever dynamic scheduling methods. He implemented a tree scheduler together with lazy task creation to manage parallelism and locality. Early results obtained on our simulator indicate substantial benefits over other scheduling methods that are oblivious of the physical nature of the interconnection network. The tree-scheduler is currently being augmented with support for multithreaded processor operation.

5.2.7 The Compiler System

David Kranz has retargeted the Orbit optimizing compiler for the APRIL processor. He also implemented a novel method of dynamic process partitioning called *Lazy Futures*. The Lazy Futures method virtually obviates the overhead associated with task creation and deletion when tasks run on the same processor on which they were created. For example, with Lazy Futures, the sequential version of an application runs at roughly the same speed as a single processor execution of a parallel version of the same application.

Maa and Johnson have been working on compiler methods for enhancing locality in parallel programs by statically partitioning code and allocating data structures based on minimization of nonlocal memory references. In Alewife, the task of the compiler is easier than in other contemporary machines for several reasons. First, we assume the program is explicitly parallel (by using program-level constructs such as futures). Second, the compiler-based static methods can be integrated with our sophisticated run-time mechanisms. For example, compilers often lack accurate execution time profiles to enable a good static schedule; instead, our compiler can provide hints to the run-time system. Finally, Alewife's shared memory organization, dynamic scheduling, and caches largely relieve the burden of micro-managing data access and synchronization.

Several linguistic extensions to our Mul-T programming language are necessary to support static scheduling of processes and remote memory allocation (e.g., future-on and make-vector-on). These facilities have been implemented in the Orbit compiler and the ASIM simulator by Lim and Kranz.

We are examining several large data-parallel applications as test cases for our compiler work. The lexical matching phase of the SUMMIT Speech recognition system (developed by the Spoken Language Systems Group) has been parallelized and ported to our system by Johnson. The particle-in-cell (PIC) program, written originally by Olaf Lubek, has also been parallelized and ported to Mul-T by Maa and others.

Structurally, PIC consists of steps of operations which produce successive intermediate matrices. The operations typically read from a few locations of the input matrices and compute some locations for the output matrices, which is used as the input to some subsequent operations. Fine-grain architectures synchronize on the individual elements instead of on entire matrices, thereby exposing the producer-consumer parallelism. Because of the relatively trivial amount of work involved in each of these operations (usually a sum of few products), the overhead of scheduling and synchronization becomes prohibitively high. To reduce these and communication (i.e., improving locality) costs, it will be beneficial to merge some of the operations together: by merging an upstream operation with a downstream one, we can potentially make the intermediate matrix element local and eliminate BOTH the communication and synchronization costs. By merging two peer operations which share some of their input matrix element accesses, we again reduce the amount of network traffic. The merging increases the task granularity, thus reducing the number of tasks which need to be scheduled at run-time and the total scheduling overhead. The desired granularity can be adjusted to match the system size (i.e., number of processors) and the actual cost structure of the run-time and hardware to insulate the programmer from details of the run-time environment. Moreover, once most of the accesses to a matrix element are from a particular task, it starts to make sense allocating the corresponding matrix elements and tasks to the same processor node.

5.2.8 Applications

Johnson, Wu, Chan, and others have been involved in developing several large parallel applications, including Speech Recognition, Particle in Cell, and solutions of partial differential equations. Performance results on real parallel applications are essential to effectively evaluate our ideas on automatic locality management.

5.3 NuMesh

Work by Agarwal, Dally, Knight, Pratt, Ward, and others continues the NuMesh project whose conception was reported last year. The NuMesh constitutes the first CAG-wide research effort, and is consequently interesting from the dual standpoints of social as well as computer engineering.

The basis for the NuMesh is the recognition that extant technologies for interconnecting digital subsystems provide topological flexibility only at substantial performance cost. Communication schemes designed to suppress the impact of physical distances among modules from the design sp.ce of the system architect, such as backplane buses and hypercubes, typically reduce best-case communication times to accommodate worst-case distances for reasons of simplicity. At the lowest level, drive levels and times must anticipate the worst-case loading allowed by their interconnect rules. This leads to a fundamental tension between flexibility as to physical and topological layout of a system's components and the degree of optimization afforded their communication: if a pin on a chip is capable of driving a foot of PC trace and ten inputs, resource (hardware, time, energy) are wasted when it is used to drive a single input on an adjacent chip.

One reaction to such inefficiency is to custom-engineer the drivers and receivers on each signal line to reflect the physical properties of the associated electrical conductor. This approach is followed to some extent in very high performance systems, in which tedious analog-domain transmission line analysis techniques are applied to each critical signal conductor in a complicated system. While some such detailed optimization can be justified in high-end system designs, it necessarily stops far short of the ad hoc optimization, say, of individual output

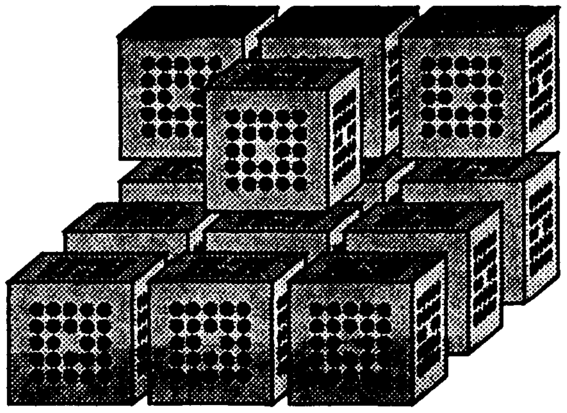


Figure 5.5: Schematic View of a NuMesh

pads. Moreover, the prohibitive cost of this methodology rules out its use in cost-effective systems.

The NuMesh project explores an alternative reconciliation of communication flexibility with performance: rather than forcing a universal communication technology to conform to arbitrary network topologies, the NuMesh forces higher-level system designs to conform to a rigidly specified universal communication topology.

5.3.1 The NuMesh Abstraction

The major goal of the NuMesh project is the definition of and support for a standard communication and interconnect standard for modules of arbitrarily complex digital systems. Abstractly, a NuMesh consists of an arbitrary number of nodes which (partially) populate a three dimensional mesh, as diagrammed schematically in Figure 5.5.

Each node in the mesh constitutes a digital subsystem which communicates directly with each of its six orthogonal neighbors via a number of dedicated signal lines. Each signal conductor is unidirectional, having exactly one receiver and one driver (on adjacent nodes). Moreover, the physical characteristics of each conductor are rigidly fixed. This mechanical and topological rigidity allows bandwidth and latency parameters of communication between adjacent nodes to be highly optimized; our goal is on the order of one GHz for each signal line in the eventual standard.

Each node combines a common communication substrate with the application-specific logic it adds to the system. The communication portion of a node includes a very high speed clocked finite state machine which is programmed (in RAM) to mediate local communications, as well as drivers and receivers and data paths which interconnect them. The set of Communication FSMs—CFSMs-occupying each node of a NuMesh operate synchronously at the basic communication clock rate (eventually 1 GHz). They are typically pre-programmed to follow a fixed, systolic communication pattern whose detailed choreography is produced by whatever compiler or CAD tool also dictates the selection and layout of the modules themselves.

The general idea is that each module's communications FSM be programmed to follow a periodic pattern of interactions with neighbors. Although the interactions may vary among processors, the periods will be identical. If module A transfers a word to its right-hand neighbor B on clock 37 of each period, then A's FSM will be programmed to drive its lines to B on that clock, while B will be programmed to load in data from A. By appropriate design of transition tables, arbitrary systolic communication patterns may be implemented among processors. In some cases, words loaded by a module are destined to be read subsequently by that modules DSP; in other cases, they are routed (typically on the next clock) to another neighbor without DSP intervention or even awareness.

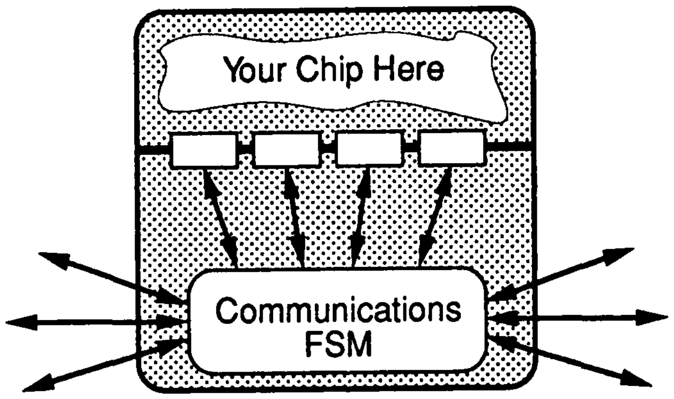


Figure 5.6: NuMesh Node

Communication in a NuMesh thus typically follows a static, precompiled, systolic pattern in which every node can in theory send data to each of its neighbors during every cycle of the communication clock. Certain of the data received at a node will be routed by that node's CFSM to the processor, or other application-specific logic housed in the node. However, we expect that the bulk of the data received at a node will be forwarded, as through traffic, to another neighbor during a subsequent cycle by the CFSM. Our goal is to make this forwarding efficient, allowing remote communications to be implemented as multiple store-and-forward steps with performance that compares favorably with more conventional remote interconnect schemes such as buses. It is our presumption of a preponderance of remote traffic that motivates our design goal that each CFSM support an aggregate communication bandwidth much higher than that required by the processor (or other application-specific logic) supported by the node containing it.

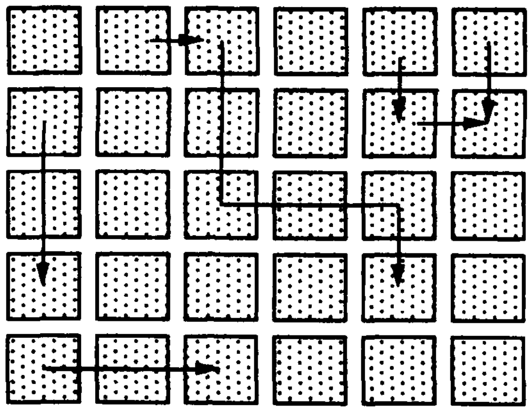


Figure 5.7: Snapshot of NuMesh Communications

Figure 5.7 depicts a typical pattern of communications at some point during a computation on a 2D NuMesh configuration. Note that communications between adjacent nodes—indicated by short arrows—take place during the single clock cycle which this snapshot depicts. Remote communications, depicted by longer arrows, require several cycles; while the first step along each such path is taken during the cycle depicted, successive steps will occupy subsequent cycles.

Certain algorithms may benefit from flow control and other synchronization measures in their underlying communications. These might be superimposed on the primitive (branch-free) communication mechanism by software convention, allowing certain data words to contain control information. Moreover, CFSM interpretation of selected data words provides a basis for dynamic routing of selected data messages by the CFSM alone, considerably enhancing the flexibility of the NuMesh communications substrate. The approach to such dynamics suggested by the NuMesh structure involves preallocating (at compile time) certain cycles of the periodic communication schedule to dynamically routed packets, identifying at that time the cycles at which each CFSM will interpret incoming data as a potential message header. Such data will dictate, perhaps after a several-clock latency (to accommodate decision logic), the routing of the message body to be transferred on subsequent clock cycles. Hardware support for such CFSM decisions is currently a subject of active study.

5.3.2 NuMesh Prototype Hardware

During the 1989-90 academic year, a feasibility study for NuMesh ideas was started with the design and prototyping of primitive NuMesh nodes built using off-the-shelf parts. Work by Mackenzie, Jenez, Abdalla, Olsen, and others has led to a 4-neighbor, 2D NuMesh node fabricated on a pair of stacked printed circuit cards. Each node carries a TI TMS320C30 digital signal processor chip along with local memory, leading to an aggregate peak performance of 33 MFlops/node. The nodes are designed to plug together to form desktop prototype NuMesh arrays, providing an early vehicle for software development and exploration of candidate applications.

Although we expect these prototype nodes to be useful both in evaluation of NuMesh ideas and in pursuit of practical applications, the expedient technology forces many serious compromises. In addition to clear size and cost disadvantages, the prototype nodes are clocked at about 30 MHz and thus impose bandwidth limitations which are more than an order of magnitude below our performance goals. Moreover, their design supports only static communication models: the CFSM is not equipped to interpret data and make routing decisions without processor intervention. We expect all of these restrictions to be relaxed somewhat in a subsequent prototype exploiting custom silicon.

Currently, the prototype NuMesh hardware is interfaced to a NuBus-based Macintosh which provides host system services and software support; interfaces to alternative workstations are contemplated in the future. The nodes may be bootstrapped from the Macintosh, using a scheme developed by Dujari, using only the communication and CFSM logic on each node. Thus, a NuMesh configuration can be mapped and explored by the host independently of the processors on each node, allowing the communication substrate to be configured and diagnosed as a separable subsystem.

Two nodes are operational, and 15 more are under construction; thus we anticipate an operational four-by-four NuMesh array during the summer. CD-quality analog I/O is provided by an interface developed by Handley, allowing application of these prototypes to speech recognition and related signal processing applications.

5.3.3 NuMesh Prototype Software

Work by Fetterman, Jenez, Metcalfe, and Trowbridge addresses the development of an initial software environment for our prototype NuMesh nodes. The goal is to translate source code from a static block diagram language (similar to Consort and related real time languages) to a fully-configured multiprocessor NuMesh, generating in the process both DSP code and CFSM programming.

The compilation problem is complicated considerably, even in the context of our limited source language, by the need to make compile-time decisions regarding the distribution of subcomputations among nodes. In general, such decisions cannot be made at a machine-independent level, since they involve reconciling computation times with communication overhead. In order to confine processor dependencies to an isolated code generator module, we provide for dyadic communications between machine-independent compiler modules faced with distribution and scheduling decisions and the processor-specific code generator modules capable of estimating (or bounding) run-times. These interactions make use of a low level processor-independent intermediate code in a data structure elaborated repeatedly by several disparate compiler modules.

Trowbridge completed a Macintosh-based frontend for the translator, and Fetterman is completing a TMS320C30 code generator which produces both optimized pipeline code and timing information from machine-independent input fragments. Metcalfe completed a UNIX-based simulator which allows experiments to be conducted on an approximate model of NuMesh computations.

Our goal is to complete this initial support software during the summer and demonstrate it, using our prototype NuMesh nodes, on one or two modest but representative applications. Speech recognition, which (via Zue's group) provided the initial stimulus for the NuMesh, remains the primary source for candidate application code. A second application area of interest is finite element modeling, currently being explored by Arthur Altman who is visiting CAG from Texas Instruments.

5.4 Publications

[1] A. Agarwal. *Analysis of Cache Performance for Operating Systems and Multiprogramming.* Kluwer Academic Publishers, 1989.

[2] A. Agarwal and M. Cherian. Adaptive backoff synchronization techniques. In *Proceedings of the 16 th Annual International Symposium on Computer Architecture*, IEEE, June 1989.

[3] A. Agarwal and A. Gupta. *Temporal, Processor, and Spatial Locality in Multiprocessor Memory References.* Plenum Press, 1989. S. Tewksbury, editor. Also appears as MIT VLSI Memo, 1989.

[4] A. Agarwal, M. Horowitz, and J. Hennessy. An analytical cache model. *ACM Transactions on Computer Systems,* May 1989.

[5] A. Agarwal and M. Huffman. Blocking: exploiting spatial locality for trace compaction. In *Proceedings of ACM SIGMETRICS 1990*, May 1990.

[6] A. Agarwal, B-H. Lim, D. Kranz, and J. Kubiatowicz. APRIL: A processor architecture for multiprocessing. In *Proceedings of the 17 th Annual International Symposium on Computer Architecture,* June 1990.

[7] A. Agarwal. *Limits to Network Performance.* MIT Laboratory for Computer Science, November 1989. Also MIT VLSI Memo 1989. Submitted for publication.

[8] A. Agarwal. *A Locality-based Multiprocessor Cache Interference Model.* Submitted for publication. Also MIT VLSI Memo 1989.

[9] A. Agarwal. *Performance Tradeoffs in Multithreaded Processors.* VLSI Memo 89-566, MIT Laboratory for Computer Science, September 1989. Submitted for publication.

[10] D. Chaiken, C. Fields, K. Kurihara, and A. Agarwal. Directory-based cache-coherence in large-scale multiprocessors. *IEEE Computer*, June 1990.

[11] D. Kranz, R. Halstead, and E. Mohr. Mul-T: a high-performance parallel Lisp. In *Proceedings of SIGPLAN '89, Symposium on Programming Languages Design and Implementation*, June 1989.

[12] E. Mohr, D. A. Kranz, and R. H. Halstead. Lazy task creation: a technique for increasing the granularity of parallel tasks. In *Proceedings of Symposium on Lisp and Functional Programming*, June 1990. To appear.

[13] D. Nussbaum, I. Vuong, and A. Agarwal. Modeling a circuit-switched multiprocessor interconnect. In *Proceedings of ACM SIGMETRICS 1990*, May 1990.

[14] S. Ward and R. Halstead. *Computation Structures.* MIT Press, 1989.

Theses Completed

[1] K. Abdalla. *A TMS320C30-based Digital Signal Processing Board for a Prototype NuMesh Module.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[2] C. Barclay. *Parallel Programming Language Constructs for L.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[3] K. Ishii. *Caching and Memory Reference Patterns in Parallel Lisp Applications.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[4] T. King. *Test of Metastability in Synchronizer Circuits.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, September 1989.

[5] J. Morrison. *Scalable Multiprocessor Architecture Using Cartesian Network-relative Addressing.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, September 1989.

[6] T. Nguyen. *Analysis of Spatial-locality-based Trace Compaction Methods.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[7] N. Osgood. *Automatic and Near-exhaustive Derivation of Machine-dependent Optimizations.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science.

[8] M. Powell. *A Simulator for Multiprocessor Implementations of L.* Master's thesis, MIT Department of Electrical Engineering and Computer Science.

[9] I. Shen. *The Utility of Block Transfer Mechanisms in the Alewife Multiprocessor.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[10] S. Smoot. *JUNE-BUG: A Parallel Debugger for the Alewife Machine.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[11] S. Trowbridge. *A Programming Environment for the NuMesh Computer.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[12] J. Vidal. *Parallelizing Compiler for Matrix Expressions.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

Theses in Progress

[1] A. Ayers. PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected September 1991.

[2] D. Chaiken. *Cache Coherence Protocols for Large-scale Multiprocessors.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected October 1990.

[3] M. Cherian. *A Study of Backoff Barrier Synchronization.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1989.

[4] M. Fetterman. Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected September 1990.

[5] K. Kurihara. *Performance Evaluation of Large-scale Cache-coherent Multiprocessors.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected July 1990.

[6] B-H. Lim. *Waiting Algorithms for Synchronization in Large-scale Multiprocessors.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected October 1990.

[7] G. Maa. PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected December 1991.

[8] D. Nussbaum. *Run-time Locality Enhancement on Scalable Multiprocessors.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected June 1991.

[9] C. Selvidge. *Compile-time Latency Management: Exploiting Program Predictability to Achieve Memory Latency Tolerance.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected June 1990.

Talks

[1] A. Agarwal. The MIT Alewife machine. Lecture given at UC Berkeley, Stanford University, DEC Systems Research Center, Distinguished Lecture Series, University of Rhode Island, June 1989.

[2] A. Agarwal. The MIT Alewife machine: exploiting locality in a large-scale multiprocessor. Lecture given at Encore Computer, Distinguished Lecture Series, April 1990.

[3] A. Agarwal. Exploiting locality for scalability. Lecture given at IEEE Workshop on Interconnections within Digital Systems, Santa Fe, NM, May 1990.

[4] A. Agarwal. The L word. Lecture given at the MIT Laboratory for Computer Science, Annual Meeting, Harwichport, MA, June 1989.

[5] A. Agarwal and B-H. Lim. Exploiting locality for scalability in the MIT Alewife machine. Lecture given at the Workshop on Shared Memory Multiprocessors, Seattle, WA, May 1990.

[6] A. Agarwal. Multiprocessor address tracing: the agony and the ecstasy. Lecture given at the Workshop on Multiprocessor Performance Evaluation, Seattle, WA, May 1990.

[7] A. Agarwal. APRIL: a processor architecture for multiprocessing. Lecture given at the Symposium on Computer Architecture, Seattle, WA (May 1990); Workshop on Multithreaded Processors, MIT Laboratory for Computer Science (February 1990).

[8] A. Agarwal. Blocking: exploiting spatial locality for trace compaction. Lecture given at ACM SIGMETRICS 1990, Boulder, CO, May 1990.

[9] K. Johnson. Exploiting concurrency in Voyager's lexical access subsystem. Lecture given to the Spoken Language Systems Group, MIT Laboratory for Computer Science, December 1989.

[10] D. Kranz. Mul-T: a high-performance parallel Lisp. Lecture given at SIGPLAN'89, Portland, OR; USA/Japan Workshop on Parallel Lisp, Sendai, Japan, June 1989.

[11] J. Kubiatowicz. Hardware-software tradeoffs in the MIT Alewife machine. Lecture given at the Workshop on Shared Memory Multiprocessors, Seattle, WA, May 1990.

[12] B. H. Lim. A run-time system for the MIT Alewife machine. Lecture given at DEC Systems Research Center, June 1990.

Information Mechanics

Research Staff

N. Margolus	T. Toffoli, Group Leader

Graduate Students

M. Biafore	H. Hrgovčić
D. Harnanan	M. Smith

Support Staff

A. Wiseman

Visitors

B. Chopard	F. Commoner

6.1 1989-90 Activity

Most of our activity last year was centered on the CAM-8 project. This is a high performance cellular automata multiprocessor that will allow one to explore a new band of the computational spectrum.

The task is immense, as it includes: (a) functional design at the system level, at the board level, and at the chip level; (b) interfacing with a host (the SPARCstation-1) that was just introduced, and whose technical specifications (S-bus, PROM monitor, etc.) are just beginning to be available; (c) host software at different levels (registers, device drivers, CAM-8 control, user interface, display and analysis, and experiment management); issues related to massive scalability (three dimensional interconnection, power distribution, and heat removal; clock distribution; synchronization; error detection and recovery; and (d) circuit and behavioral simulation at the chip level (we are designing custom gate arrays) and at the multi-chip, multi-board level; etc.

We also became project managers, personnel managers, computer system managers, etc. After appropriate searches, we hired a full-time research programmer and two circuit-design consultants. We purchased eight SPARCstations and configured them for a number of different tasks. Following successful negotiations, we obtained from VTI the free use of their VLSI design and simulation software, and behavioral simulation software from Veralog. Installing and learning to use all of these tools is an enormous job.

We also started tentative probes in view of some form of industrial partnership in the CAM-8 project.

Of course, we had to continue doing a fair amount of responding to real time interrupts, such as conferences, meetings, papers due, reports, proposals, refereeing, etc.

In particular, we wrote a new three-year proposal, *Information Mechanics*, for NSF, and we participated in a proposal for an NSF Science and Technology Center on *Quantum-effect Physics, Electronics, and Computation Structures*, with Professors Hank Smith and Dimitri Antoniadis.

Besides work by the research staff [285][286][284][219][282][283][281][220], a fair amount of theoretical research was conducted by our Ph.D. students [53][148][272][271] and our visitors [69][78][73][72]. For his Master's thesis, David Harnanan did a remarkable job at designing and simulating the interface between CAM-8 and its host.

6.2 Publications

[1] M. Biafore. *Two-dimensional Universal Automaton with Electron-like Tokens.* Technical Report MIT/LCS/TM-429, MIT Laboratory for Computer Science, June 1990.

[2] L. Chalfin and B. S. Tsirelson. *Quantum/classical correspondence in the light of Bell's inequalities.* Technical Report MIT/LCS/TM/420, MIT Laboratory for Computer Science, 1990.

[3] B. Chopard. Cellular automata model for the diffusion equation. *J. Stat. Phys.*, 1990. Submitted for publication.

[4] B. Chopard. A cellular automata model of large scale moving objects. *J. Phys. A*, 1990. To appear.

[5] F. Commoner. Synchronization graphs for quantum computation. To appear.

[6] H. Hrgovčić *The local interpretation of quantum mechanics.* PhD thesis, MIT, 1990. In progress.

[7] N. Margolus. Parallel quantum computation. In *Complexity, Entropy, and the Physics of Information,* Addison-Wesley, 1990. To appear.

[8] N. Margolus and T. Toffoli. Cellular automata machines. In *Lattice Gas Methods for Partial Differential Equations*, pages 219–248, Addison-Wesley, 1990.

[9] M. Smith. *Nuclear Fusion Through Dimensional Confinement.* Technical Memo MIT/LCS-/TM-409, MIT Laboratory for Computer Science, August 1989.

[10] M. Smith. Representation of geometrical and topological quantities in cellular automata. *Physica D*, 1990. To appear.

[11] T. Toffoli. *Analytical Mechanics from Statistics: $T = dS/dE$ Holds for Almost Any System.* Technical Memo MIT/LCS/TM-407, MIT Laboratory for Computer Science, August 1989.

[12] T. Toffoli. Frontiers in computing. In *Information Processing,* Volume 1, North-Holland, 1989.

[13] T. Toffoli. How cheap can mechanics' first principles be? In *Complexity, Entropy, and the Physics of Information,* Addison-Wesley, 1990. To appear.

[14] T. Toffoli. Cellular automata. In *Encyclopedia of Physics*, VCH, 1990.

[15] T. Toffoli and N. Margolus. Invertible cellular automata: a review. *Physica D*, 1990. To appear.

[16] T. Toffoli and N. Margolus. Programmable matter. *Physica D.*, 1990. To appear.

Large Scale Parallel Systems

Academic Staff

W. E. Weihl, Group Leader

Graduate Students

W. Hsieh	S. Perl
Q. Huang	E. Waldin
A. Joseph	P. Wang
E. Kolodner	

Undergraduate Student

B. Spiers

Support Staff

G. Sherman

Visitor

H. Bal

7.1 Introduction

The Large Scale Parallel Software Group began in the fall of 1989. Its research is focused on software issues involved in making effective use of large scale multiprocessors. Most group members are working on two projects in this area: designing a parallel programming language to support writing portable programs for MIMD parallel architectures; and developing algorithms, and system and language support for writing fault-tolerant parallel programs. As part of the first project to design a parallel programming language, we developed techniques for implementing concurrent data structures that scale well and make effective use of local caches. In addition, some group members are doing research related to transaction processing and to distributed systems. Our research in these areas is described below.

7.2 Portable Parallel Software

We are designing a new programming language to support the implementation and execution of parallel programs. The language is intended to run on large scale MIMD multiprocessors. It is clear that there will be many such machines, of both the shared-memory and message-passing varieties, available in the not-too-distant future. To make use of the machines, and to evaluate their potential, we need new programming languages that allow them to be used effectively.

Our language is intended to be used for a wide range of applications, including both symbolic and numeric computations and programs that have side effects. We expect it to run on both shared-memory multiprocessors and message-passing multiprocessors. As a secondary goal, we would like it to run on networked collections of uni- and multi- processors. In addition, we would like programs to be portable, with good performance, across a broad range of MIMD architectures.

There are a number of issues that must be addressed to build efficient parallel programs, including scheduling, choice of an appropriate granularity for processes and data, placement and migration of processes and data, effective use of caches, and synchronization. Some systems hide most or all of these issues from the programmer. Others expose the details of the specific architecture. The first approach gives portability, since the programmer writes little architecture-dependent code. However, it is difficult to get good performance using the first approach. The second approach can give good performance, but lacks portability and can also be difficult to use.

To achieve portability and performance, we are designing mechanisms that allow the programmer to control issues such as scheduling, granularity, and placement in an architecture-independent manner. For example, we developed a scheduling method that allows programs to be written so that the grain size of processes adapts to the number of processors available and on processor loads. The programmer supplies information that allows the system to make informed decisions about which tasks are most usefully split into concurrent subtasks. We are experimenting with mechanisms to achieve similar results for the other issues.

To judge performance, we plan to implement our language on one or more real multiprocessors. In addition, however, we will use simulators to allow us to experiment with a wider range of architectural features and to better understand how effective our approach is at providing portability. To avoid building separate simulators for each target architecture, we have designed and are building a retargetable simulator, which is easily tailored to simulate different architectures.

7.3 Fault-tolerance

Advances in hardware technology have made it plausible to construct a parallel computer with hundreds of thousands or even millions of processors sometime in the next decade. Programming such machines will in itself be difficult. We believe, however, that an equally difficult challenge will be that of maintaining a working machine in the presence of failures.

Various studies have shown the mean time between failures (MTBF) for current machines to be on the order of a few weeks or months. A machine 100 times as big would have an MTBF on the order of a few hours or days. Without mechanisms to limit the amount of work lost when a failure occurs, the range of problems that can be solved effectively on such large machines will be limited to those that take relatively little time to compute. However, existing techniques for achieving reliable operation in the face of failures do not scale to large numbers of processors, or to machines with very large numbers of hardware components.

The goal of this research is to develop software techniques that provide a low overhead and scalable approach to fault-tolerance. (Our efforts are directed at tolerating hardware and communication failures; we are not addressing the problem of tolerating design and programming errors.) Adequately addressing the challenge of ensuring reliable operation for large scale multiprocessors requires improvements at all levels, from low level hardware components through application software. We are focusing on software techniques because they provide more flexibility in tailoring the level of redundancy and reliability to the needs of applications, and also make it easier to reconfigure machines dynamically when failures occur. In addition, by focusing initially on the requirements of applications and of system software, we will gain a better understanding of what should be provided by lower levels.

Scalability is important to make effective use of large scale machines. In addition, low overhead is essential to allow fine-grained parallel computations. In particular, if the fault-tolerance mechanisms significantly increased the cost of communication, then programmers would be forced to use a larger grain size to amortize the overhead.

Fault-tolerance methods typically exhibit a tradeoff between overhead during normal operation and the cost of recovering from a failure. Fault-tolerance mechanisms for parallel systems can be loosely classified as either optimistic or pessimistic. Purely optimistic methods record checkpointed process states and other information asynchronously, without forcing delays at particular points in the computation. After a failure, such methods must rollback one or more processes to earlier states to find a consistent state of the system. In contrast, pessimistic methods prevent the failure of one process from causing other processes to rollback. Optimistic methods attempt to minimize overhead, but may take a long time to recover. Pessimistic methods impose significant delays, but can recover quickly.

Pessimistic methods do not meet our goal of low overhead. Purely optimistic methods, however, do not scale well: they perform very poorly in large scale multiprocessors. There are two reasons for this. First, recovery from a single failure could involve every processor in the machine, and could take long enough that another failure would be very likely to occur before recovery from the first failure completed. As a result, useful forward progress could occur very slowly. Second, given a fixed probability of failure for each individual component in each step, we can show that the expected expansion factor in execution time using a purely optimistic method grows exponentially with the number of components, even ignoring the fact that recovery takes longer for larger machines. (The expansion factor is the ratio of the actual execution time for a computation in the presence of failures to the amount of time required for the computation when no failures occur.) Thus, for any fixed component reliability, a purely optimistic method breaks down for very large machines.

The best approach seems to be an optimistic approach that limits the amount of work required to recover from a failure; such an approach will sometimes require a process to wait while recovery information is recorded, but the delays should be shorter and less frequent than with purely pessimistic methods. Over the past year, we have designed a "limited optimistic" method that represents an intermediate point in the tradeoff between overhead during normal operation and recovery time: the overhead is more than purely optimistic methods, but less than pessimistic ones, while the recovery time is less than optimistic methods, but more than pessimistic ones. The technique limits the length of a chain of processes that can be involved in a cascaded rollback after a failure; this limits the recovery time, and prevents the blowup in the expansion factor that occurs with purely optimistic methods.

In general, it is difficult to achieve both low overhead and fast recovery. As a way of circumventing this tradeoff and achieving both very low overhead and fast recovery, Anthony Joseph has been developing techniques for *application-specific fault-tolerance*. The idea is to take advantage of properties of applications to reduce the communication and coordination required for fault-tolerance, to reduce the work lost when a failure occurs, and to reduce the time taken to recover from a failure. Our initial experiments indicate that application-specific methods can perform significantly better for some applications than any application-independent method.

Application-specific methods have the disadvantage that they require additional programming that could add significantly to the complexities of writing a parallel program. In our experience so far, however, the added complexity is actually quite small. For example, Anthony Joseph has been looking at numerical algorithms. For some algorithms, such as asynchronous iterative methods [36], processes can take checkpoints and recovery from failures without any coordination or communication. This purely local method eliminates almost all of the overhead of more general methods, and also loses much less work when a failure occurs. Simulations show that it performs significantly better than other methods [154]. Other examples have been studied by Henri Bal, who wrote several fault-tolerant parallel programs in Argus [197] while visiting the Large Scale Parallel Software Group in the fall of 1989. His experience is described in [32]. We are currently studying other applications, both to understand what language and system support might simplify the task of writing fault-tolerant parallel programs, and to understand the potential performance gain of application-specific methods over application-independent methods.

7.4 Concurrent Data Structures

Many parallel programs are organized as a collection of processes accessing shared data structures, through which processes communicate and coordinate with each other. This organization is common in programs written for shared-memory architectures; it also applies to many programs written for message-passing architectures, particularly those written in an object-oriented style. Over the last year, we developed several new algorithms for concurrent data structures. Our new algorithms reduce contention among the processes sharing the data structure; as a result, our algorithms provide significantly better performance than existing algorithms, particularly when many processes are using the data structure concurrently. As part of our work on concurrent data structures, we also developed new techniques for software cache management, as well as new implementation techniques for synchronization primitives such as read-write locks. The sections below describe this work in more detail.

7.4.1 Concurrent B-trees

B-trees are widely used in databases and other applications (e.g., file systems) that require fast access to data stored on disk; they are also useful in parallel programs that require fast access to shared data, even if the data is stored in main memory. A number of algorithms for concurrent access to B-trees have been developed (e.g., see [37][187][228][180][258]); all of the previously existing algorithms require processes to lock nodes of the tree in such a way that a process updating a node blocks all other processes that attempt to access the node while the update is being performed. We developed a new concurrent B-tree algorithm that eliminates the need for a process traversing down the tree to block while an update is propagated up the tree [290]. The net result is that contention at non-leaf nodes of the tree is virtually eliminated.

The basis of our new algorithm is an abstraction that is similar to coherent shared memory, but provides a weaker semantics; we call this abstraction *multi-version memory.* Multi-version memory is used in the algorithm for all non-leaf nodes of the B-tree, while coherent shared memory is used for the leaves. Multi-version memory weakens the semantics of ordinary shared memory by allowing a process reading data to be given an old version of the data. (For example, it might simply use the version in its cache.) While this weaker semantics is not as generally useful as that provided by coherent shared memory, it turns out to be adequate for our B-tree algorithms. We describe multi-version memory in more detail in Section 7.4.3 below.

As described below, multi-version memory can be implemented so that a process reading data can use a local cached copy, and almost never needs to be delayed while waiting for messages that update or invalidate caches. As a result, our new concurrent B-tree algorithm should continue to work well in large scale parallel systems in which the number of processors sharing the tree is large or the communication delay between processors (or between processors and the global memory for a shared-memory system) is large relative to the speed of local computation.

Relatively little work has been done to study the performance of concurrent B-tree algorithms. In a Master's thesis due to be completed in September 1990, Paul Wang has been evaluating the performance of our new algorithm and comparing it to that of other B-tree algorithms. Four different B-tree algorithms have been implemented in Concurrent Aggregates (CA), a language developed by Andrew Chien [71]. CA allows a collection of objects to be viewed as a single logical object, thus making it easy to encapsulate cache management algorithms for multi-version memory and for coherent shared memory. The resulting programs have been run on a simulator for a message-passing multiprocessor. The simulations are being used to study how the performance of each algorithm varies with the number of worker processes accessing the B-tree, with the length of the delay for interprocessor messages, and with the implementations of multi-version memory and coherent shared memory. Results to date indicate that our new algorithm provides significantly lower latency and higher throughput than the other algorithms.

7.4.2 Concurrent Priority Queues

The priority queue is a fundamental data structure that has been used in a large variety of parallel algorithms, such as multiprocessor scheduling and parallel best-first search of state-space graphs. In these algorithms, each process performs an access-think cycle, in which the access is one of the *insert*, *extract*, *decrease key*, and *delete* operations on the priority queue. For his Master's thesis, Qin Huang has been designing and evaluating algorithms for parallel priority queues.

Algorithms that ensure a strict semantics for the *extract* operation, which extracts the minimum element from a priority queue, exhibit limited speedup because of the bottleneck caused by the synchronization needed to ensure that the element returned by the *extract* operation is the least element. To avoid this bottleneck, it is necessary to relax the specification of the *extract* operation so that it is not required to return the least element. The performance of the application may be better if the element returned is close to the minimum, but in many applications the correctness of the final result of the computation does not depend on which element is returned.

We designed two different algorithms that relax the specification of *extract*. One is based on Fibonacci heaps, the asymptotically most efficient data structure for sequential priority queues. This algorithm keeps a cache of a small number of the most promising elements of the queue; a process executing *extract* selects a random element from the cache, thus avoiding a single serial bottleneck. Bottlenecks in accessing the root list of the Fibonacci heap are avoided by dividing it into a number of separate sections, each of which can be accessed independently.

The second approach, which we have called a concurrent priority pool, is based on a combination of a concurrent B-tree algorithm and concurrent pools [212][176]. A concurrent pool is a distributed data structure for managing a pool of resources; it is divided into a number of segments that can be accessed independently to add and remove elements from the pool. A concurrent priority pool is like a concurrent B-tree, except that concurrent pools (extended to handle splitting and merging) are used for the leaves. The *extract* operation attempts to remove an element from the leftmost leaf by accessing one of the segments in that leaf. Since there are multiple segments, several *extract* operations can proceed concurrently without any interference.

Both algorithms allow the quality of the element returned by *extract* to be controlled, in the first case by controlling the size of the cache of promising elements and in the second case by controlling the number of segments in each leaf of the tree. Higher speedups can be obtained by relaxing the quality of the returned elements. Thus, they permit an application to tune its use of a concurrent priority queue to balance the quality of the elements returned by *extract* against the contention found in accessing the queue.

Both algorithms have been implemented in Mul-T [177], along with a concurrent binary heap algorithm developed at the University of Texas at Austin [251]. Experiments have been performed on an Encore Multimax. Results to date show that the two new algorithms provide essentially linear speedup for up to 10 processors (as many as the machine currently provides), while the concurrent binary heap is limited by contention at the root of the heap to a speedup of about 4. Further experiments are planned, both on an Encore machine with more processors and on simulators, to see how well the algorithms perform with more processors and to understand how the quality of the elements returned by *extract* is affected by the number of processors concurrently accessing the queue. Applications of parallel priority queues, such as a parallel single-source shortest path algorithm and a parallel solution to the traveling salesman problem, are also being tested to evaluate the performance of the different parallel priority queue algorithms for particular applications.

7.4.3 Software Cache Management

In many parallel applications, caching is vital for achieving high performance. For example, the root of a B-tree is visited by every operation on the tree, and is rarely updated. If only a single copy of the root is maintained (either in global memory in a shared-memory architecture, or in the memory of a single processor in a message-passing architecture), the root is likely to be a serious limiting factor in performance. Caching improves performance in part by allowing data to be accessed in local memory, thus avoiding the delay involved in accessing a remote memory, and in part by replicating data so that many processes can read it in parallel. Coherent shared memory, however, constrains caches to be managed so that the read and write operations appear to be atomic.[1] These constraints require synchronization between readers and writers, and also require communication to update or invalidate caches after a processor has written to memory.

An alternative to maintaining cache coherence is to delegate the management of cached copies to the application. The advantage of this approach is that the cache management algorithm can be tailored to the needs of the application. The disadvantage is that programs could become significantly more complex. However, we believe that this complexity can be managed by encapsulating cache management algorithms in the implementations of abstract data types.

Multi-version memory is an example of a memory-like abstraction that can take advantage of local caches, but requires less synchronization and communication than coherent shared memory. Abstractly, the state of a multi-version object at any point in time is a sequence of versions. The first version in the sequence is the *initial* version, and the last version is the *current* version. Writers update the object by extending the sequence with additional

[1]There are a number of subtly different correctness criteria that have been used for coherent shared memory, including *sequential consistency* [183] and *linearizability* [144]. We will take linearizability as our definition of correctness.

versions (thus changing the current version), and readers read the object by choosing and reading some version. The specification allows a reader to read *any* version, not just the current version. This nondeterminism allows us to implement a multi-version memory so that readers can run in parallel with writers. In addition, propagation of an update to cached copies can be done lazily in the background; thus, the invalidation (or update) load for a heavily shared object can be spread out over time, and processes need not wait for propagation to complete.

An application that can use a multi-version memory will probably perform better if readers obtain fairly recent versions, but the specification of a multi-version memory requires the application to be prepared for a reader to obtain an arbitrary version. Additional constraints could be added to the specification. For example, we might require a process reading a multi-version object to choose a version that is no older than any other version already used by the process. Alternatively, we could require it to choose one of the k most recent versions. Such constraints are not needed for the applications we have studied so far. However, they might be useful for some applications, and should have little negative impact on performance.

Brad Spiers has been studying the performance of different implementations of multi-version memory and comparing their performance to that of coherent shared memory. He has been using a simple simulator, built by Wilson Hsieh, for a shared-memory multiprocessor. The simulations show, as expected, that multi-version memory and coherent shared memory take approximately the same amount of time for workloads in which either all processes only read or in which all processes only write. Simulations are currently being done for mixed workloads, in which processes both read and write. Our expectation is that multi-version memory should provide better performance than coherent shared memory in these cases. The simulations will help us understand the magnitude of the performance difference, and how it is affected by variations in the number of processors and in the time required for accesses to global memory. They will also help us understand how often a reader obtains a version other than the current version; this information will help in predicting the utility of multi-version memory for various applications.

As discussed in Section 7.4.1, multi-version memory can be used in concurrent B-tree algorithms to reduce contention and mask network latency. In general, it can be used in any application in which a process reading data can tolerate reading an old version of the data. For example, in asynchronous iterative relaxation algorithms, a process computing a new value for one point can use values for other points from any previous iteration. Similarly, in a branch-and-bound search algorithm implemented using several worker processes, it is not necessary for each process to know the most recent value of the global bound representing the best solution found so far.

The advantages of multi-version memory over coherent shared memory suggest that it may be fruitful to view cache management as an application-level replication problem, where both the semantics of the shared data and the algorithm used to manage caches can be designed as part of the application. Such an approach fits naturally into an object-oriented programming style based on inventing application-specific abstract data types, such as that advocated by Liskov and Guttag [203]. Future research will consider what primitives should be provided by the hardware and by the programming language to support this kind of software cache management.

7.4.4 Distributed Locking

Read-write locks are used in many concurrent data structures. The traditional method for implementing read-write locks, and many other synchronization primitives, is by use of a monitor [147]. A monitor protects synchronization data with a mutual exclusion lock; any process that wishes to access the synchronization data (e.g., to acquire a read lock, or to release a write lock) must first acquire the mutual exclusion lock. To implement a read-write lock, the monitor might contain a count of the number of readers and a flag indicating whether a writer is currently active.

Because the data in a monitor can be accessed by only one process at a time, the monitor itself can be a serious source of contention. In many cases, this contention is logically unnecessary. For example, processes acquiring read locks do not need to synchronize with each other; however, processes acquiring write locks do need to synchronize with each other and with processes acquiring read locks. A distributed locking strategy can be used to avoid this unnecessary contention.

We designed a distributed locking strategy based on the idea of caching locks as well as data. If a lock is already cached in a particular mode, an acquisition request for the lock in that mode can be satisfied without communicating with other processes. Otherwise, the acquisition request is handled by ensuring that no process has the lock cached in a conflicting mode. Similar strategies have been used in distributed file systems and in the Vaxcluster lock manager.

Caching locks allows readers to run without interfering with each other. However, the cost of acquiring a write lock can be quite high, since it involves synchronizing with all readers that have the lock cached. We are experimenting with other distributed locking strategies, based on software combining trees, that reduce the cost of write locks with only minimal increase in the cost of read locks. Wilson Hsieh is using a combination of analysis and simulation to understand the tradeoffs between the cost of read locks and the cost of write locks.

For the simulations, Wilson built a simulator for a shared-memory multiprocessor that allows the user to write parallel programs in Scheme, using special functions to access shared objects. The simulator simulates a parallel machine with an arbitrary number of processors, each of which has some local memory. There is also a global memory, which must be accessed over the network. The simulator does not simulate network contention (e.g., tree saturation of the network), but does simulate contention on objects in the global memory. Each object in global memory is treated as if it has infinite memory to queue memory requests. The relative costs of network accesses and local operations can be varied by the user. The simulator is also being used by other students for other experiments.

7.5 Performance Specifications

A program has a *performance bug* when some cost of its execution—e.g., response time, throughput, or resource utilization—is higher than it is supposed to be. *Performance debugging* is the process of detecting, locating, and eliminating performance bugs. Building programs that perform well typically involves a combination of designing for good performance from the start and doing performance debugging when problems show up after the program is implemented and running. In many situations, the implementor of a program does not—and should not be expected to—understand the implementation of the underlying system at all levels. Most application programmers trying to make their programs run fast will not have detailed knowledge about the implementation of the underlying operating system, yet application performance is affected by operating system performance, which itself is affected by the manner in which the application uses the operating system.

It has long been common software engineering practice to provide functional specifications for program modules so that a client of a module need not know how it is implemented in order to use it. A functional specification tells the client exactly what functionality he can expect from the module, and a functional bug exists when a module fails to meet its specification. In the same way, a performance bug is a failure to meet a performance specification. Yet, precise performance specifications are even rarer than precise functional specifications. At best, performance specifications tend to be vaguely stated, and at worst, they exist only in implementors' and users' minds as some expectation of execution cost. Consequently, mistaken performance expectations are difficult to detect and correct, and when performance bugs do exist, the person doing the debugging must have extensive knowledge of the implementation of large parts of the system.

For her Ph.D. thesis, Sharon Perl is developing a method for writing performance specifications of concurrent systems. The goal is to make it easier for programmers to tell when a program has a performance bug and to determine which part of the program is at fault. The thesis of this research is that having explicit and precise performance specifications makes it easier to build programs that perform well and, furthermore, that writing such specifications is a practical endeavor. The latter claim will be demonstrated empirically, by developing performance specifications for a real software system. The work focuses on specifying response times, although we hope the work will extend to other aspects of performance. The former claim will be supported through argument and reports of experience with the specifications that are developed.

The context for this work is concurrent systems, ranging from multitasking uniprocessor systems to small-to-medium scale multiprocessors to distributed systems. We are not considering highly parallel systems or applications (e.g., Connection Machines), though the results may be of some use in that domain. We assume that the software to be specified has a modular structure, i.e., that it is decomposable into modules or subsystems for which performance specifications may be written. An underlying assumption of this work is that performance bugs exist that are not often detected, or that are caught fairly long after they appear. This is a reasonable assumption in our experience, particularly for systems where good performance is desirable but is not the primary concern of the implementors (e.g., systems in research environments).

This work should have the following major contributions:

- An approach to writing performance specifications that identifies a structure and content of specifications appropriate for their use as documentation and in performance debugging;
- A methodology for using performance specifications for performance debugging;

- Actual performance specifications for a significant part of a distributed file system;
- A methodology for developing performance specifications; and
- An initial version of a language or notation for writing performance specifications.

The method will be demonstrated by specifying the performance of significant parts of a replicated distributed file system.

7.6 Atomic Garbage Collection

Transactions, used in database and distributed systems, provide fault-tolerance by masking failures that occur while they are running. Automatic storage management, used in modern programming languages, enhances reliability by preventing errors due to explicit deallocation (e.g., dangling references and storage leaks). A uniform storage model simplifies programming by eliminating the distinction between accessing temporary storage and permanent storage. We call storage that is managed automatically using garbage collection, manipulated using atomic transactions, and accessed uniformly, a *stable heap*. For his Ph.D. thesis, Elliot Kolodner is designing and prototyping algorithms for managing a large stable heap. Stable heap management will make it easier to write reliable programs and could be useful in programming languages for reliable distributed computing [196][90], programming languages with persistent storage [10][18], and object oriented database systems [66][211][293][299].

A programmer views a stable heap as a single-level store. The heap is stored on disk, with pages brought into primary memory as needed. The heap has a designated set of root objects. Not all objects are treated as stable; instead the set of roots is partitioned into stable and volatile subsets, and an object is stable if and only if it is accessible from one of the stable roots

Computations run as atomic transactions [132]. Objects are created and modified by transactions; an object becomes stable if a pointer to it is placed in an existing stable object by a transaction that commits. A garbage collector reclaims an object's storage automatically when the object is no longer accessible from any of the roots. In addition, a recovery system ensures that stable objects survive failures: modifications performed by aborted transactions are undone, while modifications performed by committed transactions are guaranteed to survive both system crashes and media failures.

Automatic storage management for a stable heap is complicated by the fact that a garbage collector typically moves and modifies objects. Collectors move objects to improve paging performance; they modify objects to reduce the amount of additional storage needed by the collector itself. The movement and modification of objects during garbage collection requires coordination with the recovery system. A collection algorithm for a stable heap that solves these problems is called an *atomic garbage collector*.

Many applications, such as computer-aided design, computer-aided software engineering, and office information systems, require large amounts of storage, timely responses for transactions, and high availability. Our earlier research produced an atomic garbage collector, and recovery system suitable for small stable heaps [175][174]. In that work, the atomic collector is based on a stop-the-world copying collector and the recovery system uses a shadowing technique that requires a traversal of the stable object graph after a crash. These algorithms are not suitable for applications with large heaps: a stop-the-world garbage collector may delay transactions arbitrarily, and the traversal of the stable state slows recovery after a failure, reducing availability.

The goal of our current research is to design an integrated atomic garbage collector and recovery system appropriate for a large stable heap on stock hardware. The collector must be incremental and atomic: it cannot attempt to collect the whole heap in one pause and it must interact correctly with the recovery system. The time for recovery must be independent of heap size and adjustable to be arbitrarily short using checkpoints.

During the past year, we completed the design of the algorithms. Our approach divides the heap into volatile and stable areas. Objects are created in the volatile area. After an object becomes stable it is moved to the stable area at an appropriate time. The volatile area can be collected using incremental or generational collection. We designed an incremental atomic garbage collector based on the ideas of Ellis, Li, and Appel [105] to collect the stable areas. The collector interacts correctly with the recovery system, and it writes enough information to the log (maintained by the recovery system) to avoid redoing the entire collection after a crash.

We have worked out the details for two approaches to recovery: (1) write-ahead logging with update-in-place, and (2) a variation of intentions lists. In both approaches the information associated with stable objects maintained for active transactions is kept in the volatile area separate from the objects themselves. This allows recovery without a traversal of the object graph and lowers the space overhead for stable objects. For both approaches, the location of a committed object version in the stable area does not change; it is updated in place. This avoids creating garbage in the stable area and lowers the space overhead for recovery.

While designing the new recovery system, we found a bug in the design of the current Argus recovery system. Because of the bug, a transaction might commit before values for all the objects accessible from the object that it modified have been written to stable storage. Thus, after a crash the effects of some committed transactions might not be recoverable. In a design note [173], we describe new recovery algorithms that overcome this bug.

Currently, we are implementing a prototype of a stable heap to show the feasibility of our design. The current implementation of Argus [201] serves as the basis for the prototype; we are replacing its storage management and recovery algorithms.

7.7 Communication in Heterogeneous Distributed Systems

To send a message in a heterogeneous distributed system, it may be necessary for the sender or receiver of the message, or both, to translate the data in the message to or from its internal format. The typical method for handling this problem is to define a single standard representation for each data type to be used in messages. If a module's internal data uses a different representation, it must translate the data to or from the standard to send or receive it. If two communicating modules use the same internal representation that differs from the standard representation, this scheme results in unnecessary translation by both modules.

An alternative approach is to define multiple representations for each data type to be used for communication. By choosing a representation for the data in a message that matches its internal representation, the sender of the message can avoid excess translation.

For his Ph.D. thesis, Earl Waldin is designing a communication system that allows multiple representations of data types to be used in messages. Some existing systems allow multiple representations to be defined for transmitting simple scalar data such as integers and characters. Multiple representations can also be useful for more complex data structures, and in general for arbitrary abstract data types. Allowing multiple representations has two main advantages: it can reduce the amount of translation involved in sending messages, and it allows the system to evolve easily by adding or changing the representations of data used in messages.

A prototype is being designed as part of the Mercury system. In Mercury, a distributed program is composed of distinct *entities* that communicate with one another, where an entity resides entirely at a single node. Entities are implemented by different mechanisms in different programming languages. A receiving entity provides one or more *ports*, which are procedures that can be called remotely from other entities. The arguments and results of port invocations are passed by value; there is no way for references to data in one entity to be sent to other entities. The transmitted values belong to value-spaces (*Vspaces*), which are similar to types in programming languages. The term Vspace is used to emphasize that data is passed by value, and that the data in a message has no operations that can be performed on it. Abstract Vspaces are user-defined Vspaces; their representations are defined in terms of other abstract and built-in Vspaces.

Over the past year, algorithms for negotiating representations to be used in messages have been designed. Current work involves the design of the interface description language (IDL) and annotations for its use with the C programming language. The IDL is used to give a language-independent description of a program interface in terms of ports and Vspaces. The language-specific annotations are used to describe how a given program uses that interface. For example, the annotations may describe mappings between types in the program and Vspaces in the interface. The annotations also describe interactions between the program and the communications substrate that implements the Mercury protocols.

As part of defining the IDL, we studied the feasibility of adding interface types to Mercury along with a mechanism for dynamically checking their use. Loosely stated, an interface type plays a similar role to that of an abstract data type in a programming language. More specifically, an interface is a collection of ports (i.e., operations) through which a client may access a subsystem. A subsystem in turn consists of one or more entities that together provide a service. Each port in the interface is provided by a single entity; invocation of a port results in a remote procedure call to the corresponding entity. To a client, then, a subsystem appears as a (distributed) object that responds to invocations of the ports in its interface. The behavior of a subsystem is determined by an interface to which we assign a type. A given subsystem corresponds to an instance of a type.

Type checking the use of interfaces is more difficult than type checking the use of abstract data types in a program. The greatest difficulties arise because subsystem interfaces are constructed dynamically and because subsystems may evolve. As an example of the first, consider the following subsystem: Entity E_A constructs an instance A of interface I_A containing only ports that it provides and then exports this instance (e.g., by putting it in some catalog). Entity E_B imports A (e.g., by looking it up in the catalog) and constructs an instance B of interface I_B by combining some of the ports of A with ports that it provides. It then exports B. The subsystem, then, consists of entities E_A and E_B and its interface is described by I_B. A client can then import B and invoke its ports. The client only knows about I_B, and from its point of view, all the ports in B belong to I_B, including those implemented by E_A. From E_A's point of view its ports belong to I_A. To type-check the client's use of a port from E_A, we need to know the relationship between I_A and I_B. In addition, we would like to detect if E_B made an error in constructing B. This requires that the client and E_A exchange information at runtime.

To illustrate the problem of evolution, consider replacing E_A in the above example with a new version E'_A providing an interface I'_A that is compatible with I_A, in that every port in I_A is also in I'_A. Since clients may still have ports exported using I_A, E'_A needs to know the relationship between I'_A and I_A, as well as that between I'_A and I_B.

Initial research indicates that it may be feasible to provide interface types and type checking. Doing so requires that the programmer describe the relationship between an interface exported by an entity and that of the subsystem to which the entity belongs. This relationship can be described statically. Furthermore, it imposes an order on the replacement of entities in a subsystem. Further research is needed to determine if these constraints are acceptable.

7.8 Transaction Processing

Transaction systems are becoming widely used in database systems, office automation systems, and distributed systems. Implementations of transaction systems are large and complex, and involve subtle algorithms that interact in poorly understood ways. Formal techniques can be very useful in managing this complexity. In joint work with Alan Fekete, Nancy Lynch, and Michael Merritt, William Weihl has continued the development of a formal model for transaction systems that simplifies the description and verification of transaction-processing algorithms. The model is quite general, allowing a wide range of algorithms to be described. In the last year, we completed the description and verification of a general locking algorithm [112]. We also generalized proof techniques based on serialization graphs, originally developed for single-level transaction systems, to nested transaction systems [113]. In the original work on serialization graphs, recovery was essentially ignored by considering only executions in which all transactions commit. Our work clarifies the interactions with recovery by making explicit the assumptions about recovery that are implicit in earlier work.

In other work, William Weihl has analyzed the interactions between concurrency control and recovery in transaction systems [289]. There has been little previous theoretical work on recovery, and the extensive theoretical literature on concurrency control ignores recovery. However, not all "correct" recovery algorithms work with all "correct" concurrency control algorithms. We have developed techniques for analyzing the interactions between concurrency control and recovery, and have used them to give necessary and sufficient conditions for a concurrency control algorithm to work with each of several different recovery algorithms.

Our analysis of the interactions between concurrency control and recovery also suggests a useful methodology for verifying concurrency control and recovery algorithms that allows their interactions to be ignored as much as possible: first, give a very abstract description of both the concurrency control and the recovery algorithm. This description can be used to analyze their interactions without getting overwhelmed by details of either algorithm, and to prove that the combination of the two algorithms is correct in the sense that they ensure that transactions are atomic. Second, describe each algorithm in detail, and show that the detailed algorithm implements the more abstract algorithm. In this second step, the details of concurrency control can be ignored when showing that the detailed recovery algorithm is correct, and vice-versa. This proof methodology seems to lead to much simpler proofs.

7.9 Publications

[1] N. A. Lynch, M. Merritt, W. E. Weihl, and A. Fekete. Atomic transactions. In progress.

[2] M. Burke, R. Cytron, J. Ferrante, and W. Hsieh. The automatic generation of nested, fork-join parallelism. *Journal of Supercomputing,* 3(2), July 1989.

[3] G.T. Leavens and W. E. Weihl. *Reasoning about Object-oriented Programs that Use Subtypes.* Iowa State University, Department of Computer Science Technical Report TR #90-03, March, 1990.

[4] G.T. Leavens and W. E. Weihl. Reasoning about object-oriented programs that use subtypes. *1990 OOPSLA Conference*, October 1990. To appear.

[5] W.E. Weihl and P. Wang. Multi-version memory: software cache management for concurrent B-trees. Submitted to the Second IEEE Symposium on Parallel and Distributed Processing.

[6] W.E. Weihl. Fault-tolerant parallel computing. Position paper submitted to the Fourth ACM SIGOPS European Workshop on Fault-Tolerance Support in Distributed Systems.

[7] W.E. Weihl. Linguistic support for atomic data types. *ACM Transactions on Programming Languages and Systems,* April 1990.

[8] M. Herlihy, N.A. Lynch, M. Merritt, and W.E. Weihl. On the correctness of orphan elimination algorithms. *Journal of the ACM.* To appear.

[9] E. Kolodner, B. Liskov, and W. E. Weihl. Atomic garbage collection: managing a stable heap. In *Proceedings of the 1989 ACM SIGMOD International Conference on the Management of Data*, Portland, OR, June 1989.

[10] J.S. Emer and W. E. Weihl. Integrated interactive access to heterogenous distributed services. In *Proceedings of the 1990 Winter USENIX Conference*, Washington, DC, January 22-26, 1990.

[11] W.E. Weihl. Using transactions in distributed applications. In *Proceedings of the IEEE Compcon Spring 90 Conference*, San Francisco, CA, February 26 - March 2, 1990.

[12] A. Fekete, N. A. Lynch, and W. E. Weihl. A serialization graph construction for nested transactions. In *Proceedings of the ACM Symposium on Principles of Database Systems*, Nashville, TN, April 2-4, 1990. Also Technical Report MIT/LCS/TM-421, MIT Laboratory for Computer Science.

[13] E. Kolodner and W. E. Weihl. Crash recovery for object-oriented systems. Position paper for the 1989 Workshop on Database Programming Languages, OR, June 1989.

[14] W.E. Weihl. Support for large scale parallel computing. Position paper for the Third Workshop on Large Grained Parallelism, Pittsburgh, PA, October 10-11, 1989.

[15] A. Fekete, N. A. Lynch, M. Merritt, and W. E. Weihl. *Commutativity-based Locking for Nested Transactions.* Technical Report MIT/LCS/TM-370.b (supersedes TM-370), MIT Laboratory for Computer Science, July 1989.

[16] W.E. Weihl. The Impact of Recovery on Concurrency Control. MIT/LCS/TM-382.b (supersedes TM-382), MIT Laboratory for Computer Science, August 1989.

[17] E. Kolodner. *Design Bug in Argus Recovery System.* DSG Note 158, November 1989.

[18] E. Kolodner. Atomic incremental garbage collection and recovery for a large stable heap. Submitted to the Fourth International Workshop on Persistent Object Systems.

[19] A. Joseph and W.E. Weihl. *Application-Specific Recovery: Asynchronous Iterative Methods.* Parallel Software Group Note 1, June 1990.

[20] H. Bal. *Fault-tolerant Parallel Programming in Argus.* Technical Report IR-214, Department of Mathematics and Computer Science, Vrije Universitcit, Amsterdam, May 1990.

Theses in Progress

[1] P. Wang. *On the Performance of Concurrent B-tree Algorithms.* MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[2] Q. Huang. *Data Structures for Controlling Concurrent Search Algorithms.* MIT Department of Electrical Engineering and Computer Science, expected August 1990.

[3] E. Kolodner. *Integrating Crash Recovery and Garbage Collection.* MIT Department of Electrical Engineering and Computer Science, expected January 1991.

[4] S. Perl. *Performance Specification and Monitoring.* MIT Department of Electrical Engineering and Computer Science, expected June 1991.

[5] E. Waldin. *Communicating Values of Abstract Data Types Having Multiple Representations.* MIT Department of Electrical Engineering and Computer Science, expected June 1991.

Talks

[1] W.E. Weihl. Remote procedure call. Lecture given at Fingerlakes '89 Advanced Course on Distributed Systems, Ithaca, NY, July 10, 1989.

[2] W.E. Weihl. Transaction mechanisms. Lecture given at Fingerlakes '89 Advanced Course on Distributed Systems Ithaca, NY, July 13, 1989.

[3] W.E. Weihl. Using transactions in distributed applications. Lecture given at Fingerlakes '89 Advanced Course on Distributed Systems, Ithaca, NY, July 13, 1989.

[4] W.E. Weihl. Theory of nested transactions. Lecture given at Fingerlakes '89 Advanced Course on Distributed Systems, Ithaca, NY, July 13, 1989.

[5] W.E. Weihl. High level specifications of distributed programs. Lecture given at Fingerlakes '89 Advanced Course on Distributed Systems, Ithaca, NY, July 14, 1989.

[6] W.E. Weihl. Memory models for parallel and distributed computing. Lecture given at Memorial University, St. John's, Newfoundland, September 29, 1989.

[7] W.E. Weihl. Session Chair, Working Group on Parallelism: Lecture given at the Third Workshop on Large Grain Parallelism, Pittsburgh, PA, October 10-11, 1989.

[8] W.E. Weihl. Support for large scale parallel computing. Lecture given at the Third Workshop on Large Grain Parallelism, Pittsburgh, PA, October 10, 1989.

[9] W.E. Weihl. Memory models for parallel and distributed computing. Lecture given at Boston University, Boston, MA, November 8, 1989.

[10] W.E. Weihl. Using transactions in distributed applications. Lecture given at IEEE Compcon Spring 90 Conference, San Francisco, CA, February 28, 1989.

[11] W.E. Weihl. Multi-version memory: software cache management for concurrent B-trees. Lecture given at SRI International, Palo Alto, CA, March 27, 1990.

[12] W.E. Weihl. Multi-version memory: software cache management for concurrent B-trees. Lecture given at Hewlett-Packard Laboratories, Palo Alto, CA, March 29, 1990.

[13] W.E. Weihl. A serialization graph construction for nested transactions. Lecture given at the 1990 ACM Symposium on Principles of Database Systems, Nashville, TN, April 2, 1990.

[14] W.E. Weihl. Multi-version memory: software cache management for concurrent B-trees. Lecture given at Digital Equipment Corporation, Cambridge Research Laboratory, Cambridge, MA, May 14, 1990.

[15] S.E. Perl. Performance specifications. Lecture given at Digital Equipment Corporation Systems Research Center, Palo Alto, CA, June 13, 1990.

Mercury

Academic Staff

B. H. Liskov, Group Leader

Research Staff

D. Clark	K. Sollins
L. Shrira	J. Wroclawski

Graduate Students

E. Waldin

Undergraduate Students

J. Choi	V. Singhal

Support Staff

A. Aldrich	K-J. Madera
A. Rubin	

8.1 Mercury System Development

Rob Austein, Karen Sollins, and John Wroclawski continued to develop the Mercury system. Recently, they implemented a second generation version of Mercury for C/Unix programmers. This system combines work in the areas of RPC semantics, heterogenous communication, and binding architectures to form a substrate for multi-architecture distributed computing. We summarize our work in each of these areas separately because each result is useful on its own.

Programs in the Mercury system communicate using pipelined sequences of remote procedure calls known as *call streams.* Rob Austein and John Wroclawski developed and implemented a practical design for call streams. We defined the behavior of call streams in the presence of flow control and limited memory resources. We specified a standardized network transport layer model called the Mercury Virtual Transport. This model specifies the services Mercury expects from the underlying network. We designed a protocol which implements the full semantics of call streams using this transport model. Use of the virtual transport concept allows us to specify the call stream protocol in a network-independent fashion. We have defined a mapping protocol which implements the virtual transport over traditional byte stream transports such as TCP or ISO TP4. We have used this work to implement call streams over both Unix intra-machine and IP/TCP transport protocols.

Presentation functions allow heterogeneous systems to communicate in terms of typed data objects. Since its inception, the Mercury project has explored the effects of introducing user-defined or abstract types into the presentation layer. This year, Karen Sollins and John Wroclawski have investigated presentation architectures which support evolvable systems, in which individual modules may be enhanced or replaced gracefully without requiring changes in other modules. We propose a system which supports abstract presentation types, a well defined model of type compatibility, and a set of system-supported implicit type conversion rules. We argue that this allows decentralized systems to be specified and constructed in a manner which is type-safe and precisely captures the user's intent, while supporting flexible and evolvable interrelationships between modules. We are currently implementing a Mercury presentation layer based on these principles.

Karen Sollins and John Wroclawski implemented a binding architecture which supports long-lived modules, which may crash and restart without loss of state; and mobile modules, which may move from machine to machine invisibly to the client. Mercury remote procedures are accessed through *ports*, typed transmissable procedure-valued objects which reference procedures at remotely available modules. We implemented the notion of ports with pre-bound arguments, set at port creation rather than call time. A principal use for this mechanism is to transparently utilize a single remote procedure as the "operation handler" for many "objects" by associating the handler procedure with a number of ports, each containing a different "object" as a pre-bound argument. Pre-binding elegantly supports a single invocation mechanism which can appear either procedure-oriented or object-oriented, depending on the requirements of the particular problem.

Rob Austein and John Wroclawski developed a simple model for exception and condition handling which replaces the ad hoc mechanisms often used in C programs. Our model views exceptions as belonging to a hierarchically organized class structure, and allows the programmer to dynamically bind an exception handler to a class of exceptions. Exception handlers may take a number of actions, ultimately either restarting the computation or unwinding to an enclosing stack level. We implemented this mechanism as a C library, which is in use within the Mercury project and has been distributed to several sites outside MIT.

Rob Austein designed and implemented a lightweight process (threads) package for Unix which provides features missing from other work, including preemptive scheduling, reasonable synchronization primitives, and graceful interaction with the Unix I/O system.

8.2 Mercury Presentation to Open Software Foundation

Barbara Liskov, Bill Weihl, and John Wroclawski presented the Mercury project to the Open Software Foundation in response to a Request for Proposals for distributed computing technology. Our presentation comprised an overview of the Mercury system, detailed responses to questions considered important by the OSF evaluation team, and a proposal describing one possible integration of Mercury's ideas into a larger distributed computing toolbox.

Barbara Liskov presented our submission and served on a panel at the initial meeting of OSF member organizations. John Wroclawski attended a series of review meetings, discussing Mercury with the OSF team and other interested parties.

Although the OSF eventually chose to base their efforts entirely on commercially available technology, our work received substantial exposure and comment as a result of this presentation.

8.3 Modular Application Environment (MAE)

Mike Frumkin and John Wroclawski are exploring techniques to make applications accessible as building blocks to relatively naive users. Our concept is to define the behavior of an application in an abstract logical manner, rather than as a specific set of user interface actions. We then present the same abstract interface directly to the user through an interface tool of the user's choice, and to other applications through a set of remote procedures. A global naming and support environment allows users to specify inter-application links in an intuitive fashion. MAE uses Mercury as a communications substrate.

This work, which will constitute Mike Frumkin's Master's thesis, was supported by an RA from the Advanced Network Architecture Group during the spring of 1990.

8.4 Synchronized Clock Message Protocol

Barbara Liskov, Liuba Shrira, and John Wroclawski developed the Synchronized Clock Message Protocol, a new message passing protocol which provides guaranteed detection of duplicate messages even when the receiver has no state stored for the sender. The method is based on the assumption that clocks throughout the system are loosely synchronized. Our work shows how to build higher level protocols, such as RPC, which provide at-most-once semantics without requiring a performance-lowering connection setup step. We implemented a SCMP-based RPC protocol within the widely used Sun RPC library, and showed that SCMP-based at-most-once remote procedure calls could be provided at the same cost as less desirable RPC's that do not guarantee at-most-once execution.

Liuba Shrira presented an early version of this work at the Second IEEE Workshop on Workstation Operating Systems, and Barbara Liskov gave a talk about the work at the University of Arizona. A later version has been accepted for presentation at the 1990 SIGCOMM conference.

8.5 Publications

[1] J. Emer and W. Weihl. *Integrated Interactive Access to Heterogeneous Distributed Services.* Programming Methodology Group Memo 67, MIT Laboratory for Computer Science, December 1989.

[2] B. Liskov, L. Shrira, and J. Wroclawski. *Efficient At-most-once messages Based on Synchronized Clocks.* Technical Report MIT/LCS/TR-476, MIT Laboratory for Computer Science, April 1990. To be published in the *Proceedings of ACM SIGCOMM '90 Symposium.*

Theses Completed

[1] J. Choi. *Investigation of Triggering in a Small Distributed Application.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[2] V. Singhal. *Adaptation of a Synchronized Clock Protocol for the Mercury Transport Layer.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

Talks

[1] B. Liskov. Mercury and the object repository. Lecture given to DARPA at MIT/LCS, October 1989.

[2] B. Liskov. High level communication in heterogeneous systems. Lecture given at Lotus Development Corp., Cambridge, MA (July); Hewlett Packard, Paris, France (September); Brown University, Providence, RI (September); George Mason University, Fairfax, VA (October); Digital Equipment Corp. CRL, Cambridge, MA (November), 1989.

[3] B. Liskov. High level communication in heterogeneous systems. Lecture given at University of Arizona, Tucson, AZ; BBN, Cambridge, MA, January 1990.

[4] B. Liskov. At-most-once message protocol. Lecture given at University of Arizona, Tucson, AZ, January 1990.

[5] L. Shrira. Efficient at-most-once messages based on synchronized clocks. Lecture given at Second IEEE Workshop on Workstation Operating Systems, Pacific Grove, CA, September 1989.

[6] J. Wroclawski. The Mercury system. Lecture given at Oracle Corp., Belmont, CA, December 1989.

Programming Methodology

Academic Staff

B. Liskov, Group Leader

Research Staff

D. Curtis	L. Shrira
P. Johnson	

Graduate Students

B. Ben-Zvi	S. Ghemawat
E. Brewer	R. Gruber
C. Dellarocas	S. Markowitz
D. Hwang	C. Waldspurger
B. Gates	M. Williams

Undergraduate Students

M. Levine	A. Rhee
V. Nicotina	M. Sexton
C. Poelman	R. Stata

Support Staff

A. Rubin

9.1 Introduction

Research in the Programming Methodology Group has continued to focus on the area of distributed computing. We are working on a replicated Unix file system for use via the NFS protocol [259][276], and on a highly-available object repository for use in a heterogeneous distributed network. In addition, we continued our work on replication methods and atomic transactions.

9.2 Replicated File System

Our replicated file system has the following goals:

1. It should provide the same semantics as an unreplicated NFS server.
2. It should be usable with whatever NFS client code exists at the client machine.
3. We want to avoid having our system depend on proprietary information. Instead, our code is sandwiched between NFS and the Unix file system kernel. It is called by the NFS code at the server, and in turn makes calls on low level file system operations.
4. We want to continue to provide service even when one replica is crashed or inaccessible, but have only two copies of each file.
5. We want to achieve reliability at least as good as a single server.
6. We want to achieve performance as least as good as a single server. In particular, the delay observed by the client in doing a read or write should be no greater with our service than with a single server.

We plan to use a primary copy method as our replication technique. Our method is based on our earlier work on primary copy methods in transaction systems [236], but we adapted this approach to match the needs of this application. Most importantly, we take advantage of the fact that we need not support general atomic transactions. Instead, each individual file system operation must run atomically, but support for combining operations into multi-operation transactions is not needed.

Each file system is assigned to a pair of servers; one is the primary and the other is the backup. The roles assigned to different servers can change when there are failures, and at this point, the third server will be involved; failures are discussed further below. Different file systems can be assigned to different pairs; in this way we spread the load among the servers.

In a primary copy method, client requests are sent to the primary, which decides what to do and communicates with the backup as needed. Running single operation transactions requires a two phase protocol. In phase 1, the primary informs the backups about the operation. When the backups acknowledge receipt of this information, the operation can commit. At that point the primary can return information to the client; the backups are informed about the commit later in the background (this is phase 2). Phase 1 information must reach a sufficient number of backups so that we can guarantee that the information survives subsequent failures; in a system like ours that is intended to survive a single node failure, just one backup is needed.

The primary maintains a log in volatile memory in which it records information about client operations. The log is simply a sequence of entries; later entries represent more recent operations. Typically some entries are for operations in phase 1, while others are for operations that have committed. The primary distinguishes between these by maintaining the CP (the commit point); this is the index of the latest committed entry. Operations commit in entry order.

Operations that do not involve modifications to the file system are done entirely by the primary; no log entries are created for them and no communication with the backup is needed. (We discuss how we guarantee proper serialization for such operations later in this section.) For a modification operation, the primary sends the logged information to the backup. When the **ack arrives, the primary advances its CP (backups do acks in log entry order). The primary can work on many user requests in parallel; an operation must be delayed only if it conflicts with an earlier, uncommitted operation.

Operations are not applied to the primary's file system until after they commit, and this writing occurs in the background. Each server is equipped with an uninterruptible power supply (UPS), so that it will be able to write its log to disk in the case of a power failure.

The backup records information received from the primary in its volatile log. The primary also informs it about the current CP in each message, and the backup records this information in its CP. Like the primary, the backup moves committed information to its file system in the background.

The information in an entry includes more than just the arguments sent by the client. For example, the primary will choose the time at which a write operation is to occur, and log this information. By logging sufficient information, we can insure that the effect of applying a client operation to the file system is identical at both the primary and the backup. In addition, we can guarantee that operations are idempotent: even if an operation is performed a second time (which can happen when there is a failure), the effect is the same as if it happened just once.

Information is removed from the log when it is known to be recorded in the file systems at both the primary and backup. Each server maintains a counter called the AP (the application point); all entries with index less than or equal to the AP have been applied to the file system. Servers send their APs in messages; a log entry can be discarded when it is known to be earlier than both APs.

As mentioned, each file system is the responsibility of a pair of workers; the third server acts as a "witness" for that system [213][243]. If one of the workers becomes inaccessible, the other worker and the witness carry out a *view change* [103][102]. The remaining worker will be the primary of the new view and the witness will be the backup. Such a "promoted" witness keeps a log just like a regular backup, but it does not have a copy of the file system, so it does not apply committed requests to the file system. Instead, it keeps the earlier entries in the log on disk. (The worker in a view with a promoted witness can discard entries in its log as soon as they have been recorded in its file system, and as soon as that part of the witness' log is on the witness' disk.) We are not yet certain what we will do if the view lasts long enough that the promoted witness' log becomes too big to store. One possibility is to keep most of the witness' log on tape.

The view change algorithm guarantees that any modification operation that completed (i.e., that returned to the client) will be recorded in the system state. In addition, operations that did not return may also survive into the new state; these are operations that made it to a backup, but where the primary of the previous view had not yet notified the client.

As mentioned, read operations are done locally at the primary. This can lead to a serialization problem if a new view has formed but the old primary does not know about it. In that case, a write operation that committed in the new view may not be reflected in the result of the read returned to the client, even though the client may know that the write has happened. To avoid this serialization problem, we make use of loosely synchronized clocks [226] to define "time windows" during which a view change will now happen. Each message sent by the backup to the primary contains a time equal to its clock's time + δ; here δ is on the order of a few seconds. This time represents a promise by the backup not to start a new view until that time has passed. The primary needs to communicate with the backup about a read operation only if the time of its local clock is greater than the promised time -ϵ, where ϵ is the clock skew. When a new view starts, it must be delayed until the time of the new primary's clock is greater than the promised time of the backup. In this way, we guarantee that there cannot be a write that committed in the new view and that should occur before a read in an earlier view.

9.3 Object Repository

The object repository has two goals:

1. It is intended to provide a convenient medium for sharing of information among programs written in many different programming languages.

2. In addition, it will provide support for the construction and execution of distributed programs. Components of these programs can be implemented in different programming languages; communication will occur through the repository.

We identified requirements for the system in support of these goals. For the first goal, we believe the following requirements are important:

1. The repository should store information as objects, and objects should be able to refer to one another.

2. Each object can be accessed only by calling operations of its type. Mechanisms that allow users to define new types must be provided.

3. All accesses to objects happen within atomic transactions, so that the repository can maintain multi-object consistency constraints.

4. The repository must provide database functionality. In particular, fast access to large collections of objects is needed.

5. The repository should provide reliable persistent storage: with high probability, it will guarantee to preserve information entrusted to it.

6. The repository should provide highly available storage so that whenever a client needs to access an object, it will be able to do so with high probability.

7. The repository must be scalable in many dimensions: object sizes, number of objects, number of clients, size of network (e.g., local area network vs. geographically distributed).

8. The repository must provide a security and protection mechanism.

In addition, the repository must perform well, but it is unclear at this point what kind of performance can be expected from such a system.

These requirements also apply to the second goal (with the possible exception of database functionality). In addition, to support this goal we need to provide a way for clients to communicate via the repository. We have in mind here some way for a client to "post" a request for service so that servers for that service can find out about the request in a timely manner. We note that this model of computation differs from the more conventional one of remote procedure calls (it bears some similarity to the Linda model [116]). It offers two advantages over RPCs: reliable communication in spite of client and server failures, and good support for allowing multiple clients to communicate with multiple servers.

In the remainder of this section we describe the semantics of the object universe provided by the repository. The universe contains objects that clients can share. Each object has a unique name (an *object identifier*, or *oid*) and a value. Clients can identify objects by providing their oids, and objects can refer to one another using oids.

An object also has a type that determines the set of operations that can be applied to it. The repository guarantees that objects are accessed only by means of the operations of their type. Thus, these operations are the only way that the objects' values can be observed or modified.

The repository provides a rich set of builtin types (e.g., integers, booleans, characters) and constructors (e.g., arrays, records, unions). Sets will be provided as a builtin constructor, and clients can cause indexes to be provided for sets, to speed up queries. (We may extend this ability to "set-like" constructors.) In addition, users can define new abstract types for the repository. We are currently working on a method to allow efficient maintenance of indexes on sets where the objects in the set are of abstract type.

Clients of the repository interact with it by invoking operations on its objects. These calls (from clients to the repository) follow call-by-value semantics. Arguments and results are usually oids. For example, a call to add an employee to a set of employees would take the oid of the set and the oid of the employee as arguments.

However, sometimes actual values of objects are needed. For example, suppose one wanted to add the number ten to an array of integers. While ten is conceptually an object in the repository, it hardly makes sense to require a client to know what ten's oid is. Instead, along with oids, it must be possible to use values as arguments and results. For a given type T in the repository, T can have an *external representation*, ext(T); this is a description of the format of the data that a client will receive (when reading the value) or provide (when specifying a value). The external representation is similar to a *message representation* used for communication in distributed system [145][199]. It differs from the way the object is represented within the repository, and also from the way the value will be represented in the client program. Typically, translations are provided on both ends: the internal representation is *encoded* to produce the external representation by the provider of the value, and the external representation is *decoded* to produce the internal representation used by the receiver.

Within the repository, if the definer of type T provides an encode routine from T to ext(T), then it is legal to return result values of type T to client programs. If the definer provides a decode routine from ext(T) to T, then it is legal to accept argument values of type T from client programs. The simple builtin types (integers, reals, characters, etc.) will provide encode and decode routines; thus, like most object repositories that use an abstract type approach, simple values can be used as arguments and results. Unlike most other object repositories (e.g., Encore [268]), values of user-defined type can also be arguments or results, provided the type has the encode or decode routine.

The types of objects in the repository are language-independent and a way is needed for describing them that is independent of any programming language (such as the one used to implement them). This can be accomplished by providing a "type description" that defines the names and signatures for the type's operations and also the external representation if the type is to be transmitted as a value. The signatures will be defined in terms of other types known to the repository. In addition, the definer needs to give a specification so that programmers who wish to use objects of the new type can understand its meaning. The important point to notice here is that all of this can be done without defining an implementation for the new type, and the information is independent of the programming language that will be used to implement the type. Thus, abstract data types provide a means for programs in different languages to communicate.

To provide good performance, we will allow clients to invoke several operations in one call; this facility is needed, for example, to run queries efficiently. We have not yet decided on how powerful the language for defining such "combined operations" will be (e.g., whether it will just provide expressions, or whether general programs can be written). All operation calls must take place within atomic transactions; a transaction can contain one or more calls.

In addition to calls, we will also provide a means for clients to post requests for service, and to request notification when such requests arrive. We have not yet decided how this mechanism will work.

9.4 Disconnected Actions

Boaz Ben-Zvi has completed his Master's thesis on *disconnected actions.* This work assumes the nested transaction model supported by Argus [200]. A *topaction* is a transaction that has no parent; when it commits, its results become permanent. An action is allowed to create *subactions*, whose commits are relative to the parent: when a subaction commits, its effects become visible to the parent, but if the parent aborts later, this undoes the effects of the subaction. In Argus, a parent action stops running when it creates a subaction, and all subactions must terminate before it is allowed to continue running.

Disconnected actions are subactions that are allowed to run in parallel with their ancestors; the only constraint is that all disconnected actions must terminate before their topaction ancestor is permitted to commit. They are useful to allow lazy propagation of information. For example, consider a replicated system in which a read or write must be done to three out of the five replicas. In such a system, writing to more than three replicas can improve the performance of later reads because the needed information may reside at replicas that are closer to the reader. However, the transaction doing the write should not need to be delayed while the writing to the additional replicas occurs. Disconnected actions can be used to allow the extra writing to go on in the background, while the parent continues to do other work.

Ben-Zvi worked out the locking and commit rules that are needed to support disconnected actions. He investigated several different approaches. For example, the commit of a topaction can be made conditional on some number of a set of disconnected action descendants committing; if at least that many have committed when the topaction tries to commit, the topaction commits immediately and any of the disconnected actions that have not terminated are aborted.

9.5 Viewstamp Replication in Argus

Sanjay Ghemawat [117] implemented Oki's viewstamp replication mechanism [235][236] and measured the performance of the implementation. His work took place within the Argus system [198][200]. Argus guardians are resilient to node failures because their state variables are maintained on stable storage [184]. Having resilient guardians supports the construction of highly reliable systems that, with high probability, do not lose information entrusted to them. However, it does not support high availability: if a guardian's node is crashed or inaccessible because of a network failure, clients will be unable to use the guardian.

Ghemawat implemented a version of Argus in which each guardian is implemented as three replicas. One of the replicas is the primary; the others are backups. All operation calls are performed at the primary. Whenever a guardian would write some information to stable storage in the original Argus implementation (e.g., as part of committing a transaction), the primary sends that information to the backups. In case of a failure of the primary, the backups perform a view change [102][103], and one of them becomes the new primary. In this way, the guardian as a whole is highly available; its state is also highly reliable provided each replica has an uninterruptible power supply (UPS) that permits it to write volatile information to disk in the event of a power failure.

Top-action	Original	Replicated			
		1,1	1,3	3,1	3,3
Read	40	43	61	43	63
Write	127	50	62	70	82

Figure 9.1: The time (in milliseconds) required to run a topaction consisting of one handler call. Original refers to the unreplicated system. a, b refers to a replicated system with a replicas of the client and b replicas of the server.

Ghemawat measured the performance of his system and compared it with that of Argus. For example, Figure 9.1 shows the performance of topaction that performed a single handler call that either observed the state or modified it. The figure shows that in the case of modifications, the new system performs substantially better than Argus; this is because in Argus the new information must be written to disk, which takes longer than the roundtrip message required in the new system. In fact, the situation in Argus is actually worse than it appears, since Argus implements stable storage with a single disk instead of two disks with synchronous writes (which is what is really needed) [184]. On the other hand, the new system degrades the performance of reads because our implementation requires a roundtrip message delay in a case where no disk write is done in Argus. (The "time window" optimization for the replicated file system discussed in Section 9.2 avoids this delay; with this optimization, reads should perform the same in the replicated system as in Argus.)

Since the replication scheme requires all replicas to have disks and UPSs, it is probably not appropriate for use with all Argus guardians. Instead, it should be used for important services, such as the object repository. In addition, we believe that the replication technique would work well in a stable storage service, which provides highly available and reliable storage for guardians as a service in a network. We are investigating such an approach and intend to implement the service and compare its performance with other techniques [85][77].

9.6 Orphan Detection

Steve Markowitz [221] completed his Master's thesis in which he implemented the "map server" scheme for doing orphan detection in Argus and compared its performance with the "deadline" technique.

An orphan is a computation whose results are no longer required. Orphans are undesirable because they waste system resources and because they sometimes observe inconsistent information. Therefore Argus provides a method of detecting orphans so that they can be destroyed. Our method requires sending orphan-detection information in almost all messages [202]. Since the information grows without bound, the technique is only feasible if it can be optimized in a way that keeps the size of messages and the size of the orphan detection information small.

We developed two techniques for optimizing our basic method. The deadline strategy does this by limiting the lifetimes of certain entities (such as topactions), which in turn limits the time that orphan information must be retained [288]. The map server strategy keeps the information in a highly available central service; the service is able to do garbage collection and thus bound the information size, and furthermore, it associates small timestamps with system states, and these timestamps are sent in messages instead of the information they identify [202]. The deadline strategy was implemented and analyzed by Nguyen [229]; Markowitz has now done the same thing for the map-server technique. He discovered that although both techniques work well in small systems, the deadline scheme appears to work better in large systems because it exhibits better locality.

9.7 Lazy Replication

Rivka Ladin, Barbara Liskov, and Liuba Shrira have continued to work on the replication method developed by Ladin in her Ph.D. thesis [181]. A paper on this work [182] will appear in the *Proceedings of the Ninth ACM Symposium on Principles of Distributed Computing.* In addition, Liskov and Sanjay Ghemewat constructed a simple implementation of the method for a particular application, with the goal of determining the cost of the replication technique. To determine this, we are carrying out experiments that compare the performance of the replicated service with an unreplicated one for the same application. Preliminary results indicate that the replicated service provides response times comparable to those of the unreplicated one; we are working on a study to determine and compare the capacities of the two systems.

9.8 Avoiding Recursion Deadlock

Eric Brewer and Carl Waldspurger explored issues of locking and serialization in concurrent object-oriented programming languages. Their work focused on the problem of *recursion deadlock* in current actor systems [214][297]. The restrictions on recursion in these systems hinder the use of abstract modules, and force programmers to rethink algorithms to avoid recursion. Brewer and Waldspurger developed two mechanisms for solving this problem: a novel technique using *multi-ported actors*, and a *named threads* scheme that borrows from previous work in distributed computing.

9.9 Publications

[1] B. Ben-Zvi. *Disconnected Actions: An Asynchronous Extension to a Nested Atomic Action System.* Technical Report MIT/LCS/TR-475, MIT Laboratory for Computer Science, January 1990.

[2] J. Emer and W. Weihl. *Integrated Interactive Access to Heterogeneous Distributed Services.* Programming Methodology Group Memo 67, MIT Laboratory for Computer Science, December 1989.

[3] S. Ghemawat. *Automatic Replication for Highly Available Services.* Technical Report MIT/LCS/TR-473, MIT Laboratory for Computer Science, Cambridge, MA, March 1990.

[4] R. Ladin, B. Liskov, and L. Shrira. Lazy replication: exploiting the semantics of distributed services. In *Proceedings of the Ninth ACM Symposium on Principles of Distributed Computing*, Quebec, Canada. To appear.

[5] R. Ladin, B. Liskov, and L. Shrira. Lazy replication: exploiting the semantics of distributed services (extended abstract). In *Proceedings of the Workshop on Fault-tolerant Support in Distributed Systems*, Bologna, Italy. To appear.

[6] B. Liskov, R. Gruber, P. Johnson, and L. Shrira. A highly available object repository for use in a heterogeneous distributed system. In *Proceedings of the Fourth International Workshop on Persistent Object Systems Design, Implementation, and Use*, Martha's Vineyard, MA. To appear.

[7] B. Liskov, L. Shrira, and J. Wroclawski. *Efficient At-most-once Messages Based on Synchronized Clocks.* Technical Report MIT/LCS/TR-476, MIT Laboratory for Computer Science, April 1990. Also to appear in *Proceedings of ACM SIGCOMM '90 Symposium.* Also submitted for journal publication.

[8] B. Liskov, R. Gruber, P. Johnson, and L. Shrira. A replicated Unix file system. *Proceedings of the Workshop on Fault Tolerant Support in Distributed Systems*, Bologna, Italy. To appear.

[9] B. Liskov. Challenges in distributed systems. In *Research Directions in Computer Science: An MIT Perspective*, MIT Press, 1990. To appear.

Theses in Progress

[1] R. Stata. *Block Assignment Problem.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected.

Theses Completed

[1] B. Ben-Zvi. *Disconnected Actions: An Asynchronous Extension to a Nested Atomic Action System.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, January 1990.

[2] P. Friedman. *An Environment for Debugging Concurrent Software in Trellis.* Bachelor's and Master's theses, MIT Department of of Electrical Engineering and Computer Science, May 1990.

[3] S. Ghemawat. *Automatic Replication for Highly Available Services.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, January 1990.

[4] S. Markowitz. *Central-server-based Orphan Detection for Argus.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[5] V. Nicotina. *Implementing Mobile Guardians in Argus.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[6] C. Poelman. *Issues in the Design of a Portable Debugger.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[7] A. Rhee. *Witnesses and Using Time to Avoid Communication in Viewstamped Replication.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[8] M. Sexton. *A Test Driver Generator.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

Talks

[1] B. Liskov. Mercury and the object repository. Lecture given to DARPA, MIT/LCS, October 1989.

[2] B. Liskov. High level communication in heterogeneous systems. Lecture given at Lotus Development Corp., Cambridge, MA (July); Hewlett Packard, Paris, France (September); Brown University, Providence, RI (September); George Mason University, Fairfax, VA (October); Digital Equipment Corp. CRL, Cambridge, MA (November), 1989.

[3] B. Liskov. The Mercury communication system. Lecture given at Open Software Foundation Member Meeting, November 1989.

[4] B. Liskov. High level communication in heterogeneous systems. Lecture given at University of Arizona, Tucson, AZ; BBN, Cambridge, MA, January 1990.

[5] B. Liskov. Distributed applications. Lecture given to American Express Corp., MIT/LCS, January 1990.

[6] B. Liskov. At-most-once message protocol. Lecture given at University of Arizona, Tucson, AZ, January 1990.

[7] B. Liskov. Structure of distributed systems. Lecture given at IEEE Conference (Keynote Address), Nice, France (March); Purdue University, W. Lafayette, IN (April), 1990.

[8] L. Shrira. Efficient at-most-once messages based on synchronized clocks. Lecture given at Second IEEE Workshop on Workstation Operating Systems, Pacific Grove, CA, September 1989.

Programming Systems Research

Academic Staff

D. Gifford, Group Leader

Graduate Students

N. Glasser	L. Lemaire
J. O'Toole	J. Rees
M. Sheldon	F. Turbak

Undergraduate Students

A. Francomano	S. Raisty

Support Staff

L. Bouck	R. Bisbee

Visitor

P. Jouvelot

10.1 Introduction

The Programming Systems Research Group works in two areas: programming language technology for parallel systems, and distributed database technology for large scale information systems.

10.2 Community Information

The Boston Community Information System is a large scale information system in use at over 150 sites in the Boston area. It provides *New York Times* and *Associated Press* news wires to users via digital broadcast (to PCs), remote procedure calls, and electronic mail. We published a description of our system in *Communications of the ACM* (see list of publications below).

Since last year, we doubled the number of users served by the electronic mail component of the system to over 100. We conducted a survey of the electronic mail users as well as compiled quantitative data for a forthcoming report.

10.3 FX Programming

The FX programming language incorporates a fundamental compiler technology for parallel computers that can be used by a range of existing programming languages without modification. This compiler technology is based on a new formal technique for statically scheduling expressions for parallel execution. We are using implementation experience and experimentation to guide the development of the technology.

10.3.1 Approach

Our approach to parallel computing seeks to develop a new scientific basis for parallel program execution that retains the simple semantics present in sequential programming languages. Our approach is unlike other approaches to parallel computing because it does not require primitives for explicit parallelism (although it can accommodate them) and because it does not forbid side effects in programs.

The foundation of our work is an *effect system* that permits us to statically determine expression scheduling constraints and thus decompose a program at compile time for parallel execution. Just as a type system describes *what* each expression in a program computes, an effect system describes *how* each expression computes. For example, an effect system can determine expression side effects (read, write, and initialize regions of memory), control effects (representing control transfers), and communication effects (between cooperating processes).

Our experimental results with a prototype compiler suggest that effects are a useful way of discovering and exploiting parallelism in complex programs without burdening the programmer. Our experimental methodology is based upon testing our technology on programmers outside of our own research group by an iterative cycle of language design, implementation, and test.

10.3.2 Recent Accomplishments

We completed experiments using our parallel compiler for FX-87 that show between a factor of two and five speedup under ideal conditions. New systems for type inference, effect inference and first class modules have been implemented in the context of FX-90, and our implementations are being used outside of our research group.

In addition, we completed:

- a dialect of FX, in use at other research sites and in an MIT graduate course. Based upon user feedback, the language has been updated for easier use.
- a prototype FX-90 implementation incorporating type and effect inference: no declarations are necessary to get the benefit of type and effect analysis.
- an extension of our earlier system of first class modules and static dependent types. Effects enable a new approach to type systems that permits modules to be treated as run time values, a first for statically typed languages. Modules can be dynamically composed to construct new systems. This system has been implemented as part of the prototype FX-90 implementation.

10.3.3 Plans for FY91

In the following year, we intend to:

- complete FX technology experiments under real conditions on the Encore Multimax and Connection Machine and publish the experimental results; and
- encourage technology transfer by publishing the complete documentation of FX and making the implementation widely available.

10.3.4 Technology Transfer

Outgoing technology transfer activity is split into two parts. First, through our scientific publications and experimental results, we seek to influence other groups to understand and use the technology we develop. Second, we support the transfer of FX implementations to interested users. To date, we sent the implementations of FX to approximately three other university research groups, and we have received feedback from these users.

In addition, we are also working on transferring our distributed database technology, developed under a previous DARPA contract, to outside firms for commercialization. This technology is used at over 100 sites world wide.

Incoming technology transfer in the programming language area is facilitated by our constant interaction with other DARPA contractors, including Stanford, Yale, and CMU.

10.3.5 Other Information

Pierre Jouvelot has joined our research group as a visiting scientist, and is working on adapting FX technology to the Connection Machine. We hosted many visitors, including visitors from industry, over the past year and discussed our ongoing research. In September 1990 David Gifford was awarded the Karl Van Tassel Career Development Professorship.

10.4 Publications

[1] D. Gifford. Polychannel systems for mass digital communication. In *Communications of the ACM*, 33(2):141–151 February 1990.

[2] P. Jouvelot and D. Gifford. Parallel functional programming: the FX project. In *Parallel and Distributed Algorithms*, M. Cosnard, et al. editors, Elsevier/North-Holland, pages 257-267, 1989.

[3] M. Sheldon and D. Gifford. Static dependent types for first-class modules. In *Proceedings of the ACM Lisp and Functional Programming Conference*, Nice, France, June 26-29, 1990.

Theses Completed

[1] J. Rauen. *A Module System for Scheme.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

Theses in Progress

[1] L. LeMaire. *An Architecture for Software Reusability Tools.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected September 1990.

[2] F. Turbak *An Inquiry into the Shape of Computation.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected September 1990.

[3] J. O'Toole. PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected May 1992.

[4] M. Sheldon. *An Automatically Indexed Module Repository.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected May 1992.

Awards

[1] D. Gifford. Awarded the Van Tassel Professorship for 1989-90 to support his research in molecular biology and its relation to computation.

Spoken Language Systems

Research Staff

J. Glass	S. Seneff
H. Leung	V. Zue, Group Leader
M. Phillips	

Graduate Students

N. Daly	H. Meng
D. Goddeau	P. Niyogi
L. Hetherington	J. Pitrelli
R. Kassel	M. Soclof
J. Marcus	

Undergraduate Students

U. Chaudhari	D. Whitney
M. McCandless	

Technical Staff

D. Goodine	J. Polifroni

Support Staff

V. Palay

11.1 Introduction

Spoken language input to computers is a major goal in our research in developing a graceful human-machine interface. Despite some recent successful demonstrations of speech recognition capabilities, current systems typically fall far short of human capabilities of continuous speech recognition with essentially unrestricted vocabulary and speakers, under difficult acoustic environments. Our approach to this problem is to seek a good understanding of human communication through spoken language, to capture the essential features of the process in appropriate models, and to develop the necessary computational framework to make use of these models for machine understanding.

It is our belief that the development of advanced human/machine communication systems will require expertise in signal processing, system theory, pattern recognition, and computer science, built on a solid understanding of speech science and linguistics. We place heavy emphasis on designing systems that can make use of the knowledge gained over the past four decades in human communication, with the hope that such systems will one day have a performance approaching that of humans. Specifically, our approach is based on the following premises:

- The speech signal contains information regarding the intended linguistic message. It also contains information on the acoustic environment and the identity and physiological/psychological states of the speaker. As far as speech recognition is concerned, the latter sources of information can be considered as undesirable noise. Robust speech recognition is critically tied to our ability to successfully extract the linguistic information and discard those aspects that are *extra-linguistic.*

- Past research in spoken language communication has established *phonemes* as psychologically real units for representing words in the lexicon. Therefore, phonemes and other equivalent descriptors, such as distinctive features and syllables, are the most appropriate units to relate words to the speech signal for machine recognition as well.

- While phonemes are *discrete* abstract linguistic entities, their acoustic realizations in speech are inherently *continuous*, reflecting the movement of the articulators from one position to the next. Many of the acoustic cues for phonetic contrasts are encoded at specific times in the speech signal. In order to fully utilize these acoustic attributes, we believe that one must explicitly establish acoustic landmarks in the signal.

- Previous attempts at explicit utilization of speech knowledge have resulted in the development of systems that are based on heuristic rules. Such efforts typically require intense knowledge engineering, and as such are often hampered by the lack of a unified control strategy. As a result, system development is slow, and the performance fragile. In contrast, we seek to make use of the available speech knowledge by embedding such knowledge in a formal framework whereby powerful mathematical tools can be utilized to optimize its use.

- Despite significant advances made in phonetics, phonology, and other aspects of linguistics over the past decades, we still lack a complete understanding of the human speech communication process. To deal with our present state of ignorance and the inherent variability that exists throughout the process, the speech recognition system must have a stochastic component. However, it is our belief that speech-specific knowledge will enable us to build more sophisticated stochastic models than what is currently being attempted, and to reduce the amount of training data necessary for high performance.

- The ultimate goal of our research is the *understanding* of the spoken message, and the subsequent accomplishment of a task based on this understanding. To achieve this goal, we must fully integrate the speech recognition part of the problem with natural language processing so that higher level linguistic and pragmatic constraints can be utilized.

- The development of a spoken language understanding system will require interactions with several disciplines in computer science. Parallel computing will be necessary for real time processing. Efficient algorithms can greatly reduce the search space for the recognition process. Finally, theories of learning will help the system to adapt to new speakers, environments, and tasks.

The research projects in the Spoken Language Systems Group fall into several areas. First, a number of basic research topics are being explored. These include the formulation and testing of various computational models for human auditory processing, speech perception, and natural language processing that are suitable for spoken language understanding. We are also attempting to quantify the acoustic cues for phonetic contrasts, and the effects of speaking rate and style on the acoustic properties of speech. Secondly, these research results are funneled into the development of an experimental spoken language system. Thirdly, alternative approaches to speech recognition, including the use of artificial neural nets and strategies derived from vision research, are being explored. Finally, part of our effort is devoted to the development of the necessary infrastructure, including the development of speech research tools and databases.

11.2 Research Reports

11.2.1 Continuous Speech Recognition: The SUMMIT System

Recently, we put together a speech recognition system which embodies some of the research that we have been conducting in automatic speech recognition. The system, which we call SUMMIT, is intended to serve as a testbed for a segmental-based approach to speech recognition. In addition, it enables us to explore how speech recognition can be integrated with natural language processing in order to achieve speech understanding.

The SUMMIT system starts the recognition process by first transforming the speech signal into a representation that models some of the known properties of the human auditory system [265]. The representation is illustrated in Figure 11.1(a), for the sentence "Where is the nearest hospital?" Using the output of the auditory model, acoustic landmarks of varying robustness are located and embedded in a hierarchical structure called a dendrogram [118], as shown in Figure 11.1(b). The acoustic segments in the dendrogram are then mapped

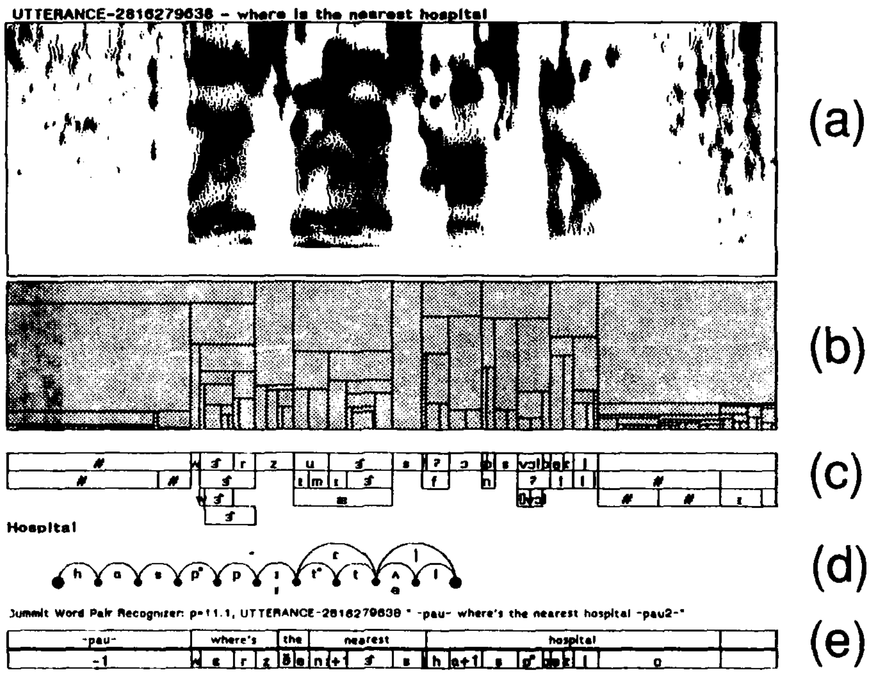

UTTERANCE-2816279638 - where is the nearest hospital
(a)
(b)
(c)
Hospital
(d)
Summit Word Pair Recognizer: p=11.1, UTTERANCE-2816279638 " -pau- where's the nearest hospital -pau2-"
-pau-
where's
the
nearest
hospital
(e)

Figure 11.1: Intermediate representation leading to the recognition of the sentence, "Where is the nearest hospital?" The display contains: (a) synchrony spectrogram, (b) a dendrogram describing the multi-level acoustic segmentation, (c) a phonetic recognition network, (d) a word pronunciation network, and (e) the recognition result.

to phoneme hypotheses, using a set of automatically determined acoustic parameters in conjunction with conventional pattern recognition algorithms [246]. The result is a phoneme network, in which each arc is characterized by a vector of probabilities for all the possible candidates, as shown in Figure 11.1(c).

Words in the lexicon are represented as pronunciation networks, which are generated automatically by a set of phonological rules. This is illustrated in Figure 11.1(d) for the word "hospital." Probabilities derived from training data are assigned to each arc, using a corrective training procedure, to reflect the likelihood of a particular pronunciation. Presently, lexical decoding is accomplished by using the Viterbi algorithm to find the best path that matches the acoustic-phonetic network with the lexical network. The recognized word string is shown in Figure 11.1(e).

We recently evaluated SUMMIT's performance in a number of ways. Phonetic classification performance was evaluated by comparing the labels provided by the classifier to those in a time-aligned transcription, using 38 context-independent phone labels [302]. This particular set was selected because it has been used in other recent evaluations within the DARPA communi y. For a single speaker, the top-choice classification accuracy was 77%. The correct label is within the top three nearly 95% of the time. For multiple and unknown speakers, the top-choice accuracy is about 70%, and the correct choice is within the top three over 90% of the time. Figure 11.2 shows the rank order statistics for both the speaker-dependent and speaker-independent cases.

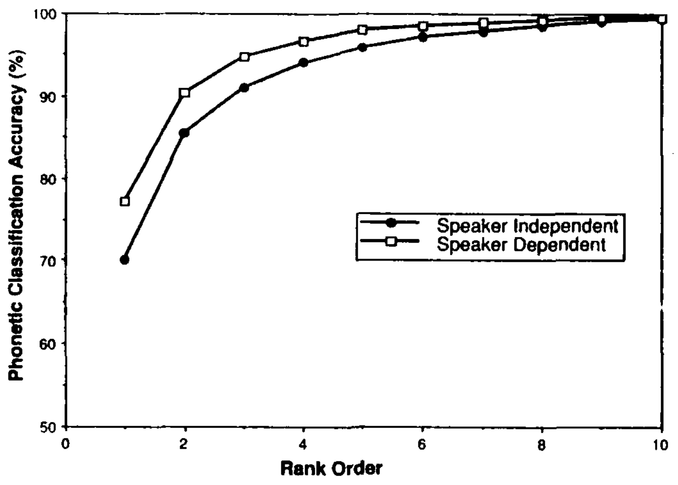


Figure 11.2: **Rank order statistics for the current phone classifier on a speaker-independent** task. There are 38 context-independent phone labels: 14 vowels, 3 semivowels, 3 nasals, 8 fricatives, 2 affricates, 6 stops, 1 flap, and one for silence.

Word accuracy for the SUMMIT system was evaluated on the DARPA 1000-word Resource Management task [301]. Two different speaker-independent test sets provided by NIST, consisting of 150 and 300 sentences, respectively, were used [241]. The SUMMIT system achieved a word accuracy of 87.1% and 86.4% on the two test sets, respectively, using the designated word-pair grammar with perplexity of 60, and approximately 70 context-independent phone models. SUMMIT's performance compares favorably with systems that are based on hidden Markov modeling, when evaluated on the same data and using a similar number of phone models [186]. Since other researchers have been able to improve their system's performance by increasing the number of models to accommodate context-dependency, we expect that we can similarly improve SUMMIT's performance.

11.2.2 Natural Language Processing: The TINA System

A new natural language system, TINA, was developed in our Group [266] which integrates key ideas from context free grammars, Augmented Transition Networks (ATN's) [296], and Lexical Functional Grammars (LFG's) [64]. TINA is specifically designed to accommodate full integration between speech recognition and natural language processing, and has a set of features reflecting this philosophy.

The grammar begins with a set of context-free rewrite rules, which are augmented with parameters to enforce syntactic and semantic constraints. These rules are converted automatically to a network form, leading to extensive structure sharing. All arcs in the network have associated probabilities, which can be trained automatically from a set of parsed sentences. The parser uses a best-first search strategy. Control includes both top-down and bottom-up cycles, and key parameters are passed among nodes to deal with long-distance movement and agreement constraints. The probabilities provide a natural mechanism for exploring more common grammatical constructions first. TINA also includes a new strategy for dealing with movement, which can handle efficiently nested and chained gaps, and rejects crossed gaps.

Over the past year, TINA has been ported to the DARPA 1000-word Resource Management task. We used the 791 designated training sentences and 200 (unseen) test sentences to evaluate our parser for coverage and perplexity. The training was a two-step process. We first expanded the coverage of the grammar until it could handle all of the 791 training sentences (100% coverage). We then built a new subgrammar from these sentences, with probabilities on arcs updated according to their usage within the training set (any rules that only appeared in the TIMIT domain were automatically discarded). This resulted in a grammar that was tightly defined for the RM task. We then tested this grammar for coverage and perplexity on the 200 test sentences. The results were that 84% of the test sentences were parsable, and the perplexity was 368 if all words that could follow each word were considered to be equally likely. The surprising result was that the perplexity dropped 9-fold when arc probabilities were incorporated into the measurement, down to 41.5. We also looked at the parses to establish the depth from the top of the correct parse. We found that 88% of the training sentences gave a correct parse as the first choice; this number increased to 90% for the test sentences. Both sets gave the correct parse within the top three over 98% of the time.

11.2.3 Spoken Language Understanding: The VOYAGER System

Over the past year, we initiated an effort in spoken language understanding. The project is motivated by our belief that many of the applications suitable for human/machine interaction using speech typically involve interactive problem solving. That is, in addition to converting the speech signal to text, the computer must also understand the linguistic structure of a sentence in order to generate the correct response.

In order to explore issues related to a fully-interactive spoken language system, we selected a task in which the system knows about the physical environment of a specific geographical area, and can provide assistance on how to get from one location to another within this area. The system, which we call VOYAGER, can also provide information concerning certain objects located inside this area. The current version of VOYAGER focuses on the geographic area of the city of Cambridge between MIT and Harvard University, as shown in Figure 11.3, and can answer a number of different types of questions about certain hotels, restaurants, hospitals, and other objects within this region.

VOYAGER is made up of three components. The first component, SUMMIT, converts the speech signal into a set of word hypotheses. The natural language component, TINA, then provides a linguistic interpretation of the set of words. The parse generated by the natural language component is then transformed into a set of query functions, which is passed to the backend for response generation. The backend is an enhanced version of the direction assistance program developed by Jim Davis of the Media Laboratory at MIT. The response generator maintains some knowledge about recent discourse history, which allows it to respond appropriately to queries such as "How do I get there?" Currently, VOYAGER can generate responses in the form of text, graphics, and synthetic speech.

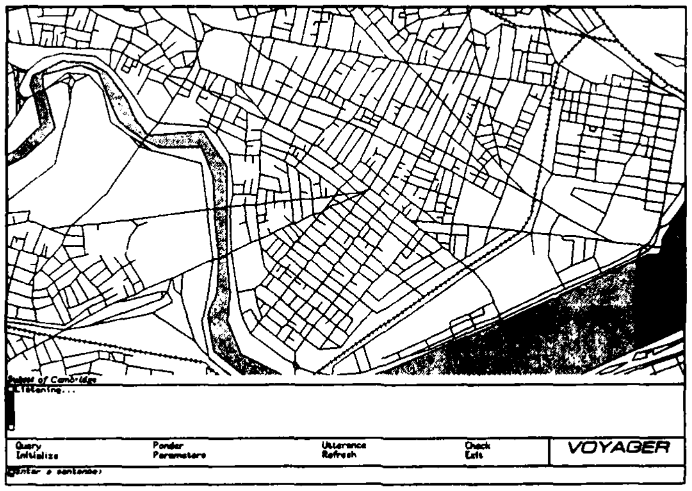


Figure 11.3: A display showing the geographical region known to the VOYAGER system.

As of now, VOYAGER has a vocabulary of over 300 words, and it can deal with about half a dozen types of queries, such as the location of objects, simple properties of objects, how to get from one place to another, and the distance and time for travel between objects. Within this limited domain of knowledge, it is our hope that VOYAGER will be able to handle any reasonable query that a native speaker is likely to initiate. As time progresses, VOYAGER's knowledge base will undoubtedly grow.

In order to evaluate VOYAGER's performance, we collected a corpus of some 5000 spontaneously spoken sentences from 100 speakers. The system was trained on approximately 70% of the data and tested on 10%. Errors in the system can occur in several ways; the recognizer can mis-recognize a word, the natural language system can fail to generate a parse, an unknown word can appear, or a query can be outside of VOYAGER's domain. All in all, the system could correctly execute approximately 50% of the queries during a recent evaluation.

The current implementation of VOYAGER makes use of a Macintosh II, augmented with DSP boards, for data capture and signal processing. Subsequent phonetic classification, lexical access, linguistic analysis, and response generation are all performed on a Sun-workstation. The overall response time is approximately 15 times real time. Refined algorithms, together with the availability of faster workstations and more powerful signal processing chips should enable the current VOYAGER implementation to run in real time in the future.

11.2.4 Phonetic Recognition Using Multi-layer Perceptrons

Over the past two years, we have been experimenting with the use of artificial neural networks (ANN) for vowel classification. Our work was motivated by the belief that such networks might offer a flexible framework for us to utilize our improved, albeit incomplete speech knowledge. Using the output of Seneff's auditory model as the input to the multi-layer perceptrons (MLP) with one hidden layer, classification accuracy ranging from 62% to 100% were achieved under varying experimental conditions. These results compared favorably to those of human listeners and traditional pattern classification techniques. These experiments helped us gain a better understanding of the behaviors of this particular kind of ANN along many dimensions, including the effects of training procedures, the nature of the hidden layer, and the use of adaptation techniques on classification accuracy. Some of these results have been documented in several publications [192], Leung-88b, Leung-89.

More recently, we have expanded our earlier work by moving towards the classification and recognition of all phonemes in American English [193]. By incorporating novel normalization and training procedures, we were able to obtain a context-independent classification accuracy of 74% for 38 phones, using as input a set of 80 automatically determined acoustic attributes. The same system configuration achieved a phonetic recognition accuracy of 55%, including substitution, insertion, and deletion errors. We are in the process of incorporating the ANN phonetic recognition module into the SUMMIT system and evaluating its impact on overall system performance.

11.2.5 Isolated Word Recognition over Telephone Network

Over the past year, we initiated an effort to develop a small-vocabulary, isolated-word recognition system. The focus of this research is to explore how our phonetically-and segmentally-based approach will fare with the bandlimited and distorted speech transmitted through local and long distance telephone networks, spoken by real users.

As a first step, we implemented a system that recognizes 25 city names, and have performed a number of recognition experiments using data from real users over dial up telephone lines collected by NYNEX. Preliminary evaluation of our system on such realistic data showed that a top-choice accuracy of 95% can be achieved with a 20% rejection criterion. We are also using this task as a framework in which to explore the use of unsupervised learning techniques to enable the automatic expansion and modification of the vocabulary.

11.3 Student Reports

Nancy Daly

Nancy is pursuing a doctoral thesis on prosodic aids for speech recognition. Prosody is the stress, rhythm, and intonation of speech. While the importance of prosodic information has long been documented for human speech communication, automatic speech recognition systems developed thus far have all but ignored this source of information. The purpose of her thesis research is to see how prosodic information could be incorporated into speech recognition systems to improve their performance.

Currently, Nancy is investigating the problem of distinguishing yes/no questions from "wh-" questions and other types of sentences. Her investigation of this problem spans several directions. First, she is attempting to document how this type of prosodic encoding is achieved by a talker, by asking listeners to categorize questions during listening tests. Next, she is investigating the intonation contour in order to establish whether a terminal high boundary tone is actually present for yes/no questions. Finally, she is interested in devising algorithms for automatic extraction of acoustic attributes, leading to the detection of these high boundary tones.

David Goddeau

Dave joined the group as a graduate student in September 1989. During the summer, he worked for the group porting the SUMMIT auditory model to C, and then optimizing to run on a DSP board. During the fall term, he contributed to porting other sections of the system to the DSP board and became familiar with the computing environment of the group. He is exploring several ideas in his search for a PhD thesis topic. The current focus of his work is the detection of unknown words in a human/machine spoken dialog.

Lee Hetherington

Lee joined the group in September 1989. He spent the fall term becoming familiar with the group's computational facilities and attended spectrogram reading classes. He is searching for a Doctoral thesis topic in the area of adaptation and learning for improved speech recognition. Lee is currently working on the problem of adding new words to the vocabulary of SUMMIT. Specifically, he will be working on unsupervised learning of the pronunciation network and/or the phonetic models when a new word is added to the vocabulary. The goal is to enable the recognizer to improve with use.

Rob Kassel

Rob just completed his Master's thesis entitled "An Information-Theoretical Approach to Studying Phoneme Collocational Constraints." Linguistic constraints are well known and well exploited at the syntactic and semantic levels. Past work in phonology has also suggested the existence of strong constraints of phoneme sequences, which are best expressed in terms of phonological equivalence classes. However, these constraints are typically stated introspectively. This research studied the co-locational constraints of phonemes using a data-driven, self-organizing approach. Even if they are quantified, the equivalence classes are usually pre-defined by linguists. Information-theoretic metrics are used to discover phonological equivalence classes that can best capture the constraints. A major goal of his research was to compare the constraining power of these equivalence classes with those of the distinctive features as suggested from phonological theory.

Jeffrey Marcus

Jeff continued work on his thesis, tentatively entitled "Incorporating Units of Different Sizes in Segment-based Speech Recognition." One goal of the thesis is to extend current techniques for combining statistical estimates of recognizer model parameters made on phonetic units of various sizes, such as phones, diphones and words. Another goal is to incorporate measurements made over acoustic segments of various sizes, for instance over phone-like and diphone-like units. This contrasts with most current speech recognizers, which make measurements over constant-duration time frames. Towards these ends, Jeff worked for some time on refining an acoustic segmentation algorithm with the aim of obtaining segments which map to phonetic units in a predictable manner. Over the next year, he will be focusing on the modeling of function words, such as "the," "are," etc. from the VOYAGER database, using the techniques mentioned above.

In addition to this research, he also worked on the problem of statistically comparing the performance of two speech recognizers. He presented this work at the European Conference on Speech Communication and Technology.

Helen Meng

Helen is interested in designing and implementing a novel representation of speech using distinctive features for speech recognition. Distinctive features are a small set of orthogonal properties used to classify both phonemes and other levels of phonology and phonetics. The compact inventory of distinctive features possesses immense descriptive power, and can concisely represent speech variations such as coarticulatory phenomena, contextual effects as well as inter-speaker differences. It is believed that these properties of distinctive features are extremely beneficial for automatic speech recognition.

Recent research has focused on the comparison of auditory-based acoustic representations. The objective of this work is to find a favorable front end for future distinctive feature extraction. The mel-based signal representations have been implemented, and compared with Seneff's auditory model on the basis of vowel classification, using the artificial neural network developed by Hong Leung. Further comparisons will be made on the basis of acoustic segmentation, and the acoustic correlates of the distinctive features will be characterized and quantified for the purpose of feature extraction. This line of investigation will lead to the transformation of an acoustic representation into a novel representation in terms of distinctive features, and the resulting recognition performance will be evaluated.

Partha Niyogi

Since joining the group in September 1989, Partha has been familiarizing himself with the research activities of the group (mainly through reading papers) and its computational facilities. In addition to his coursework, he has been attending spectrogram reading classes.

Partha has also been searching for a suitable Master's thesis topic. In particular, he is investigating how speaker-independent phonetic recognition performance can be improved by considering the correlation in the acoustic space of different phonemes due to vocal tract constraints. He will explore both clustering and normalization techniques to reduce the variance of acoustic parameters, thus leading to better recognition performance.

John F. Pitrelli

John just completed his Ph.D. thesis entitled "Hierarchical Modelling of Phoneme Duration: Application to Speech Recognition." Duration is potentially a strong cue for certain phonemic distinctions, including inherently long vs. short vowels, and voiced vs. unvoiced obstruent consonants. Phoneme durations are affected, though, by an abundance of factors ranging from detailed phonetic context effects to syntax and semantics. Our lack of understanding of these effects and their interactions hinders our use of potentially useful duration information to the extent that most speech recognition systems currently use only rudimentary duration models or use time-warping procedures, which distort duration information.

His approach to the duration modeling problem was to use a hierarchical model to account for discrete-valued factor variables, such as phonetic context features and syntactic-unit-final lengthening. Research was focused on three areas. One was to search for ways to measure speaking rate which would be suitable for use in a duration model for recognition, and to explore the effects of speaking rate on phoneme duration. The second was speech synthesis where experiments involved replacing the duration model in a speech synthesizer with the hierarchical model, and performing perceptual tests to determine whether naturalness and/or intelligibility were improved. The third area was recognition experiments designed to determine how much the duration model improved recognizer performance. John pursued two lines of experimentation in this area. One was to incorporate his duration model into an existing recognizer, to assess how much duration information adds to the performance achieved using other information. The other was to measure the potential discrimination power of duration information by evaluating a duration-only classifier on particular distinctions associated with duration effects.

Michal Soclof

Michal just completed her Master's thesis entitled "A Comparison of Spontaneous Speech and Read Speech in Human-machine Problem Solving Dialogues." The purpose of her research was to analyze and quantify the phenomena that occur in spontaneous speech, and compare them to read speech. Spontaneous speech is defined in this context as speech generated by a person when talking to a computer in a problem solving situation, as opposed to when talking to another individual. The ultimate goal of an interactive human-machine interface through speech is to enable the user to communicate with the machine using spontaneous speech. In order to build this type of system, it is necessary to study the acoustic and linguistic variations that occur in goal-directed, spontaneously uttered speech. The phenomena that occur in spontaneous speech include agrammatical and ill-formed sentences, false starts, and non-speech vocalizations.

In order to conduct this study, a large corpus of spontaneous utterances was gathered. The corpus also included a read version of each of the spontaneous sentences. The analysis of the data was divided into three categories: frequency analysis, natural language analysis, and acoustic/phonetic analysis. The frequency analysis included studying the frequency of occurrence in read and spontaneous speech of non-speech vocalizations such as mouth clicks, filled pauses, unfilled pauses, and false starts. The natural language analysis entailed finding the location of the spontaneous phenomena within the sentence structure. The acoustic/phonetic analysis was done on the sentence, word and phoneme level. This included studying the duration of the read vs. spontaneous sentences, and the duration of filled and unfilled pauses.

11.4 Publications

[1] S. Seneff. Probabilistic Parsing for spoken language applications. In *Proceedings of the International Workshop in Parsing Technologies*, Pittsburgh, PA, 1989.

[2] J. Pitrelli. A hierarchical model for phoneme duration in american english. In *Proceedings of European Conference on Speech Communication and Technology*, pages 324-327, Paris, France, 1989.

[3] J. Marcus. Significance tests for comparing speech recognizer performance using small test sets. In *Proceedings of European Conference on Speech Communication and Technology*, pages 465-468, Paris, France, 1989.

[4] V. Zue, S. Seneff, and J. Glass. Speech database development: TIMIT and beyond. In *Proceedings of the Workshop on Speech Input/Output Assessment and Speech Databases*, Amsterdam, The Netherlands, 1989.

[5] V. Zue, J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, and S. Seneff. The VOYAGER speech understanding system: a progress report. In *Proceedings of the Second DARPA Speech and Natural Language Workshop*, pages 51-59, Harwichport, MA, 1989.

[6] V. Zue, J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, S. Seneff, and M. Soclof. The collection and preliminary analysis of a spontaneous speech database. In *Proceedings of the Second DARPA Speech and Natural Language Workshop*, pages 126-134, Harwichport, MA, 1989.

[7] V. Zue, J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, and S. Seneff. Preliminary evaluation of the VOYAGER spoken language system. In *Proceedings of the Second DARPA Speech and Natural Language Workshop*, pages 160-167, Harwichport, MA, 1989.

[8] V. Zue, J. Glass, D. Goodine, M. Phillips, and S. Seneff. The SUMMIT speech recognition system: phonological modeling and lexical access. In *Proceedings of ICASSP*, pages 49–52, Albuquerque, NM, 1990.

[9] V. Zue, J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, and S. Seneff. The VOYAGER speech understanding system: preliminary development and evaluation. In *Proceedings of ICASSP*, pages 73-76, Albuquerque, NM, 1990.

[10] H. Leung and V. Zue. Phonetic classification using multi-layer perceptrons. In *Proceedings of ICASSP*, pages 525-528, Albuquerque, NM, 1990.

Thesis Completed

[1] R. Kassel. *An Information-theoretical Approach to Studying Phoneme Collocational Constraints.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990. Supervised by V.W. Zue.

[2] A. Lim. *A New Approach to Speech Recognition in the Presence of Co-channel Speech Interference.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990. Supervised by G. Kopec and V.W. Zue.

[3] K. Ng. *A Comparative Study of the Practical Characteristics of Neural Network and Conventional Pattern Classifiers.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990. Supervised by R.P. Lippmann and V.W. Zue.

[4] J. Pitrelli. *Hierarchical Modelling of Phoneme Duration: Application to Speech Recognition.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, May 1990. Supervised by V.W. Zue.

[5] M. Soclof. *A Comparison of Spontaneous Speech and Read Speech in Human-Machine Problem Solving Dialogues.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990. Supervised by V.W. Zue.

[6] D. Whitney. *Building a Paradigm to Elicit a Dialog with a Spoken Language System.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990. Supervised by V.W. Zue.

Theses in Progress

[1] N. Daly. *Prosodic Aids to Speech Recognition.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected 1991. Supervised by V.W. Zue.

[2] D. Goddeau. *Detecting and Parsing Unknown Words in Speech Understanding Systems.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected 1991. Supervised by V.W. Zue.

[3] I.L. Hetherington. *Supervised and Unsupervised Incremental Training in Speech Recognition.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected 1992. Supervised by V.W. Zue.

[4] J. Marcus. *Building Acoustic Models of Words and Phrases for Speech Recognition Using Variable-sized Lexical and Acoustic Units and a Data Analytic Approach.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, expected 1991. Supervised by V.W. Zue.

[5] H. Meng. *A Comparison of Auditory-based Representations of Speech and the Use of Distinctive Features as an Intermediate Representation for Automatic Speech Recognition.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected 1990. Supervised by V.W. Zue.

[6] P. Niyogi. *A Study of the Correlation in Acoustic Space of Different Phonemes Due to Vocal Tract Constraints.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, expected 1991. Supervised by V.W. Zue.

Talks

[1] V.W. Zue. Yesterday, today, and tomorrow: on the state of voice I/O technology. Keynote Address made at the at the Annual Meeting of the American Voice Input/Output Society, Newport Beach, CA, September 1989.

[2] V.W. Zue. MIT's Voyager spoken language system. Distinguished Lecture given at Oregon Graduate Center, Beaverton, OR, November 1989.

[3] V.W. Zue. Computers that listen: recent progress at MIT. Lecture given at MIT EECS Colloquium, November 1989.

[4] V.W. Zue. Dennis Klatt's contribution to automatic speech recognition. Invited lecture given at the 118th Meeting of the Acoustical Society of America, St. Louis, MO, November 1989.

[5] S. Seneff. A natural language system for spoken language applications. Invited lecture given at the ATR Symposium on Basic Research for Telephone Interpretation, Kyoto, Japan, December 1989.

[6] V.W. Zue. Future directions for spoken language research. Invited lecture given at the ATR Symposium on Basic Research for Telephone Interpretation, Kyoto, Japan, December 1989.

[7] V.W. Zue. Computers that listen: recent progress at MIT. MIT ILO Seminar, Tokyo, Japan, February 1990.

[8] V.W. Zue. Assessment of speech I/O technology. Keynote Address given at the at the Citicorp-TTI Voice Technology Seminar, Santa Monica, CA, March 1990.

[9] V.W. Zue. Computers that listen: recent progress at MIT. Lecture given at the MIT Industrial Liaison Program and MIT International Financial Services Research Center Seminar, Cambridge MA, May 1990.

Systematic Program Development

Academic Staff

J. V. Guttag, Group Leader

Research Staff

A. S. Esterkin	S. J. Garland

Graduate Students

D. N. Jackson	M. B. Reinhold
M. T. Vandevoorde	K. A. Yelick

Visitor

M. C. Godfrey

Support Staff

A. Rubin

12.1 Objectives

Our main goal is to develop techniques and tools that will facilitate the efficient production of high quality software. Most of our attention is concentrating on exploring the role of formal specifications in building programs from components, encouraging module-level reuse, and allowing early detection of design errors.

12.2 Approach

Effective programming involves choosing abstractions that can be combined to solve a problem, specifying their meanings, and implementing them. It is better to think about combining abstractions than about combining implementations because: 1) specifications are easier to understand than implementations, 2) software is easier to maintain if it relies only on properties guaranteed by specifications, and 3) components are more likely to be reusable if a distinction is made between abstractions and implementations.

The Larch family of specification languages is unique in the way it supports a two-tiered definitional approach to specification. Each specification has components written in two languages: one designed for a specific programming language and another independent of any programming language. The former are called Larch interface languages, and the latter the Larch Shared Language (LSL). Larch allows one to specify reusable theories (at the LSL-level), as well as interfaces to actual software components.

The Larch style of specification emphasizes brevity and clarity rather than executability. To make it possible to test specifications without executing or implementing them, Larch permits specifiers to make claims about logical properties of specifications and to check these claims at specification time. The emphasis in our tools is thus on reasoning at the component level, rather than at the code level. The most interesting tool is LP, a system used for reasoning about semantic properties of specifications.

Our work on parallel programming is related to our work on specifications in that it emphasizes module-level programming. Various methods have been proposed to address the problem of writing and reasoning about parallel programs, but these tend to address only single module programs. Notions of module composition are missing, and as a result, a style of program development is encouraged in which the entire program is designed and implemented as a single unit. Conversely, methods that have been developed for "programming in the large" of sequential programs are not applicable, and attempts to extend them to parallel programs often result in programs that exhibit very little real concurrency. Our goal is to be able to decompose parallel programs into independently specifiable units without prohibiting efficient implementations.

12.3 Recent Accomplishments

We released version 2.0 of the LP system for reasoning about specifications. LP is being used for reasoning about specifications of software components at MIT, CMU and Digital Equipment Corporation's System Research Center (DEC SRC) and for reasoning about circuit descriptions at DEC SRC and the Technical University of Denmark.

We completed a description of how to translate and formulate the necessary semantic checks for all of LSL using LP. This will make it possible to begin serious experiments with the use of LP in debugging specifications.

We developed a novel approach to designing and implementing efficient, yet modular, parallel programs. This approach was used to implement one small and one reasonably large parallel program.

12.4 Plans for Current Year

We expect to complete the design and preliminary implementation of a Larch/C interface language. By the end of the fiscal year, we hope to have used it to specify an interesting set of reusable components that can be used by C programmers.

We plan to write up our approach to building parallel programs. This will include a careful evaluation of the performance of the programs we have built using it, and a discussion of the strengths and weaknesses of the approach.

We plan to enhance LP by adding efficient special purpose reasoning procedures. These will be aimed at specific theories that have proved critical to current users of LP.

We plan to complete a new LSL implementation and connect it to both LP and the Larch/C processor.

12.5 Collaboration Outside LCS

We have collaborated for many years with researchers at DEC SRC on the design of the Larch family of languages. This collaboration will continue. SRC has been a primary Beta site for testing our software. They contributed many valuable suggestions that were incorporated in the current release of LP. Finally, we have worked with SRC on the specification of threads.

We have been working with researchers at CMU on the design of Larch/C. Researchers at CMU have used Larch to specify a variety of software components. They have also used LP to reason about those components.

Researchers at the Technical University of Denmark have used LP to reason about descriptions of circuits. They currently have software that allows them to generate input to a fabrication facility from descriptions written in a high level circuit description language. They are currently writing a compiler to compile descriptions written in their language into input suitable for LP (up to now the translation has been done by hand). Last summer they fabricated a chip that was verified, using LP.

12.6 Publications

[1] S.J. Garland, J.V. Guttag, and J.J. Horning. Debugging Larch Shared Language specifications. *IEEE Transactions on Software Engineering*, September 1990. To appear.

[2] J.V. Guttag, J.J. Horning, and A. Modet. *Revised Report on the Larch Shared Language.* Digital Equipment Corporation Systems Research Center Report 58, July 1990.

[3] M.B. Reinhold. *Type Checking is Undecidable When "Type" is a Type*, Technical Report MIT/LCS/TR-458, MIT Laboratory for Computer Science, December 1989.

[4] J.S. Staunstrup, S.J. Garland, and J.V. Guttag. Localized verification of circuit descriptions. *In Proceedings of an International Workshop on Automatic Verification Methods for Finite State Systems*, Grenoble, France. Also Lecture Notes in Computer Science, 407:349–364 Springer-Verlag, 1989.

[5] P. Zave and D. Jackson. Practical specification techniques for control-oriented systems. *In Proceedings of the IFIP 11th World Computer Congress*, San Francisco, CA, 1989.

Theory of Computation

Academic Staff

B. Awerbuch	P. Elias	S. Goldwasser
L. Guibas	T. Leighton	C. Leiserson
N. Lynch	M. Karchmer	A. Meyer
S. Micali	R. Rivest, Group Leader	D. Shmoys
M. Sipser	E. Tardos	

Visitors and Post-Docs

A. Aggarwal	S. Cosmadakis	S. Istrail
M. Kearns	Joe Kilian	Leonid Levin
B. Maggs	R. Meunier	N. Nisan
G. Plaxton	A. Rabinovich	S. Safra
N. Shavit	J. Spencer	L. Stockmeyer
B. Trakhtenbrot		

Graduate Students

J. Aslam	M. Bellare	B. Berger	M. Betke
B. Bloom	A. Blum	B. Blumofe	T. Cormen
L. Cowen	C. Crépeau	A. Dhagat	B. Eisenberg
M. Ernst	L. Fortnow	D. Gillman	W. Goddard
S. Goldman	K. Graves	R. Greenberg	M. Grigni
M. Hansen	R. Hirschfeld	A. Ishii	L. Jategaonkar
T. Jim	N. Kahale	S. Kipnis	M. Klugerman
R. Koch	D. Kravets	A. Lent	J. Leo
Y. Mansour	M. Mohtashemi	M. Newman	C. Norton
R. Ostrovsky	M. Papaefthymiou	J. Park	C. Phillips
S. Rao	J. Riecke	P. Rogaway	J. Rompel
A. Rudich	R. Schapire	L. Schulman	E. Schwabe
M. Singh	M. Smith	C. Stein	D. Wald
J. Wein	D. Williamson	Y. Yin	

Undergraduate Students

J. Fernandez	S. Arora	D. Lisinski

Technical and Administrative Staff

W. Ang	B. Tilson

Support Staff

S. Bemus	C. Brownlie	B. Dale
D. Grupp	B. Hubbard	D. Jones
D. Sergent	E. Yang	

13.1 Introduction

The MIT Theory of Computation (TOC) group is one of the largest theoretical computer science research groups in the world. It includes faculty, students, and visitors from both the Electrical Engineering and Computer Science Department, and the Applied Mathematics Department.

The principal **research areas** investigated by members of the TOC Group are:

- algorithms: combinatorial, geometric, graph-theoretic, number theoretic;
- cryptology;
- computational complexity;
- parallel computation;
- distributed computation: algorithms and semantics;
- machine learning;
- semantics and logic of programs; and
- VLSI design theory.

13.2 Faculty Reports

Baruch Awerbuch

Awerbuch worked on a number of topics related to design and analysis of communication protocols.

Also, Awerbuch worked on many specific problems in dynamic networks. Together with Shavit and Mansour [28], Awerbuch discovered the first polynomial solution to the end-to-end communication problem. This is one of the basic network problems; it was conjectured in [3] that it has no polynomial solution. Together with Goldreich and Herzberg (Technion), [27] he developed a quantitative framework for analyzing performance of broadcast protocols in dynamic networks. Together with Kutten (IBM) and Cidon (IBM), he discovered an efficient algorithm for maintaining topology [25] in a dynamic network. In [24], those techniques have been modified to yield efficient control mechanisms for new generation, fast hardware-switched networks developed in IBM.

Together with Goldberg (Stanford), Luby (ICSI), and Plotkin (Stanford) he found a new technique [26] for removing randomness from distributed computing that has yielded fast deterministic algorithms for Maximal Independent Set, $\Delta + 1$ Coloring and Breadth First Search. Together with Peleg (Weizmann) and Baratz (IBM), Awerbuch developed a new framework for cost-sensitive analysis of distributed algorithms [23].

The emphasis in Awerbuch's work is distributed data structures and their application. In [29], Awerbuch formalized the problem and gave the first solution to the problem of online dynamic directories, in which both updates and searches are efficient. Also, he showed that network routing can be performed with compact routing tables, without causing a significant increase in communication [30].

Peter Elias

The paper on error-correcting codes under list decoding, which appeared as a Technical Report at the time of the last progress report, has since been accepted for publication [104].

The work on iterative error-correcting coding schemes referred to in the last progress report, exploring of the effect on performance on reusing the low order check bits after higher order bits have been used, has led to analytical and simulation results which show that the reuse of check bits brings performance much closer to channel capacity. That work has been deferred for most of the past year while the author was pursuing sociological research. A preliminary report of the sociological work has come out in two parts [279][278], and has been accepted for publication. Work on the iterative codes will resume during the summer of 1990.

Shafi Goldwasser

Research Topics

Fairness in Distributed Computation

Together with Leonid Levin, we extended some of our previous results on fairness in distributed computation started in 1988-89 as outlined below.

In 1988-89, Goldwasser together with Beaver [39] investigated the problem of performing a distributed computation in a network with broadcast channels and any number of faulty processors. It was shown how the existence of an oblivious transfer protocol is a necessary and sufficient condition to compute any polynomial time boolean function defined on the processors private inputs privately, correctly, and with a novel property: *fairness*. The faulty processors can find out the function value "if and only if" the non-faulty processors find out the function value, in a a certain technical probabilistic sense.

It was left open, whether the same is true for non-boolean functions. Together with Levin [129], we resolve this problem. We show how to compute any function from strings to strings privately, correctly, and fairly for a definition of fairness extended to the non-boolean case.

In the same work, a set of new definitions for "what should be desired from a fault-tolerant computation" in presence of malicious faults is proposed. The new definitions are more general and yet simpler than the ones used previously in this field. We show that our new requirements are achieved by previous solutions of [128] and others, and imply all properties required by previous definitions.

Randomness in Interactive Proofs

Interactive proofs [130] are an extension of the classical NP notion of efficient provability in which nondeterminism is enhanced by two new ingredients: *randomness* and *interaction*. While the verifier in an NP proof is deterministic, the verifier in an interactive proof is allowed to flip coins. In addition, the prover and the verifier can interact for a polynomial number of rounds of message exchange. Recent results by Fortnow, Karloff, Lund, Nisan [114], and Shamir [267] showing that IP=PSPACE suggest that adding the new ingredients indeed enlarges the class of languages which can be efficiently recognized.

Furthermore, both of the new ingredients are necessary: if the verifier were not probabilistic, then the prover could simply simulate all the moves of the verifier on his own and IP would collapse to NP; in the absence of interaction, IP would collapse to BPP.

An important question then is *how much* of these resources are really necessary. How much interaction is necessary has received much attention (see for example, [31] and others). How much *randomness* is necessary for IP has, on the other hand, received no attention.

Together with Bellare, Rompel, and Goldreich [42], we started an investigation into how much randomness is necessary to recognize a language $L \in \text{IP}$ in a given number of rounds and a given error probability. In particular, we show how to transform an interactive proof system for L where tosses l coins per round and error probability $\frac{1}{3}$ to an interactive proof system for L with the same number of rounds, error probability $\frac{1}{2^k}$, the verifier tosses $O(l+k)$ coins per round. Previous transformation required $O(lk)$ coins per round.

Professional Activities

During 1989-90, Goldwasser was an invited speaker to the short course on "Number Theory and Cryptography" held by the AMS in Boulder, CO during August. The lecture notes will be assembled into a book by the American Mathematical Society. Goldwasser was also an invited speaker to the British Colloquium on computer science held in Manchester, Britain in March 1990, and an invited speaker to the International Congress of Mathematicians (ICM) in Japan, August 1990.

Goldwasser also participated in the Oberwolfach Conference on Cryptography, Germany, October 1989; the DIAMICS Workshop on Distributed Computing and Cryptography, October 1989; and Workshop on Circuit Complexity in Barbados, organized by McGill University, February 1990.

Goldwasser edited "Advances in Cryptology: CRYPTO'88," published by Springer-Verlag, which appeared in February 1990. The book contains the proceedings of the CRYPTO88 Conference held in Santa Barbara in August 1980 for which she was a Chairperson. In addition, Goldwasser continued to be an editor for *SIAM Journal of Computing, Journal of Cryptology,* and a new journal on the foundations of computer science.

Finally, Goldwasser is writing up two sets of of lecture notes. The first on computational number theory and cryptography (a result of the short AMS course), and the second a manuscript for an introductory cryptography classes.

Leonidas Guibas

During the past year, Guibas together with several coworkers, studied certain natural combinatorial questions in geometry. The results obtained are of the following form: if we are given points in some Euclidean space and many "nice" objects, each "defined" by only a few of the points, then a large number of these objects must intersect at a common point (not necessarily one of the given points). For instance, if we are given n sites on the real line, and $m \geq 2n$ intervals defined by pairs of these sites, then there has to be a point of the real line contained in at least $m^2/4n^2$ of the intervals.

These covering results lead to many interesting consequences in combinatorial and computational geometry. They can be used to show that, given any set P of n points in space, there exists another set Q of roughly $n^{1/2}$ points such that the Delaunay triangulation of $P \cup Q$ has size roughly only $O(n^{3/2})$, even though that of P itself can have size $\Omega(n^2)$ [70]. Furthermore, the set Q can be efficiently computed from P. Thus in three dimensions, if a point set has an unacceptably large Delaunay triangulation, we can always add some extra points and *reduce* the size of the triangulation. Such results are important in building finite element codes.

Another application of the packing results yields the following: for any set S of n points in the plane, and any $n^{3-\alpha}$ triangles spanned by these points, there exists another point (not necessarily of S) contained in at least (roughly) $n^{3-3\alpha}$ of the triangles. This implies that any set of n points in three dimensional space has at most (roughly) $n^{8/3}$ halving planes [14]. This is a big improvement on the previously best known upper bound for one of the most famous problems in combinatorial geometry.

On a different front, Guibas, together with some ex-students and coworkers from Japan, continued the development of geometric algorithms based on the *topological sweep* paradigm introduced by him and Edelsbrunner [101] a few years back. New results this year include an extension of the topological sweep algorithm to handle an arrangement of planes in three dimensions [11], as well as a delicate refinement of the planar method that is capable of sweeping the portion of an arrangement inside a convex region in time proportional to the complexity present in that region [15]. Both of these extensions have many applications that are given in the referenced papers.

Guibas and coworkers at DEC/SRC continued previous work on developing a framework for building robust geometric algorithms. They developed a proof that, given any set of points in the plane, there exists a polygon that is convex enough to be found with approximate tests, such as floating-point arithmetic, and is also very close to tightly containing all the points of the given set. Such a polygon is a very useful approximation to the convex hull of the points in an environment (such as all real geometry systems) where one needs to operate with primitives of limited precision. They also developed an algorithm for finding this approximate convex hull [137]. The computation of convex hulls (this time in an infinite precision model) also gave rise to a new data structure called *compact interval trees.* This is a structure that allows one to compute very quickly convex hulls of subpaths of a given simple polygonal path in the plane. This and related results are given in [135].

Finally, Guibas, Knuth, and Sharir developed an extremely simple new incremental algorithm for computing Voronoi diagrams or Delaunay triangulations of point sets in the plane. The algorithm takes optimal time $O(n \log n)$ if we randomize the sequence of insertion of the sites. It also has many other pleasant properties, including the fact it allows one to build a point-location structure for searching the corresponding diagram without additional effort [136].

Service to the profession: During the past year Guibas served on the Program Committee for the SIGGRAPH '90 ACM Conference, and continued to serve on the editorial board of nine journals.

Tom Leighton

Leighton and his colleagues continued to make substantial progress on packet routing and sorting algorithms, fault-tolerance in networks, and on approximation algorithms for NP-complete problems.

Highlights of this year's research include the discovery of the first $O(\log N)$-step algorithm for online routing in a nonblocking network (joint with Sanjeev Arora and Bruce Maggs), and the discovery of a simple sorting circuit with depth $7.5 \log N$ that works for almost all inputs (joint with Greg Plaxton). Both results appear to be practical and could well have applications to the design of parallel machines and communications networks. In particular, Leighton and Maggs are working with members of Tom Knight's group on the design of a parallel network for the Transit Machine.

The new ACM Symposium on Parallel Algorithms and Architectures got off to a great start with its first meeting in Santa Fe last year, combining the best of theory and practice in the parallel area. This year's meeting will be in Crete, and an excellent collection of papers has already been selected for presentation at the meeting. Leighton will continue to serve as Conference Chair for the symposium until the 1992 meeting. The 1991 meeting will be in late July on Hilton Head Island, South Carolina.

Leighton continues to write his book on parallel algorithms and architectures (ever so slowly), and he is optimistic that with the addition of Bruce Maggs as coauthor, the project can be completed this year!

Both Mark Hansen and John Rompel will complete their theses this year, and have done a great job. Mark is going into business, and is off to a great start after winning a $5,000 prize in the MIT entrepreneurship contest. John Rompel, on the other hand, became the first student ever to win two best student paper awards at the FOCS and STOC conferences. John's awards were for his discovery of methods for derandomizing parallel algorithms (joint with Bonnie Berger), and for his design of secure digital signature schemes.

Charles E. Leiserson

Leiserson's research is currently focusing on the problems of building very large scale computers having, perhaps, more than a billion processors. In addition, he has been studying parallel algorithms and architectures, as well as phenomena in VLSI circuitry. A highlight of his research this year is an algorithm developed with his student Alexander Ishii that provides the first polynomial-time test for correct functioning of level-clocked circuitry [149]. Also, much of Leiserson's earlier work on digital circuit retiming with James B. Saxe of Digital Equipment Corporation was revised and published [189].

Leiserson also completed writing his textbook [80] *Introduction to Algorithms*, coauthored by Ronald Rivest and Thomas Cormen. The book offers a comprehensive, but elementary, introduction to the analysis of computer algorithms. It is published jointly by the MIT Press and McGraw-Hill Book Company.

Three of Leiserson's Ph.D. students completed their degrees in the past year. Ronald I. Greenberg completed his thesis *Efficient Interconnection Schemes for VLSI and Parallel Computation*, and assumed an assistant professorship at the University of Maryland. Bruce

M. Maggs completed *Locality in Parallel Computation*, and became a postdoc in the group. Cynthia A. Phillips's thesis is entitled *Theoretical and Experimental Analyses of Parallel Combinatorial Algorithms*, and she assumed a research position at Sandia National Laboratory in New Mexico.

Leiserson is currently supervising Bobby Blumofe, Thomas Cormen, Alexander Ishii, Shlomo Kipnis, and James Park.

In September, Leiserson assumed leadership of the newly formed VLSI and Parallel Systems Group at MIT when Paul Penfield, who had run the Microsystems Research Center, became head of the EECS Department. The programs run by the MRC were divided between VPS and the Microsystems Technology Laboratories. The group currently has eight faculty drawn from three MIT Laboratories: Artificial Intelligence Laboratory, Laboratory for Computer Science, and Research Laboratory for Electronics. The goal of the VPS Group is to understand how integrated circuit technology can be applied to the design and construction of high performance computer systems.

The VPS Group ran four important programs last year. The Sixth MIT VLSI Conference in April brought together technical leaders in the university, industry, and government communities. The attendees agreed it was one of the strongest conferences we have had. Jointly with the Microsystems Technology Laboratories, the VPS Group also supported the MIT VLSI Seminar Series, the MIT VLSI Research Review, and the MIT VLSI memo series.

Nancy Lynch

Please see her entry under the Theory of Distributed Systems chapter.

Albert R. Meyer

Meyer's research focuses on semantics and logic of programming languages. During the past year, he worked on the following particular research topics.

Research Topics

- *Semantics of Concurrency*: Meyer and Bloom continued their study of basic notions of concurrent process equivalence [56][55][57][58].
- *Semantics of Terminating Evaluation.* Research with Riecke and Cosmadakis (IBM Watson Research Center) on the "lazy" lambda calculus which repairs the mismatch between classical semantics in which expressions M and $\lambda x.M$ mean the same thing, even though evaluation of M may diverge while evaluation of $\lambda x.M$ always terminates immediately, cf. [223][59][81]. In [82], they demonstrate completeness and decidability of an axiom system for equational sequents involving pure, simply typed terms of the "lazy" lambda calculus.
- *Dataflow Semantics*: See the report of Arie Rudich.
- *Theory of Sequential Functions*: See the report of Trevor Jim.
- *Type-checking for records with inheritance*: See the report of Lalita Jategoankar.

Professional Activities

- Co-chair, "International Conference on Theoretical Aspects of Computer Software," Sendai, Japan, September 1991.
- General Chair, IEEE Symposium on Logic in Computer Science (LICS).
- Moderator for three computer science research email forums on (1) types, (2) concurrency, and (3) logic.
- Member, Program Committee, International Symposium on Logic at Botik, Pereslavl-Zalessky, USSR, July 1989; "Kleene '90" Logic Symposium, Chaika, Bulgaria, June 1990.
- Member, NSF Computer Science Advisory Board.
- Thesis Supervision:

  Ph.D. 1. Bard Bloom, completed August 1989.
  2. Jon Riecke.
  3. David Wald.
  4. Lalita Jategoankar.

  S.M. 1. Michael Ernst.
  2. Arthur F. Lent.
  3. Lalita Jategoankar, completed August 1989 [151].
  4. Trevor Jim.
  5. Arie Rudich, completed May 1990 [256].

  S.B. 1. Arthur F. Lent, completed January 1990 [190].

- Editorial Activity:

  Editor-in-Chief, *Information and Computation*; Managing Editor, *Annals of Pure and Applied Logic*; Co-editor, MIT Press *Foundations of Computing Series*; Editorial Board Member, *SIAM Journal of Computing*, *Journal of Computer and System Sciences*, *Theoretical Computer Science*, and *Advances in Applied Mathematics*; Advisory Editor, *Handbook of Logic in Computer Science* and *Handbook of Theoretical Computer Science*; Co-editor, *Proceedings of Logic at Botik* [224]; and Member, MIT Press Editorial Board.

Silvio Micali

Most of Micali's research efforts have been devoted to develop theory in the area of secure protocols. Here, there is good news and bad new. The good news is that we are dealing with a novel and exciting subject that may prove very useful in an electronic world. The bad news is that the field is very difficult and largely unexplored.

A program is already a very difficult beast to tame and deciding whether a given program is correct not only is impossible in general, but is also "impossible" in practice. Worse yet, a protocol is a program run by many parties, some of which may cooperate in disrupting the joint effort! In this situation, not only is it difficult to find correct protocols, but also to discuss what correctness should mean. We were fortunate, though, to make progress on finding the right notion of security for multiparty protocols and a theoretical study of how efficient can these protocols be made. We describe some token results in this line.

The Right Notions. The field of secure protocols has quickly outpaced its foundations. Valuable definitions for secure protocols have been given (in various degrees of explicitness) by many researchers—including us. However, these definitions differed among themselves in subtle but crucial ways, and were sometimes designated protocols "secure" that should not be so. analysis. In [166], we finally provide an important step in order for foundational issues to catch up. Namely, we distill the proper notion of secure computation, and endow the field with a sound and most general definition of it.

Minimizing Resources. Having clarified what secure protocols should be, we turned our attention to investigating what are the minimum amount of resources for indeed implementing protocols in a secure way.

In [40], we consider the following problem. Assume we have n parties, 1 through n; each party i has a private input x_i known only to him. The parties want to evaluate correctly a given function f on their inputs—that is to compute $y = f(x_1, ..., x_n)$—while maintaining the privacy of their own inputs. That is, they do not want to reveal more than the value y implicitly reveals. This problem is the archetypal secure protocol problem. It was known to be solvable before, but the number of rounds of communication for correctly and privately evaluating f grew with the complexity of the function f. This was indeed a drawback, since one can perform a lot of cryptographic computation in a few seconds, but the time to send e-mail back and forth a thousand times easily requires a month. Fortunately we have been able to show that there exists a protocol for correctly and privately evaluating any function, independently of its complexity, in a constant number of rounds (actually 16.)

Sometimes, there are circumstances in which not even a constant number of communication rounds is deemed practical. What then? In [43], we show that fundamental protocols, like the oblivious transfer, can be implemented securely without any interaction at all! In essence, we exhibit a method to compute special, matching encoding-decoding key pairs. User A, having produced such a pair (P,S), will publish P and keep secret S. At this point, a sender B can look up P, make sure that it satisfies a given condition and encode two message m_1 and m_2, using the public encoding key. B is guaranteed that user A, thanks to his secret decoding key S, will be able to decode exactly one of these two encrypted message: either m_o or m_1, at random. Moreover, the sender B will not know which of the two messages A succeeded in reading. An exchange like the one discussed above is called an oblivious transfer. It is a fundamental primitive in protocol design and it was previously believed to require a lot of interaction for achieving its "paradoxical" constraints. Instead, we show that A and B do not need to interact, except for A publishing—once and for all—the public, encoding key. This result improves a previous one of Bellare and Micali where, however, the transfers were not independent. That is, if B sent to A two more messages, and A succeeded the first time around in reading the first message, then he will read the first message also the second time.

In [165], we minimize the resources needed in a zero-knowledge proof. These are the number of ciphertexts (as each ciphertext involves an overhead in number of bits) end the number of rounds of interaction. We show that two ciphertexts are enough. Also, we show that after a small pre-processing step, given any one-way function, one can prove an unbounded number of theorems in zero knowledge, each without interaction. That is, each zero-knowledge proof consists of a single message, from the prover to the verifier, to which the verifier needs not to respond. Thus, the proving process is extremely efficient, once the preprocessing step is done. Moreover, such a preprocessing step depends logarithmically on the desired probability of error; ch is also very efficient. Previous methods also exhibited a polynomial dependence on the size of the theorem (i.e., the number of bits in its statement—which can be very large).

Additional results on minimizing the resources in zero-knowledge protocols can be found in references.

Making it easy. In [46], we show how the design of cryptographic protocols can be vastly simplified. Essentially, we exhibit a cryptographic compiler that, given a protocol that is secure with respect to an honest participant, returns a protocol that is secure even against an arbitrarily cheating one.

Make it efficient. In [106], we address the problem of efficiency for the fundamental primitive of digital signatures. Computing a digital signature is feasible for the legal signer, though not trivial. However, it should be infeasible for a forger. An RSA signature with a 512-bit security parameter appears hard to forge with current knowledge about factoring algorithms, and can be easily computed in a few seconds. However, should the difficulty of, say, integer factorization substantially decrease, then signing will require minutes even to the legal signer. This is unpleasant, since the signing of a message can start after the message is chosen. Thus, we develop an alternative approach to digital signature that allows us to transfer the bulk of the computation in an offline stage that can be performed when the message of interest is not yet chosen. Online, that is after the message has been selected, a trivial amount of work will be required instead.

One of the proposed directions of research is finding how much security is obtainable in protocols without relying on complexity theory. Previous work by Goldwasser, Ben-Or, Wigderson and Chaum, Crepeau and Damgard showed that having physically secure communication channels and an honest majority may be enough. Recently Kilian and Micali [163] showed that one can do without an honest majority, by using some simple physical mechanism like an urn and ballots.

Ronald L. Rivest

Rivest's research focuses primarily on the theoretical aspects of machine learning.

Together with Javed Aslam, Rivest investigated the problem of inferring the structure of a Markov chain from its output. The special case of inferring the structure Markov chains, where each node has degree at most two, has been effectively handled by a new algorithm.

Together with Bonnie Eisenberg, Rivest investigated the power of "membership queries" for learning (in the distribution-free sense of Valiant) various concept classes. A general theorem has been proven showing that for many concept classes membership queries do not improve the efficiency of learning.

Rivest supervised an S.B. thesis of Robert T. Adams on the problem of inferring Lisp programs from examples, in which queries to the user are used to resolve ambiguities that arise during the inference process.

Together with Tom Cormen and Charles Leiserson, Rivest has finished an introductory text on algorithms [80]. This text should be suitable for both introductory undergraduate and introductory graduate courses. Rivest developed a new one-way hash function, called "MD4," which he proposed for adoption as a standard "message digest" algorithm.

Michael Sipser

Sipser's work continues to focus on aspects of complexity theory, including the structure of complexity classes and proving lower bounds on the complexity of specific problems.

Together with Michelangelo Grigni, Sipser [134] proposed a system of monotone complexity classes, paralleling the familiar system. This includes monotone analogs of the classes P, NP, L, NL, NC, and AC0 among others. This allows a succinct statement of a number of the previous known results in the area of monotone complexity, as well suggesting several new lines of research. Monotone computation is of interest because it provides a nontrivial setting where the analogs of a number of the long-standing unsolved problems in computational complexity theory may be settled, possibly shedding some light on the case for general computation. One of the new results that has been obtained has been to show that the monotone counterpart to the class NL, is not closed under complementation, in contrast with the recent results of Immerman and Szelepcseny. One may view this as an argument that their simulation must be inherently more complicated than many of the older simulations, which also go through for the monotone case.

Together with Ravi Boppana, Sipser has completed a survey of recent work on the complexity of finite boolean functions [63]. This will appear in the forthcoming *Handbook of Theoretical Computer Science.*

Last August, Sipser participated in a conference at the University of Chicago organized around a series of ten lectures that he gave on circuit complexity and the P versus NP question. The meeting was quite successful, drawing over one hundred participants.

13.3 Student, Research Associate, and Visitor Reports

Javed A. Aslam

Aslam has been working with Ron Rivest on the inference of random walk processes. A random walk process is modeled as a graph where nodes represent states and edges represent transitions from state to state. If the edges are colored, then the output of a random walk process is the sequence of colors corresponding to the transitions taken in a random walk on the underlying graph. The aim of this research is to be able to reconstruct the underlying graph given an output sequence of colors. Aslam and Rivest [17] developed polynomial time algorithms to construct minimum consistent underlying graphs of degree 2. Algorithms to construct underlying graphs of degree > 2 are being pursued.

Aslam has also been working with Aditi Dhagat on the problem of 2- coloring k-hypergraphs. Aslam and Dhagat [16] developed an optimal, online algorithm for 2-coloring a k-hypergraph when the hypergraph has fewer than 2^{k-1} edges. Further, they have shown that if the hypergraph has more than $(3 + 2\sqrt{2})^k$ edges, then no online coloring algorithm exists.

Mihir Bellare

Bellare, Micali, and Ostrovsky investigated zero-knowledge interactive proofs in several directions.

Quadratic residuosity and graph isomorphism are classic problems and the canonical examples of zero-knowledge languages. However, despite much research effort, all previous zero-knowledge proofs for them required either unproven complexity assumptions or an unbounded number of rounds of message exchange. For both languages (and more generally for any random self-reducible language), [45] we exhibit zero-knowledge proofs that require only five rounds and no unproven assumptions.

Statistical zero-knowledge is a very strong privacy constraint which is not dependent on computational limitations. In [46], it is shown that given a complexity assumption a much weaker condition suffices to attain statistical zero-knowledge. As a result it is possible to simplify statistical zero-knowledge and to better characterize, on many counts, the class of languages that possess statistical zero-knowledge proofs.

Security in cryptography is traditionally proven via *reductions.* This can lead however to considerable loss of provable security in practice. [44] analyses various *coin flipping in the well* protocols from this point of view and finally provides a new protocol which performs better than previous ones.

Bellare, Cowen, and Goldwasser investigated in [41] the properties of the *Secret Key Exchange Protocol.* This work is described in Cowen's report.

The power of interactive proofs arises from the use of two resources: interaction and randomness. While the first has been much investigated, the second is considered for the first time in [42]. Here, it is shown how to substantially reduce the amount of randomness necessary for the verifier to be convinced of membership in a language, within a given error probability and a given number of rounds of interaction.

Bonnie Berger

This year, Berger continued her research on removing randomness from algorithms, especially focusing on parallel algorithms. All related work is contained in her completed doctoral dissertation [37]. There are three steps to removing randomness from parallel algorithms: first, design a randomized parallel algorithm; second, modify this algorithm to work using a weaker form of randomness (one which has a smaller probability space); third, deterministically search this probability space for a point on which the algorithm performs well.

Berger, together with Rompel, [49][36] developed a general framework for removing randomness from randomized *NC* algorithms whose analysis uses only polylogarithmic independence. Previously no techniques were known to determinize those *RNC* algorithms depending on more than constant independence. One application of their techniques is the first

NC algorithm for the set discrepancy problem, which can be used to obtain many other *NC* algorithms, including a better *NC* edge coloring algorithm and the first *NC* lattice approximation algorithm. As another application of their techniques, they provided the first *NC* algorithm for a hypergraph coloring problem. Additionally, they showed the set discrepancy problem actually requires $(\log n)$-wise independence. This work received the Machtey Award for Best Student Paper at FOCS'89.

Berger, working with Rompel and Peter Shor [50], gave the first *NC* approximation algorithms for the unweighted and weighted set cover problems. Their algorithms use a linear number of processors and give a cover that has at most $\log n$ times the optimal size/weight, thus matching the performance of the best sequential algorithms. Previously, there were no known parallel algorithms for the general set cover problem. Berger, Rompel, and Shor devised a randomized algorithm, depending on only pairwise independence, and then converted it to a deterministic one. Furthermore, they applied their set cover algorithm to learning theory, giving an *NC* algorithm to learn the concept class obtained by taking the closure under finite union or finite intersection of any concept class of finite VC-dimension which has an *NC* hypothesis finder. In addition, they gave a linear-processor *NC* algorithm for a variant of the set cover problem, and used it to obtain *NC* algorithms for several problems in computational geometry.

Berger, working with Peter Shor [51], devised the first nontrivial randomized polynomial-time and *RNC* algorithms for the maximum acyclic subgraph problem (or equivalently, the minimum feedback arc set problem), and used known, highly sequential techniques to convert the randomized polynomial-time algorithm to a deterministic one.

Berger [47] introduced a new combinatorial method to bound the expectation of an absolute value from below by a fourth moment. She presented a special case of this lower bound in a particularly useful form—yielding a general mathematical inequality on expectations. She applied her fourth moment method to obtain the first nontrivial, and also fairly efficient, *NC* algorithm for the maximum acyclic subgraph problem. The method she used to derive this algorithm allowed her to obtain a new bound on tournament ranking. She also applied her fourth moment method to devise the first *NC* algorithm (which is also fairly efficient) for the problem of obtaining large discrepancy. This can be used to obtain the first *NC* algorithms for the Gale-Berlekamp switching game and the edge discrepancy problem. In fact, she showed that it is truly necessary to consider a fourth moment—that is, the requirement of 4-wise independence is tight.

Diverging from the topic of removing randomness, Berger with Cowen [48], investigated a new, natural enlarged class of scheduling problems, allowing concurrency and weak precedence constraints, as well as the classical partial order constraints among tasks. They proved that the problem of scheduling with both prerequisites and co-requisites, and the problem of scheduling with just $\leq$ constraints, are both NP-Complete for $k \geq 3$ parallel processors. They gave an $O(n^3)$ algorithm for optimally scheduling with any subset of $\{<, \leq, =\}$ constraints for $k = 2$, and obtained approximation algorithms for $k \geq 3$ processors. Their results have applications to the Horizon architecture, an architecture currently receiving a good deal of attention in Supercomputing research.

Berger is currently beginning an NSF Mathematical Sciences Postdoctoral Research Fellowship, at MIT under the sponsorship of Daniel Kleitman.

Margrit Betke

Betke began her studies as a graduate student at MIT in September 1989. She spent most of the year on coursework, and worked on problems in computational geometry. Betke is interested in algorithms for machine learning. She plans to work with Ron Rivest during the summer of 1989.

Avrim Blum

Blum has been working on problems in computational learning theory and continuing his work in approximation algorithms for graph coloring.

In the area of machine learning theory, he developed a model for learning from examples in situations where there is a very large number of potential features or attributes that examples might have, even though each example itself may have only a few of them [60]. For instance, this might occur if one wished to classify research papers based on keywords; each paper has only a few keywords, but the space of potential keywords is quite large and perhaps not even known beforehand. He shows that many of the basic kinds of learnable concepts in the standard theoretical models can be learned through different methods in this new model as well. Blum has also worked on examining the relationship between two existing popular theoretical learning models [61]. He shows a method using techniques from cryptography to create concepts easy to learn in one model but hard to learn in the other. This method allows one to focus on important differences between the models.

Together with Singh, Blum has been studying the problem of learning concepts that can be described as a function of a small number of monomials over a set of Boolean variables. These concepts generalize the class of k-term DNF studied by Pitt and Valiant [247]. Blum and Singh show that the generalized class can be learned, but that to avoid intrinsic computational limitations, the learner must use an expanded hypothesis representation. That is, if the learner is forced to represent his hypotheses in the same form as the concept being learned, then the problem becomes hard (NP-complete), but if the learner is allowed a freer selection of hypotheses, then learning can be done quickly.

In the area of approximation algorithms for graph coloring, Blum improved on previous performance guarantees for the problem of approximate coloring of 3-chromatic graphs. He presents an algorithm that will color any 3-chromatic graph with $O(n^{3/8+o(1)})$ colors in the worst case [62]. In addition, he is examining the problem of coloring graphs generated by random and "semi-random" sources.

Tom Cormen

Cormen, along with Charles Leiserson and Ron Rivest, completed writing the textbook *Introduction to Algorithms* [80], which is being copublished by The MIT Press and McGraw-Hill. The book should appear in May or June 1990.

Cormen is now pursuing doctoral research. He is currently working with Leiserson and Shlomo Kipnis on the notion of locality-enhancing graph embeddings: given a source graph

G and a target graph H, embed G in H so that if two vertices u and v in G are distance l apart (i.e., the shortest path between u and v contains l edges), then the distance between u and v in H is $f(l)$, where f is as small a function as possible. Cormen and Kipnis have proven upper bounds on embedding linear graphs in d-dimensional meshes, X-trees, and hypercubes, and they are continuing this line of research.

Lenore Cowen

In [41], Bellare, Cowen, and Goldwasser investigated the properties which must be inherent in any *Secret Key Exchange Protocol*, which is a protocol enabling two parties to establish a common and secret key over public channels. They showed strong structural requirements for such a protocol in both the uniform and non-uniform model. These in turn allowed them to prove that one-way functions are *necessary* for secret key exchange (as a corollary, one way functions are necessary for oblivious transfer). Their results imply that the existence of a secret key exchange protocol also implies existence of strong bit commitment schemes, (i.e., schemes in which the committer can reveal only one possible value, regardless of the computational power of the committer.)

Cowen, with Berger [48], investigated a new, natural enlarged class of scheduling problems, allowing concurrency and weak precedence constraints, as well as the classical partial order constraints among tasks. They proved that the problem of scheduling with both prerequisites and co-requisites, and the problem of scheduling with just $\leq$ constraints, are both NP-complete for $k \geq 3$ parallel processors. They gave an $O(n^3)$ algorithm for optimally scheduling with any subset of $\{<, \leq, =\}$ constraints for $k = 2$, and obtained approximation algorithms for $k \geq 3$ processors. Their results have applications to the Horizon architecture, an architecture currently receiving a good deal of attention in supercomputing research [100][179][280].

Cowen plans to continue work on issues raised in her work with Berger; she also plans to investigate further cryptographic problems with Goldwasser.

Aditi Dhagat

Dhagat was a teaching assistant for the graduate Theory of Computation course (6.840/18.404) during the fall semester. She also worked with Sipser on the problem of constructing pseudorandom number generators which are secure without requiring unproven assumptions. This is in the spirit of results of Nisan and Wigderson [232], where pseudorandom generators secure against constant depth polynomial were shown to exist. The security relied on proven lower bounds on circuit size for computing the parity function. A later result of Nisan [231] constructs pseudorandom generators secure against one-way logspace machines, where the security comes from using universal hash functions. This generator stretches a random seed of size $\log^2 n$ to a pseudorandom string of length n. Dhagat and Sipser are also working on the problem of improving this stretch, while maintaining the strength of the security of the generator.

During the spring semester, Dhagat worked with Aslam on a problem suggested by Joel Spencer. She and Aslam [16] have shown online algorithms for 2-coloring k-hypergraphs. Their results show that if the k-hypergraph has no more than 2^{k-1} edges, then it can be

2-colored by an optimal online algorithm. If the k-hypergraph has bounded degree (say k) and has more than $(3 + 2\sqrt{2})^k$ edges, then they show that there is no online algorithm to 2-color, regardless of computation time.

Michael Ernst

Ernst spent most of his time on coursework this year, but also continued work with Meyer. He will intensify his research on programming language semantics during the summer, and expects to complete his S.M. thesis within the year.

Wayne Goddard

Goddard is a second-year student focusing on combinatorics. He has also been working on "partial" sorting problems. In these problems, one is asked to determine only some of the relationships between the elements; for example one might be given a collection of subsets of the elements, and asked to find the maximum in each set. Together with King and Schulman [119], Goddard found randomized algorithms for two such problems which use significantly less comparisons than naive algorithms. It is also shown in [119] that these algorithms are optimal to within a constant factor.

Sally A. Goldman

Goldman continued to work in the area of computational learning theory. With Rivest and Schapire, Goldman studied the problem of designing polynomial prediction algorithms for learning binary relations [126]. They study these problems under an online model in which the instances are drawn by the learner, by a helpful teacher, by an adversary, or according to a probability distribution on the instance space. Their goal is to minimize the number of incorrect predictions. They represent the relation as an $n \times m$ binary matrix, and present results for when the matrix is restricted to have at most k distinct row types, and when it is constrained by requiring that the predicate form a total order. Namely, for both cases they present upper and lower bounds on the number of mistakes any learning algorithm makes when learning such a matrix under their extensions of the online learning model.

With Kearns, Goldman is exploring some of the interesting questions raised by the extended mistake bound model discussed above. Namely, they are studying how the complexity of the learner's task (as judged by the number of mistakes) depends on the presentation order of the instances. When a teacher selects the presentation order, they consider the maximum number of mistakes made by any learner that predicts according to some concept that agrees with all previously seen instances. Thus they ask: what is the minimum number of examples a teacher must reveal to uniquely identify the target concept? They are also studying the related question of how many mistakes the learner must make in the worst case when selecting the presentation order for the instances himself.

With Kearns and Schapire, Goldman has been investigating aspects of the PAC model [287]. They developed a new technique for exactly identifying certain classes of Boolean formulas from random examples [124]. (Furthermore, they prove that these classes of circuits are not efficiently learnable in the PAC model.) Their method is based on the observation of the input-output behavior of the target formula on a fixed probability distribution which is determined by the *fixed point* of the circuit's *amplification function* (defined as the probability that a 1 is output when each input is 1 with probability p). By demonstrating that the circuit's behavior is *unstable* in an appropriate sense on this distribution, they are able to infer all structural information about the circuit (with high probability) by performing various statistical tests on easily sampled variants of the fixed distribution.

With Kearns and Schapire, Goldman has also been studying the complexity of weakly learning [125]. Namely, how much data must be collected from an unknown distribution in order to extract a small but significant advantage in prediction. Such a study is motivated in part by settings in which there is a limited supply of examples.

Goldman will be finishing her Ph.D. [123] in June, and has accepted a faculty position at Washington University in St. Louis beginning in August.

Michelangelo Grigni

Grigni is a fourth year graduate student supervised by Sipser. His primary research involves the construction of fast robust networks for various kinds of communication problems, continuing work begun on the broadcasting problem with David Peleg [133] of the Weizmann Institute.

With Sipser, Grigni is developing a classification of monotone space complexity classes, including monotone logspace and monotone bounded- width branching programs. Independently (and recently with Guibas), Grigni is attempting to tighten the large complexity gaps of various sum-sorting problems.

Mark D. Hansen

Hansen completed his work in graph embeddings [139] and spent most of this year working with Leighton and Aggarwal on computational geometry problems relating Voronoi diagrams and query-retrieval problems [4]. The research which he has done in both of these areas is presently being incorporated into his Ph.D. thesis. Hansen anticipates graduating in December 1990.

In his most recent work with Aggarwal and Leighton, Hansen describes a new technique for solving a variety of query-retrieval problems in optimal time with optimal or near-optimal space. In particular, he uses the technique to construct algorithms and data structures for circular range searching, half-space range searching, and computing k-nearest neighbors in a variety of metrics. For each problem and each query, the response to the query is provided in $O(k)$ or $O(k + \log n)$ time where k is the size of the response and n is the size of the problem. (E.g., for the n-point k-nearest neighbors problem, the k-nearest neighbors of any query point are provided in $O(k + \log n)$ steps.) Depending on the problem being solved, the space required for the data structure is either linear or $O(n \log n)$. Hence, the time bounds are optimal and the space bounds are optimal or near-optimal. Previously known data structures for these problems required a factor of $\Omega(\log n(\log\log n)^2)$ or $\Omega(\log n \log\log n)$ more space and/or more time to answer each query.

The compaction technique introduced incorporates planar separators, filtering search, and the probabilistic method for discrepancy problems. The fundamental idea is that k^{th}-order

Voronoi diagrams (and other suitable proximity diagrams) can be compacted from $k^{O(1)}n$ space to $O(n)$ space and still retain all the information that is essential for solving query problems.

Alexander T. Ishii

Ishii generalized his VLSI timing analysis algorithms using the notion of a "base step" function to encapsulate assumptions about when signal values change during the operation of a circuit. He has shown how various base step functions can be used to provide sufficient conditions for a circuit to operate properly. The base step function is used to derive a "computational expansion" of the circuit from which a collection of simple linear constraints are derived. These constraints can be efficiently checked using standard graph algorithms. In addition, the algorithm can be adapted to determine the maximum frequency at which a circuit can be clocked and to produce the limiting critical path.

Ishii and Leiserson developed a new base step function which is less pessimistic than the analogous ones used in previous timing verifiers, yet correctly handles timing constraints that are "cyclic" or extend across the boundaries of multiple clock phases or cycles. If a circuit is modeled as a graph $G = (V, E)$, where V consists of components—latches and functional elements—and E represents intercomponent connections, the new base step function results in an algorithm which verifies the proper timing of a circuit in worst-case $O(|V||E|)$ time and $O(|V|^2)$ space [149].

Ishii has also been working with Thomas Knight of the MIT Artificial Intelligence Laboratory on a self-terminating, digitally-controlled, and ECL-compatible output pad driver for high speed integrated circuits. By automatically series-terminating driven lines with their characteristic impedances, the driver realizes speed, power, and noise improvements over conventional designs. Series termination is realized by exploiting the intrinsic series resistance of the output drive transistors. The design has not yet been fabricated, but simulations indicate that data-transition rates in excess of 100MHz are possible.

Lalita A. Jategaonkar

Jategaonkar finished her Master's thesis [151], which further develops previous work with John C. Mitchell, a professor at Stanford University. The thesis was supervised by Albert Meyer. This research develops an extension of the programming language ML in which a restricted object-oriented style can be achieved. In keeping with the framework of ML, a type derivation system and a type inference algorithm are presented. It is proved that the algorithm is sound and complete with respect to the type derivation system, and that it infers a most general typing of every typeable expression in the language. A technical report based on the thesis was also published this year [152], and Jategaonkar is currently working on a journal version of this work.

Trevor Jim

Working with Meyer, Jim continued work on models of the language PCF [249], a simply-typed lambda calculus.

He has shown that the models of Berry and Curien [83][52], based on stable functions and sequential algorithms, are not restrictions of the usual Scott model. These models were developed as alternatives to the standard model, which contains troublesome "non-sequential" elements.

Extending the work of Bloom [54], Jim has shown that there is no extension of PCF by SOS rules for which Berry's stable function model is fully abstract. Further directions include extending this result to the other models, and the design of a restricted model based on Berry's bidomains.

This research will comprise Jim's Master's Thesis, which he plans to write over the summer.

Nabil Kahale

Kahale entered the department in September 1989. He achieved an earlier work on modular properties of the Bell Numbers. His results will appear in the *Journal of Combinatorial Theory Series A* [155].

Kahale is now working with Tom Leighton on the analysis of various routing protocols on a mesh of arrays by applying new mathematical techniques. He also plans to work on the construction of graphs having good expansion properties. Expanders are largely used in the design of algorithms and networks in the field of parallel computation.

Michael Kearns

Kearns continued his research in the area of machine learning. Together with Lenny Pitt of the University of Illinois, Kearns gave an algorithm for learning *pattern languages* with a bounded number of variables from random examples drawn according to an arbitrary product distribution over the substitution variables [157]. Pattern languages are motivated by problems in string matching and genetics. Recent results by other researchers indicate that the pattern learning problem with an unbounded number of variables is intractable, so the algorithm of Kearns and Pitt is optimal in an informal sense.

Together with Sally Goldman and Rob Schapire, Kearns developed an algorithm for exactly identifying certain classes of circuits based on *amplification functions* [124]. The central idea is to observe the input-output behavior of the unknown circuit on a simple probability distribution that is unstable, in the sense that small perturbations of this distribution cause noticeable statistical changes in the circuit's output. These techniques can be applied to show the existence of *universal identification sequences* for classes of functions, which are fixed input sequences whose outputs suffice to exactly identify any circuit from the class.

Again with Goldman and Schapire, Kearns investigated the sample complexity of weak learning, in which the learning algorithm needs only to find an hypothesis whose accuracy is slightly better than random guessing [125]. This model is motivated by situations in which examples are rare and also by cryptographic settings, where small biases can greatly compromise security. Results are given that demonstrate the power of randomized hypotheses in a weak learning setting, and a partial combinatorial characterization of weak learning sample complexity is given.

Kearns and Schapire introduced a new model of learning *probabilistic concepts*, in which each instance may have some probability of being positive and some probability of being negative [158]. Such a model addresses settings as diverse as weather prediction, where one is typically given a probability of rain for the day, and simple object classification, where a probability may best model the degree to which an object has the desired property. In this model, Kearns and Schapire give many provably efficient learning algorithms and investigate the underlying theory of learning probabilistic concepts. This includes general techniques for constructing good learning algorithms and a characterization of the sample size based on uniform convergence results that generalizes the Vapnik-Chervonenkis dimension.

Finally, Kearns has spent part of the year making revisions to his doctoral dissertation, *The Computational Complexity of Machine Learning*, which will be published by The MIT Press in September.

Beginning in September 1990, Kearns will be at the International Computer Science Institute in Berkeley, California.

Joe Kilian

Kilian has been working on the complexity of various cryptographic protocols. For instance, given two infinitely powerful parties who have access to an ideal commitment scheme (envelopes), can one party efficiently commit to a set of bits, and give a zero-knowledge proof for an arbitrary boolean predicate on these committed bits? Here, "efficient" means using a polynomial number of envelopes. Surprisingly, the answer is yes [38]. This result is used to prove new upper bounds on the communication complexity of certain secure distributed computations [38].

Joe also worked on so-called robust transformations of interactive proof systems [159]. In a robust transformation, the new prover is not allowed to be substantially more powerful than the original prover. He considered the problem of how to robustly transform interactive proof systems into zero-knowledge interactive proof systems.

Joe worked on developing interactive proof systems with some provable security properties [160]. Finally, he found even more ways of implementing oblivious transfer in terms of seemingly weaker cryptographic protocols [161].

Shlomo Kipnis

Kipnis continued his investigation of *bussed interconnections*. He is trying to further explore the power of bussed interconnection schemes for routing permutations and realizing various communication patterns. Bussed interconnection schemes and their relation to *difference covers* were explored by Joe Kilian, Shlomo Kipnis, and Charles Leiserson in [162].

Kipnis also investigated priority arbitration schemes that employ m busses to arbitrate among n modules. His novel *binomial arbitration scheme*, which uses at most $m = \lg n + 1$ busses, enables achieving an arbitration time of $t = \frac{1}{2} \lg n$ (in units of bus-settling delay). This scheme substantially outperforms the commonly used *binary arbitration scheme*, which uses $m = \lg n$ busses and achieves an arbitration time of $t = \lg n$. Furthermore, his *generalized binomial arbitration scheme* achieves a bus-time tradeoff of the form $m = \Theta(tn^{1/t})$ between the number of arbitration busses m and the arbitration time t, for values of $1 \leq t \leq \lg n$

and $\lg n \leq m \leq n$. These new schemes can be adopted with no changes to existing hardware and protocols; they merely involve selecting a good set of priority arbitration codewords [167]. Kipnis also applied for a patent on these new arbitration schemes, through the Technology Licensing Office.

In addition, he compiled a survey paper on the problem of range queries in computational geometry. Range queries are a fundamental problem in computational geometry with applications to computer graphics and database retrieval systems. The survey paper identifies three general methods for range queries in computational geometry and classifies many of the recent research results into one or more of these methods [168].

Michael Klugerman

Klugerman took courses in algorithms, complexity theory, and probability. He TA'd 18.409 Probabilistic Methods in the spring, and worked on routing problems on the hypercube. Also, he plans to write a paper with Daniel Kleitman and others on a geometry problem given to us by Paul Erdos.

Dina Kravets

Kravets spent the fall semester TAing 18.435, and continuing work with Alok Aggarwal and James Park on algorithms for totally monotone and Monge arrays. Their algorithms for parallel searching in generalized Monge arrays (together with Sandeep Sen) will appear in [6]. During the spring semester, she started working with Leo Guibas on some problems in computational geometry.

Arthur Lent

Lent became a graduate student at MIT in February 1990. Prior to then, he was an undergraduate member of TOC. Working under Meyer's supervision, he completed his Bachelor's thesis [190] in January 1990. The thesis presents a call-by-name SECD machine. Earlier work on SECD machines [248][185] had focused on call-by-value SECD machines. The basic correctness result for the machine demonstrates that the operational behavior of this machine correspondes with the operational behavior of PCF [249].

Lent also developed a set of notes for a three week unit on "Programming Language Theory" for use in the undergraduate class "Computability, Programming, and Logic." These notes gave a development of the semantics of a call-by-value variant of PCF, that was an attempt to capture the "Functional Kernel of Scheme (FKS)." The notes provided two operational characterizations, one via rewrite rules, and one via a SECD machine, and established their operational equivalence. The notes culminated in a development of a denotational semantics for FKS. The denotational semantics were based on Plotkin's partial function model. Lent and Riecke jointly worked to establish a direct proof of adequacy for this model.

Lent also became interested in intuitionistic logic, and its applications to computer science. His reading in intuitionistic logic has focused mainly on current efforts to explore notions of "Feasibly Constructive Arithmetic," which is an attempt to generalize the notion of polynomial time to higher type. One goal of this exploration would be to extract polynomial time algorithms from feasibly constructive proofs.

Leonid Levin

Comparing VLSI models [150] shows when the effect of wires' speed overtakes any effect of wires' width and gives the best possible (slightly superlinear) threshold at which all effects of wires' width vanish.

[150] shows the unexpected equivalence of average case NP-completeness in two classes of distributions: P-time computable and P-time samplable.

Random Linear predicates $R(x)$ of inputs x of one-way functions f were previously shown (in [Goldreich Levin 89]) to be unpredictable from $f(x), R$. Their security was equal to the security of f within a constant power. [194] makes this result tight: constant power is just 1 and the security loss is only a factor of $|x|^{O(1)}$.

Bruce Maggs

Maggs is studying algorithms for routing packets in switching networks. With Tom Leighton, he developed a multibutterfly routing algorithm that is both robust against faults and efficient from a practical point of view. For example, on an N-input multibutterfly with k faulty switches, the algorithm can route any permutation between some set of $N - O(k)$ inputs and $N - O(k)$ outputs in $O(\log N)$ time. Bruce Maggs, Sanjeev Arora, and Tom Leighton also designed the first efficient on-line algorithm for path selection in an optimal-size nonblocking network. Viewed in a telephone switching context, the algorithm can put through any sequence of calls among N parties on a network of size $O(N \log N)$ with $O(\log N)$-bit-step delay per call, even if many calls are made simultaneously. Finally, Bill Aiello, Tom Leighton, Bruce Maggs, and Mark Newman discovered a randomized $O(\log N)$-bit-step algorithm for bit-serial message routing on a hypercube. The result is asymptotically optimal, and improves upon the best previously known algorithms by a logarithmic factor. The algorithm is adaptive, and by generalizing the Borodin-Hopcroft lower bound on oblivious routing, they show that any other $O(\log N)$-bit-step algorithm must be adaptive too.

Yishay Mansour

Mansour's interests include distributed communication networks, machine learning, and complexity.

In the area of machine learning, he continues to investigate the possibilities of achieving learnability by considering the Fourier transform. In [215], he describes various learning algorithms in this spirit.

In the area of distributed computing, he is mainly interested in routing algorithms. In the work with Cidon, Kutten, and Peleg [74], they investigate simple routing strategies. In the work with Schulman [218], they prove a tradeoff between time and space for sorting on a ring.

In a work with Nisan and Tiwari [216], they investigate the computational complexity of universal hash functions. They are able to establish a time space tradeoff for any implementation of such functions.

Mojdeh Mohtashemi

Mojdeh entered the department in September 1989. During fall 1989, she was a teaching assistant for the undergraduate course in Structure and Interpretation of Computer Programs taught by Gerald Sussman. Mojdeh spent most of the year on coursework and readings in cryptography.

Carolyn H. Norton

Norton, working with Éva Tardos (Cornell) and Serge Plotkin (Stanford) [234][233], developed a strongly polynomial algorithm to solve finite dimensional linear programs, when feasible space is not given explicitly by a set of inequalities, but is instead given by a separation algorithm.

She is currently working with Dimitris Bertsimas on studying the relation between integer programs and their linear programming relaxations.

Rafail Ostrovsky

Ostrovsky is working on feasibility results concerning the implementation of secure computation and proof systems in insecure communication environments. More specifically, he explores two basic questions: the first is basing security primitives on assumptions as general as possible and making connections between these primitives, and the second is investigating various models of computation in which secure computation can be implemented.

Mihir Bellare, Silvio Micali, and Rafail Ostrovsky show how the number of rounds of interaction can be reduced (to constant) for Perfect zero-knowledge interactive proofs for Graph Isomorphism and Quadratic Residuosity (more generally, for any random self-reducible problem) [45]. In [46], they show how a statistical zero-knowledge protocol for an *honest* verifier can be compiled into a statistical zero-knowledge protocol which works for any (even cheating) verifier. They examine the power of the Prover for giving a Zero-Knowledge proof; they compare the "black-box" definition zero-knowledge to the standard definition; and they examine if zero-knowledge property can be retained if Prover wants to convince verifier with probability one [46].

Joe Kilian, Silvio Micali, and Rafail Ostrovsky show how zero-knowledge protocols for NP can be made non-interactive, assuming a pre-processing Oblivious Transfer protocol [164].

In [239], Ostrovsky presents the first poly-logarithmic simulation of an arbitrary computation on probabilistic Oblivious RAM.

Ostrovsky, Venkatesan, and Yung [240] examine the model of two-party partial-information games, when one of the players is infinitely-powerful, while the other is polynomially bounded. They show that any such game is playable, given *any* one-way function and that there exists a bit-commitment protocol from an infinitely-powerful player to a polynomial player, unless there is no hard on average problem in PSPACE.

Marios C. Papaefthymiou

Papaefthymiou has been investigating parallel architectures under the supervision of C. E. Leiserson.

His research explores both theoretical and practical aspects of synchronous circuitry optimization. A general framework for this problem has been given by C. E. Leiserson and J. Saxe [188]. Papaefthymiou demonstrated the closed-semiring structure of the retiming operation on unit-delay circuits. He also gave a concise mathematical characterization of the minimum clock period of a clocked circuit in terms of the minimum register-to-delay ratio cycle in the system's graph representation. This result led to improved algorithms for retiming of circuits with maximum delay D: an $O(V^{1/2}E \lg VW \lg VD)$ algorithm for retiming with clock period that does not exceed the minimum by more than D, and an $O(VE \lg D)$ algorithm for minimum clock period retiming.

Recently, Marios Papaefthymiou with C.E. Leiserson have been investigating efficient algorithms for mixed integer optimization problems.

James K. Park

Park spent the last year working with Alok Aggarwal (IBM Yorktown Heights), Dina Kravets, and Sandeep Sen (Duke University) on a number of problems relating to Monge arrays. An $m \times n$ array $A = \{a[i,j]\}$ is called *Monge* if for $1 \leq i < m$ and $1 \leq j < n$, $a[i,j] + a[i+1,j+1] \leq a[i,j+1] + a[i+1,j]$. Monge arrays were introduced in 1987 by Aggarwal, Klawe, Moran, Shor, and Wilber [5], who showed that several problems in computational geometry and VLSI theory could be reduced to the problem of finding the maximum entry in each row of a Monge array. Since this seminal paper, Monge arrays have been studied by a number of researchers, and many additional applications of the Monge array abstraction have been developed.

Park's most recent work with Monge arrays falls into three categories. First, Aggarwal and Park have been studying the use of Monge arrays in solving economic lot-size problems, an important class of problems from operations research [8][7]. Second, Aggarwal, Kravets, Park, and Sen have been developing parallel algorithms—specifically PRAM and hypercube algorithms—for searching in Monge arrays and generalized Monge arrays [6]. Third, Kravets and Park have been investigating several selection and sorting problems in the context of Monge arrays [178].

In the coming year, Park plans to complete his doctoral thesis, a study of Monge arrays and their applications. He also plans to begin work with Aggarwal and Marie Klawe (University of British Columbia) on a monograph on the subject of Monge arrays.

Greg Plaxton

Plaxton is working on the development of algorithms for sorting on "cube-type" networks. The class of cube-type networks includes the hypercube, butterfly, cube-connected cycles and shuffle-exchange. Note that a special case of sorting is *routing*, which is arguably the most important primitive operation required by any large scale parallel computer. With Robert Cypher (IBM Almaden), Plaxton has devised a deterministic algorithm for sorting on cube-type networks that runs in $O(\log n (\log\log n)^2)$ time. The best previous deterministic algorithm for the sorting problem is due to Batcher, and runs in $O(\log^2 n)$ time [35]. If a certain amount of preprocessing is allowed, the running time of Cypher and Plaxton's algorithm can be reduced to $O(\log n \log\log n)$. The best known lower bound for sorting on cube-type networks remains $\Omega(\log n)$ (a trivial bound).

More recently, Greg Plaxton and Tom Leighton have been investigating *randomized* algorithms for sorting on cube-type networks. The results follow from considering a particular tournament based on the butterfly, and demonstrating that the tournament possesses a strong ranking property. The ranking property of this tournament is exploited by using it as a building block for efficient parallel sorting algorithms under a variety of different models of computation. Three important applications are provided. First, a sorting circuit of depth $7.5 \log n$ is defined that sorts all but a superpolynomially small fraction of the $n!$ possible input permutations. Second, a randomized sorting algorithm is given for cube-type networks that runs in $O(\log n)$ word steps with very high probability. This algorithm has a significantly smaller probability of failure than any previous randomized algorithm for sorting on cube-type networks in the word model. Third, a randomized algorithm is given for sorting n $O(m)$-bit records on an $n \log n$ node butterfly that runs in $O(m + \log n)$ bit steps with very high probability. All previously known algorithms for sorting on cube-type networks require $\Omega(\log^2 n)$ bit steps.

Jon G. Riecke

Riecke has pursued the theory of functional languages. The research focused on moving the existing theory towards more realistic programming languages.

In joint work with Stavros Cosmadakis and Albert Meyer, Riecke investigated the theory of "lazy" functional languages [82]. The term "lazy" applies to functional languages which pass arguments call-by-name but which halt at functional abstractions. Almost all call-by-name languages exhibit this lazy behavior, but the standard theory of functional languages does not adequately explain it. Building on the denotational semantics work of Abramsky [1], Ong [237][238], Bloom and Riecke [59], and Cosmadakis [81], the three have found a complete and decidable logic for proving computationally valid equations ("observational congruences") between a certain class of programs. This logic forms the basis of reasoning about code in most lazy functional languages. Complexity-theoretic issues in the logic will be addressed in a forthcoming paper by Cosmadakis and Riecke.

Using the work in lazy languages as a basis, Riecke also examined the logic of call-by-value languages [254]. Call-by-value languages form the vast majority of functional languages e.g., Lisp, SCHEME, and ML), so this case, like the lazy case, is important from a practical point of view. The discovery of a complete and decidable proof system for a limited class of call-by-value equations is the main contribution of this work.

Riecke has also been working on connections between the operational and denotational semantics of different programming languages. Specifically, he looked at the problem of finding *translations* from one programming language to another. A general recursion-theoretic argument shows that almost any programming language can simulate another. The problem lies in finding tasteful translations that do not rely on the computational power of the programming language. Translations from call-by-value to lazy languages, for example, have been found, as well as negative theorems demonstrating the impossibility of finding well-structured translations from lazy to call-by-value. The work will appear in a forthcoming paper.

Phillip Rogaway

The problem of secure distributed function evaluation entails a network of processors endeavoring to compute some function on privately held inputs, but in a manner which preserves the privacy of these inputs. Working with Silvio Micali, Rogaway showed how to accomplish this goal using only a constant number of rounds of interaction [40]. This vastly improves the efficiency of [127] and related protocols for secure computation.

Rogaway investigated the possibility of constant round, information theoretically secure computation. In [38], it is shown that the class of functions computable in this way is quite rich.

Foundational issues in secure computation have, so far, lagged behind the field's accomplishments. In [166], careful definitions for correct and private computation under a dynamic adversary are developed.

This summer, Rogaway will be working on writing up his thesis.

John Rompel

This year Rompel continued his research on low-independence randomness. This research was along two main lines: the use of low-independence distributions to remove randomness from parallel algorithms and the use of low independence hash functions in cryptographic applications.

Rompel, together with Berger [49], developed a general framework for removing randomness from randomized NC algorithms whose analysis uses only polylogarithmic independence. Previously, no techniques were known to determinize those RNC algorithms depending on more than constant independence. One application of their techniques is an NC algorithm for the set discrepancy problem, which can be used to obtain many other NC algorithms, including a better NC edge coloring algorithm. As another application of their techniques, they provided an NC algorithm for a hypergraph coloring problem.

Rompel, working with Berger and Peter Shor [50], gave NC approximation algorithms for the unweighted and weighted set cover problems. Their algorithms use a linear number of processors and give a cover that has at most $\log n$ times the optimal size/weight, thus matching the performance of the best sequential algorithms. Previously, there were no known parallel algorithms for the general set cover problem. Berger, Rompel, and Shor devised a randomized algorithm, depending on only pairwise independence, and then converted it to a deterministic one. Furthermore, they applied their set cover algorithm to learning theory, giving an NC algorithm to learn the concept class obtained by taking the closure under finite union or finite intersection of any concept class of finite VC-dimension which has an NC hypothesis finder. In addition, they gave a linear-processor NC algorithm for a variant of the set cover problem first proposed by Chazelle and Friedman, and used it to obtain NC algorithms for several problems in computational geometry.

The other line of Rompel's research concerns the cryptographic applications of low independence hash functions. The first such application he considered was that of digital signatures. He showed in [255] that secure signature schemes could be constructed from any one-way function. This improved the best previous result, due to Naor and Yung, that one-way permutations suffice to construct signatures. Furthermore, his result is optimal, since it was known that one-way functions were necessary for signatures. The construction starts with an arbitrary one-way function and, after a series of six steps, ends with a one-way hash function (which Merkle showed can be used to construct signatures). Most of the steps are based on the use of low independence hash functions to alter the probability distribution induced by a function.

Another application of low independence hash functions, explored in joint work with Bellare and Goldwasser, is amplifying the error probability of interactive proofs without increasing the randomness required [42]. For this construction, low independence hash functions are used to randomly sample a function. By taking successively smaller samples using successively larger independence, they are able to exploit a tradeoff, keeping the error probability and number of random bits constant.

Rompel is currently writing his Ph.D. thesis based on his work on low independence randomness, which he plans to finish in August. After that, he plans to do postdoctoral research, under an NSF Fellowship, at the International Computer Science Institute in Berkeley, CA.

Arie Rudich

Rudich has been working with Meyer on semantics for dataflow networks, Specifically, he is studying various notions of observing "completion." Rudich finished his Master's thesis this year. The thesis investigates a notion of "completion," based on observing fair computations. It presents a semantics for nondeterministic dataflow networks which is fully abstract with respect to observing finite input-output relations of fair computations. This semantics thus reflects both safety and liveness properties of network computations with respect to finite observations. Prior models [172][242][210][153][274] required observing infinite behaviors to accommodate liveness properties.

Rudich also studied algorithms for inference of finite automata, trying to extend the ideas to infer nondeterministic and infinite automata.

Robert E. Schapire

Schapire continued to work on problems relevant to the distribution-free ("PAC") learning model introduced by Valiant [287]. In particular, he considered the problem of improving the accuracy of a hypothesis output by a learning algorithm in this model, and has shown that a model of learnability, called *weak learnability*, in which the learner is only required to perform slightly better than random guessing is as strong as a model in which the learner's error can be made arbitrarily small [260]. His result may have significant applications as a tool for efficiently converting a mediocre learning algorithm into one that performs extremely well. His result also has some unexpected theoretical consequences relating to the general complexity of learning in the Valiant model.

Schapire also looked at the problem of learning *pattern languages*, a simple class of languages introduced by Angluin [12]. He was able to show in a very strong sense that such languages are unlearnable in the PAC model (assuming $P/poly \neq NP/poly$), regardless of the representation used by the learning algorithm [261].

With Goldman and Kearns, Schapire has been investigating other aspects of the PAC model. They developed a new technique for learning certain functions under particular fixed but simple distributions by observing the statistical behavior of the function under simple perturbations of the fixed distribution [124]. They have also been studying the complexity of weakly learning, that is, of obtaining a small but significant advantage in prediction [125]. Such a study may be especially relevant in settings in which the supply of examples is severely limited.

With Kearns, Schapire has been exploring a new and important extension to the Valiant model, namely to the problem of learning concepts that may exhibit uncertain or probabilistic behavior on some examples [158]. Such *probabilistic concepts* arise naturally in many situations, such as weather prediction, where the measured variables and their accuracy are insufficient to determine the outcome with certainty. While building on the recent results of Haussler [142] on the sample complexity of learning in probabilistic settings, Kearns and Schapire focus primarily on the design of efficient algorithms for learning probabilistic concepts. Their work also extends many of the results in the standard PAC model to the new probabilistic model.

Finally, Schapire has been working with Goldman and Rivest on the problem of inferring a binary relation between n objects of one kind and m of another [126]. This can be viewed as the problem of inferring an $n \times m$ binary matrix. Their goal has been to minimize the number of prediction mistakes made by a learner presented with such a matrix one entry at a time. They have been able to prove numerous upper and lower mistake bounds for several variations of this problem.

Leonard Schulman

In the summer and fall of 1989, Schulman worked with W. Goddard on general sorting problems. In these problems, only partial information is required about the order relations among given elements; the goal is to obtain this information with a minimum of comparisons. Their research will be presented, along with the work of V. King, at STOC 1990 [119].

In the fall of 1989, work of Schulman and Mansour on the problem of sorting in parallel, on a ring of processors, was accepted for publication in *Journal of Algorithms* [217].

Currently, Schulman is working on problems in communication and circuit complexity. He is guided in this work by his advisor M. Sipser, and by M. Karchmer. He intends to continue this work during the summer of 1990.

In other work, Schulman, with D. Kleitman, M. Klugerman, W. Goddard, and others, has been investigating some problems in combinatorial geometry.

Eric J. Schwabe

This year, Schwabe continued his study of the butterfly network vs. the shuffle-exchange graph as interconnection networks for parallel computation. Although these well-known hypercube-derived networks share many characteristics, the question of their relative computational strength has remained open.

Earlier work of his narrowed the gap between the two networks by showing that a class of structured hypercube algorithms which could be simulated efficiently on the shuffle-exchange graph could in fact be simulated on the butterfly just as efficiently. This result was recently submitted for publication [262].

Schwabe resolved the long-standing open question of the relative computational strength of these two networks by proving that any T- step computation on an N-node butterfly can be simulated in $O(T)$ steps on an N-node shuffle-exchange graph, and vice versa [263]. This result established the computational equivalence of many common hypercube-derived networks, and in addition yielded the first constant-slowdown simulation of the shuffle-exchange graph on the hypercube.

Over the next year, he plans to work with Koch, Leighton, Maggs, Rao, and Rosenberg [171] on a joint journal submission of related results on network emulations, while continuing to study problems in parallel computation on hypercube-derived networks. He also hopes to return to earlier unfinished work concerning networks for efficient parallel memory management.

Mark Smith

Smith entered the department in September 1989, and he spent most of his time on coursework. He plans to do readings in Circuit Complexity under the supervision of Mauricio Karchmer this summer.

Clifford Stein

Stein has been working on developing sequential and parallel algorithms for combinatorial optimization problems. In August, he completed his Master's thesis. The first result in the thesis is a parallel algorithm to find a maximal set of edge-disjoint cycles in an undirected graph in $O(\log n)$ time using m processors on a CRCW PRAM. He then uses as a primitive for finding a *cycle cover* containing $O(m+n\log n)$ edges using $O(\log^2 n)$ time on m processors. The thesis also contains a result giving RNC algorithms for the assignment problem which use a number of processors independent of the size of the largest number in the problem. Stein has been working on submitting these results for publication [275][170][169].

Together with Philip Klein of Brown and Éva Tardos of Cornell, Stein developed new efficient approximation algorithms for the concurrent multicommodity flow problem. Besides being an important problem in its own right, the concurrent flow problem has many interesting applications. Leighton and Rao used concurrent flow to find an approximately "sparsest cut" in a graph, and thereby approximately solve a wide variety of graph problems, including minimum feedback arc set, minimum cut linear arrangement, and minimum area layout. Raghavan and Thompson used concurrent flow to approximately solve a channel width minimization problem in VLSI. Klein, Stein, and Tardos give a fully polynomial approximation scheme for concurrent flow. Their algorithm is simple, and as a corollary, they get $O(m^2 \log m)$ time algorithm to find an approximately sparsest cut in an m-edge graph, and an $O(k^{1.5}(m + n\log n))$ time algorithm to find an approximation to the channel width minimization problem in an n-node, m-edge, k-channel graph.

Larry J. Stockmeyer

Stockmeyer has been working with Hagit Attiya, Cynthia Dwork, and Nancy Lynch on the time to reach agreement in a timing-based model, where there is uncertainty in message delivery time and processor speeds, and where processors may fail by stopping. Upper and lower bounds, tight to within a factor of 2, have been obtained. Stockmeyer is a member of the program committee for the 1990 FOCS Conference.

David Wald

Wald came to MIT as a graduate student in September 1989. He is working with Meyer, examining calculi for describing concurrent processes.

Joel Wein

Wein continued to work on several aspects of parallel computation and combinatorial optimization. He extended his results on Las Vegas *RNC* algorithms, reported on in the last progress report, to the case of planar multicommodity flows [291]. Working with Leighton and Shmoys, he developed randomized approximation algorithms for the problem of *job shop* scheduling, that achieve schedules that are $O(\log^2 M)$ worse than optimal, where M is the number of machines. The best previously known deterministic polynomial time algorithm was an $O(M)$ approximation.

In the area of practical parallel algorithms, with Zenios of the University of Pennsylvania, he developed an improved Connection Machine algorithm for the assignment problem. This algorithm, a modification of Bertsekas' auction algorithm, exploits two different levels of parallelism and an efficient method of communicating the data between them that avoids the need to use the router, thus yielding significant speedups over previous implementations [292].

David Williamson

Williamson spent the past year investigating approximation algorithms for several different NP-complete problems.

The main focus of his work has been the Held-Karp heuristic for the Traveling Salesman Problem (TSP) [143]. The heuristic computes a lower bound on the cost of the optimal tour, a bound which in practice seems to be very tight. Previously, Shmoys and Williamson [270] shown that the Held-Karp heuristic on the symmetric TSP with triangle inequality has a certain monotonicity property; namely, the bound produced by the heuristic for a subset of a particular input is no greater than that produced for the input itself. This past year, Williamson extended these results to show that the monotonicity property also holds for the asymmetric case with triangle inequality, and that this property implies that the Held-Karp heuristic produces a value that is no less than $1/\lceil\log n\rceil$ times the cost of the optimal tour in the asymmetric case with triangle inequality. He also showed that in the asymmetric case, the Held-Karp heuristic produces a value that always is at least the value given by another lower-bound heuristic for the TSP due to Balas and Christofides [33]. Additionally, he demonstrated that there are a number of other equivalent formulations of the Held-Karp heuristic in addition to the ones shown by Held and Karp in their original paper. For instance, in the symmetric case with triangle inequality and non-negative edge costs, the value of the Held-Karp heuristic is equal to the value of the linear relaxation of the minimum-cost biconnected-graph problem. Finally, he showed that in the Euclidean case, solutions produced by the Held-Karp heuristic are planar. These results are collected in Williamson's Master's thesis [295], which he completed this spring.

Williamson also worked on flow-shop scheduling, another NP-complete problem. A common approach to approximating the best flow-shop schedule is to find a good permutation of jobs. Potts, Shmoys, and Williamson [250] found a family of instances for which this approach does poorly. In particular, they showed that for these instances the best possible permutation schedule is a factor of $\sqrt{m}/2$ worse than the overall best schedule, where m is the number of machines in the instance.

Finally, Williamson considered a problem from learning theory called the minimum consistent subset problem under the nearest-neighbor rule. Given a collection T of points with labels, a consistent subset $S \subseteq T$ is one that correctly classifies all the points of T under the nearest-neighbor rule. Williamson showed that finding the smallest such set S is NP-complete, and also that it seems unlikely that there is any approximation algorithm that can find a consistent subset of size within any constant factor of the smallest such subset.

Yiqun Yin

This year Yin spent most of her time on course work. She is currently interested in the theoretical aspect of machine learning, and to do some research in the bandit problems during summer.

13.4 Publications[1]

[1] A. Aggarwal and T. Leighton. A tight lower bound for a train reversal problem. *Information Processing Letters*, 1990. To appear.

[2] A. Aggarwal, T. Leighton, and K. Palem. Area-time optimal circuits for iterated addition in VLSI. *IEEE Transactions on Computers*, 1990. To appear.

[3] A. Aggarwal and J. Park. Algorithms for economic lot-size problems with bounds on inventory and backlogged demand. Manuscript in preparation, 1990.

[4] A. Aggarwal and J. Park. Improved algorithms for economic lot-size problems. Research Report RC 15626, IBM Research Division, T. J. Watson Research Center, March 1990.

[5] A. Aggarwal, M. Hansen, and T. Leighton. Solving query-retrieval problems by compacting Voronoi diagrams. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 331–340, 1990.

[6] A. Aggarwal, D. Kravets, J. Park, and S. Sen. Parallel searching in generalized Monge arrays with applications. In *Proceedings of the Second Annual ACM Symposium on Parallel Algorithms and Architectures*, 1990. To appear.

[7] W. Aiello, T. Leighton, B. Maggs, and M. Newman. Fast algorithms for bit-serial routing on a hypercube. In *Proceedings of the Second Annual ACM Symposium on Parallel Algorithms and Architectures*, 1990. To appear.

[8] E. Anagnostou, L. Guibas, and V. Polimenis. Topological sweeping in three dimensions. In *Proceedings of the First SIGAL Symposium*, 1990. To appear.

[9] B. Aronov, B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, and R. Wenger. Points and triangles in the plane and halving planes in space. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*. To appear.

[10] S. Arora, T. Leighton, and B. Maggs. On-line algorithms for path selection in a nonblocking network. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 149–158, 1990.

[11] T. Asano, L. Guibas, and T. Tokuyama. Walking on an arrangement topologically. In preparation.

[12] J. A. Aslam and A. Dhagat. Online algorithms for the Lovász local lemma via chip games. Submitted for publication.

[1]For the more extensive listing of the 1989-90 TOC Group publications, please see the 1990 Technical Memo (to appear) which includes full annotated bibliographies.

[13] J. A. Aslam and R. L. Rivest. Minimum consistent inference of random walks. Submitted for publication.

[14] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, 1989. To appear.

[15] B. Awerbuch, I. Cidon, I. Gopal, M. Kaplan, and S. Kutten. Distributed control for PARIS. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, 1989. To appear.

[16] B. Awerbuch, I. Cidon, and S. Kutten. Optimal maintenance of replicated information. Unpublished manuscript, April 1990.

[17] B. Awerbuch, A. Goldberg, M. Luby, and S. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[18] B. Awerbuch, O. Goldreich, and A. Herzberg. A quantitative approach to dynamic networks. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, 1989. To appear.

[19] B. Awerbuch, Y. Mansour, and N. Shavit. End-to-end communication with polynomial overhead. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[20] B. Awerbuch and D. Peleg. Online tracking of mobile users. Unpublished manuscript, August 1989.

[21] B. Awerbuch and D. Peleg. Routing with polynomial communication-space trade-off. Unpublished manuscript, August 1989.

[22] D. Beaver, J. Feignebaum, J. Kilian, and P. Rogaway. Cryptographic applications of locally random reductions. Unpublished manuscript, 1989.

[23] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 503–513, 1990.

[24] D. Beaver and S. Goldwasser. Multi party fault tolerant computation with faulty majority. In *Advances in Cryptology—CRYPTO '89*, Lecture Notes in Computer Science, Springer-Verlag, 1989.

[25] D. Beaver and S. Goldwasser. Multi party fault tolerant computation with faulty majority based on oblivious transfer. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[26] M. Bellare and S. Goldwasser. New paradigms for digital signature schemes and message authentication based on non-interactive zero knowledge proofs. In *Advances in Cryptology—CRYPTO '89*, Lecture Notes in Computer Science, Springer-Verlag, 1989.

[27] M. Bellare, S. Goldwasser, and J. Rompel. Randomness in interactive proofs. Submitted to the 31st Annual Symposium on Foundations of Computer Science, 1990.

[28] M. Bellare, S. Micali, and R. Ostrovsky. From asyntotics to practice. In *Advances in Cryptology—CRYPTO '90*, Lecture Notes in Computer Science, Springer-Verlag, 1990. To appear.

[29] M. Bellare, L. Cowen, and S. Goldwasser. On the structure of secret key exchange protocols. In J. Feigenbaum and M. Merritt, editors, *Proceedings of the DIMACS Workshop on Distributed Computing and Cryptography*. American Mathematical Society, Princeton, NJ, 1989.

[30] M. Bellare and S. Micali. How to sign given any trapdoor function. *SIAM Journal on Computing*, 1989. Accepted, subject to minor revisions.

[31] M. Bellare, S. Micali, and R. Ostrovsky. Perfect zero-knowledge in constant rounds. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 482–493, 1990.

[32] M. Bellare, S. Micali, and R. Ostrovsky. The (true) complexity of statistical zero knowledge. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 494–502, 1990.

[33] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Efficient identification schemes based on multi prover interactive proofs. In *Advances in Cryptology—CRYPTO '89*, Lecture Notes in Computer Science, Springer-Verlag, 1989.

[34] M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40– 46, January 1990.

[35] J. Bentley, T. Leighton, M. Lepley, D. Stanat, and M. Steele. A randomized data structure for ordered sets. *Advances in Computing Research: Randomness and Computation*, pages 413–428, 1989.

[36] B. Berger. Data structures for removing randomness. Technical Report MIT/LCS/TR-436, MIT Laboratory for Computer Science, December 1988.

[37] B. Berger. Lower bounding discrepancies with fourth moment analysis. Unpublished manuscript, 1990.

[38] B. Berger, M. Brady, D. Brown, and T. Leighton. Nearly optimal algorithms and bounds for multilayer channel routing. *Journal of the ACM*, 1990. To appear.

[39] B. Berger and L. Cowen. $\{<, \leq, =\}$-constrained scheduling. Technical Report CICS-P-236, MIT Center for Intelligent Control Systems, June 1990.

[40] B. Berger and J. Rompel. Simulating ($\log^c n$)-wise independence in NC. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 2–7, IEEE, 1989. Winner of 1989 Machtey Award. To appear in *Journal of the ACM.*

[41] B. Berger, J. Rompel, and P. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 54–59, IEEE, 1989. Submitted to *Journal of Algorithms*, FOCS '89 special issue.

[42] B. Berger and P. Shor. Approximation algorithms for the maximum acyclic subgraph problem. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 236–243, 1990.

[43] F. Berman, D. Johnson, T. Leighton, P. Shor, and L. Snyder. Generalized planar matching. *Journal of Algorithms*, 11(2):153–184, 1990.

[44] D. Bertsimas and M. Grigni. On the space-filling curve heuristic for the Euclidean traveling salesman problem. *Operations Research Letters*, 8:241–244, October 1989.

[45] S. Bhatt, F. Chung, J. Hong, T. Leighton, and A. Rosenbery. Optimal simulations of butterfly networks. *Journal of the ACM*, 1990. To appear.

[46] S. Bhatt, F. Chung, T. Leighton, and A. Rosenbery. Efficient embeddings of trees in hypercubes. *SIAM Journal on Computing*, 1990. To appear.

[47] B. Bloom. Can LCF be topped? Flat lattice models of typed lambda calculus. *Information and Computation*, 87(1/2):263–300, July/August 1990. Also available as Cornell University Department of Computer Science Technical Report 898-1073.

[48] B. Bloom and A.R. Meyer. A remark on the bisimulation of probabilistic processes. In A. R. Meyer and M. A. Taitslin, editors, *Logic at Botik '89: Symposium on Logical Foundations of Computer Science*, Volume 363 of Lecture Notes in Computer Science, pages 26–40, Springer-Verlag, 1989.

[49] B. Bloom and A.R. Meyer. Experimenting with process equivalence. In *Proceedings of the International BCS-FACS Workshop on Semantics for Concurrency*, Lecture Notes in Computer Science. Springer-Verlag, July 1990. To appear.

[50] B. Bloom and J.G. Riecke. LCF should be lifted. In T. Rus, editor, *Proceedings of the Conference on Algebraic Methodology and Software Technology*, pages 133–136. Department of Computer Science, University of Iowa, 1989.

[51] A. Blum. Separating PAC and mistake-bound learning models over the boolean domain, April 1990. Submitted for publication.

[52] A. Blum. Some tools for approximate 3-coloring. April 1990. Submitted for publication.

[53] A. Blum and M. Singh. Learning functions of k terms. April 1990. Submitted for publication.

[54] A. Blum. Learning boolean functions in an infinite attribute space. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 64–72, 1990.

[55] R. Boppana and M. Sipser. The complexity of finite functions. *Handbook of Theoretical Computer Science*, 1990. To appear.

[56] V. Breazu-Tannen and A.R. Meyer. Computable values can be classical. In *Conference Record of the 14th Annual ACM Symposium on Principles of Programming Languages*, pages 238–245, 1987. Reprinted in G. Huet, editor, *Logical Foundations of Functional Programming,* chapter 12, pages 285–295. University of Texas at Austin, Year of Programming Series. Addison-Wesley, 1990.

[57] V. Breazu-Tannen and A.R. Meyer. Polymorphism is conservative over simple types. In *Proceedings of the Symposium on Logic in Computer Science*, pages 7–17, IEEE, 1987. Reprinted in, G. Huet, editor, *Logical Foundations of Functional Programming,* chapter 13, pages 297–314. University of Texas at Austin, Year of Programming Series. Addison-Wesley, 1990.

[58] K. B. Bruce, A. R. Meyer, and J. C. Mitchell. The semantics of second-order lambda calculus. *Information and Computation*, 85(1):76–134, 1990. Reprinted in, G. Huet, editor, *Logical Foundations of Functional Programming,* chapter 10, pages 213–272. University of Texas at Austin, Year of Programming Series. Addison-Wesley, 1990.

[59] B. Chazelle, H. Edelsbrunner, L. Guibas, J. Hershberger, R. Seidel, and M. Sharir. Slimming down by adding; selecting heavily covered points. In *Proceedings of the Sixth Annual Symposium on Computational Geometry.* ACM, 1990. To appear.

[60] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly-exponential stratification scheme for real semi-algebraic varieties and its applications. In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *Automata, Languages and Programming: 16th International Colloquium*, Lecture Notes in Computer Science, pages 179–193. Springer-Verlag, 1989.

[61] B. Chazelle and L. Guibas. Visibility and intersection problems in plane geometry. *Discrete and Computational Geometry*, 4:551–581, 1989.

[62] I. Cidon, S. Kutten, Y. Mansour, and D. Peleg. Greedy packet scheduling. Unpublished manuscript, 1990.

[63] K. Clarkson, H. Edelsbrunner, M. Sharir, L. Guibas, and E. Welzl. Combinatorial complexity bounds for arrangements of curves and surfaces. *Discrete and Computational Geometry*, 5:99–160, 1990.

[64] E. Coffman, L. Flatto, and T. Leighton. First-fit allocation of linear lists: Tight probabilistic bounds on wasted space. *Journal of Stochastic Processes and their Applications*, 1990. To appear.

[65] E. G. Coffman, Jr., L. Flatto, and F. T. Leighton. First-fit allocation of queues: tight probabilistic bounds on wasted space. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 272–279, 1990.

[66] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press/McGraw-Hill, 1990.

[67] S. S. Cosmadakis, A. R. Meyer, and J. G. Riecke. Completeness for typed lazy inequalities. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 312–320, 1990.

[68] R. E. Cypher and C. G. Plaxton. Techniques for shared key sorting. Unpublished manuscript.

[69] R. Cypher and C. G. Plaxton. Deterministic sorting in nearly logarithmic time on the hypercube and related computers. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 193–203, 1990.

[70] H. Edelsbrunner, L. Guibas, J. Hershberger, J. Pach, R. Pollack, R. Seidel, M. Sharir, and J. Snoeyink. On arrangements of Jordan arcs with three intersections per pair. *Discrete and Computational Geometry*, 4:523–539, 1989.

[71] H. Edelsbrunner, L. Guibas, and M. Sharir. The complexity and construction of many faces in arrangements of lines and segments. *Discrete and Computational Geometry*, 5:161–196, 1990.

[72] H. Edelsbrunner, L. Guibas, and M. Sharir. The complexity of many cells in arrangements of planes and related problems. *Discrete and Computational Geometry*, 5:197–216, 1990.

[73] H. Edelsbrunner, L. Guibas, J. Hershberger, R Seidel, M. Sharir, J. Snoeyink, and E. Welzl. Implicitly representing arrangements of lines or segments. *Discrete and Computational Geometry*, 4:433–466, 1989.

[74] H. Edelsbrunner and L. J. Guibas. Topologically sweeping an arrangement. *Journal of Computer and System Sciences*, 38(1):165- 194, 1989.

[75] P. Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 1990. Accepted for publication.

[76] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. In *Advances in Cryptology—CRYPTO '89*, Lecture Notes in Computer Science, Springer-Verlag, 1989.

[77] P. Feldman and S. Micali. Optimal algorithms for Byzantine agreement. In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *Automata, Languages and Programming: 16th International Colloquium*, Lecture Notes in Computer Science, pages 267-276. Springer-Verlag, 1989.

[78] M. Formann, T. Hagerup, J. Haralambides, T. Leighton, A. Simvonis, E. Welzl, and G. Woeginger. Drawing graphs in the plane with high resolution. Submitted to 31st IEEE Symposium on Theory of Computing, April 1990.

[79] W. Goddard, V. King, and L. Schulman. Optimal randomized algorithms for local sorting and set-maxima. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 45–53, 1990.

[80] S. A. Goldman, M. J. Kearns, and R. E. Schapire. Exact identification of circuits using fixed points of amplification functions. April 1990. Submitted for publication.

[81] S. A. Goldman, M. J. Kearns, and R. E. Schapire. On the sample complexity of weak learning. April 1990. Submitted for publication.

[82] S. A. Goldman, R. L. Rivest, and R. E. Schapire. Learning binary relations and total orders. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 46–51, IEEE, 1989. Abstract also appeared in *Proceedings of the Second Annual Workshop on Computational Learning Theory*, July 1989.

[83] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity, or, all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 1990. Accepted for publication.

[84] S. Goldwasser. Interactive proof systems. In J. Hartmanis, editor, *Computational Complexity Theory*, Proceedings of Symposia in Applied Mathematics, 1989.

[85] S. Goldwasser. The search for provably secure cryptosystems. In *Proceedings of AMS Short Course on Computational Number Theory and Cryptography*, Bolder, CO, July 1989. To appear.

[86] S. Goldwasser, editor. *Advances in Cryptology: Proceedings of CRYPTO '88*. Volume 363 of Lecture Notes in Computer Science, Springer-Verlag, February 1990.

[87] S. Goldwasser and L. Levin. Computing general functions fairly in presence of immoral majority. In *Advances in Cryptology—CRYPTO '90*, Lecture Notes in Computer Science, Springer-Verlag, 1990.

[88] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. *Advances in Computing Research*, 1989. Also invited and accepted to *Randomness and Computation*.

[89] M. Grigni and D. Peleg. Tight bounds on minimum broadcast networks. *SIAM Journal on Discrete Mathematics*, 1990. To appear, an earlier version appears as Technical Memo MIT/LCS/TM-374, MIT Laboratory for Computer Science.

[90] M. Grigni and M. Sipser. On monotone complexity classes. In *Proceedings of Conference on the Complexity of Boolean Functions*, 1990. To appear.

[91] L. Guibas, J. Hershberger, and J. Snoeyink. Compact interval trees: a data structure for convex hulls. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 169–178, 1990.

[92] L. Guibas, D. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. In *Automata, Languages and Programming: 17th International Colloquium*, Lecture Notes in Computer Science, Springer-Verlag, 1990. To appear.

[93] L. Guibas, D. Salesin, and J. Stolfi. Constructing strongly convex approximate hulls with inaccurate primitives. In *Proceedings of the First SIGAL Symposium*, 1990. To appear.

[94] L. Guibas, M. Sharir, and S. Sifrony. On the general motion planning problem with two degrees of freedom. *Discrete and Computational Geometry*, 4:491–521, 1989.

[95] M. Hansen, T. Leighton, and C. E. Leiserson. Lecture Notes for *18.436/6.849, Theory of Parallel and VLSI Computation (spring 1989).* MIT Research Seminar Series 7, MIT Laboratory for Computer Science, December 1989.

[96] M.D. Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[97] A. T. Ishii and C. E. Leiserson. A timing analysis of level-clocked circuitry. In *Advanced Research in VLSI: Proceedings of the Sixth MIT Conference*, pages 113–130, April 1990.

[98] G. Itkis and L. A. Levin. Power of fast VLSI models is insensitive to wires' thinness. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 402–407, IEEE, 1989.

[99] L. A. Jategaonkar. Ml with extended pattern matching and subtypes. Technical Report MIT/LCS/TR-468, MIT Laboratory for Computer Science, August 1989.

[100] N. Kahale. New modular properties of the Bell numbers. *Journal of Combinatorial Theory Series A*, 1989. Accepted for publication.

[101] C. Kaklamanis, et al. Asymptotically tight bounds for computing with faulty arrays of processors. Submitted to 31st IEEE Symposium on Theory of Computing, April 1990.

[102] *Communication Complexity: A New Approach to Circuit Depth.* MIT Press, 1989.

[103] M. Kearns and L. Pitt. A polynomial-time algorithm for learning k-variable pattern languages from examples. In *Second Annual Workshop on Computational Learning Theory*, pages 57–71. Morgan Kaufmann, August 1989.

[104] M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. April 1990. Submitted for publication.

[105] J. Kilian and S. Micali. Secret, physically secure computation. In preparation, 1990.

[106] J. Kilian, S. Micali, and R. Ostrovsky. Minimum resource zero-knowledge proofs. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 474–479, IEEE, 1989.

[107] J. Kilian, S. Micali, and R. Ostrovsky. Simple non-interactive zero-knowledge proofs. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[108] J. Kilian, S. Micali, and P. Rogaway. The notion of secure computation. In *Advances in Cryptology—CRYPTO '90*, Lecture Notes in Computer Science, Springer-Verlag, 1990.

[109] J. Kilian. Achieving zero-knowledge robustly. Unpublished manuscript, 1990.

[110] J. Kilian. Interactive proofs with provable security against passive adversaries. Unpublished manuscript, 1990.

[111] J. Kilian. On the power of finite cryptographic games. Unpublished manuscript, 1990.

[112] J. Kilian, S. Kipnis, and C. E. Leiserson. The organization of permutation architectures with bussed interconnections. *IEEE Transactions on Computers*, 1989. To appear. Also Technical Memo MIT/LCS/TM-379 and VLSI memo 89-500. Earlier version appeared in *Proceedings of the 28th IEEE Annual Symposium on Foundations of Computer Science*, pages 305–315, 1987.

[113] S. Kipnis. Priority arbitration with busses. Technical Memo MIT/LCS/TM-408, MIT Laboratory for Computer Science, October 1989. Also in *Sixth MIT Conference on Advanced Research in VLSI*, April 1990.

[114] S. Kipnis. Three methods for range queries in computational geometry. Technical Memo MIT/LCS/TM-388, MIT Laboratory for Computer Science, March 1989.

[115] M. Klawe and T. Leighton. A tight lower bound on the size of planar permutation layouts. In *Proceedings of SIGAL International Symposium on Algorithms*, August 1990. To appear.

[116] P. Klein and C. Stein. A parallel algorithm for approximating the minimum cycle cover. Submitted for publication, 1989.

[117] P. Klein and C. Stein. A parallel algorithm for eliminating cycles in undirected graphs. *Information Processing Letters*, 1990. To appear.

[118] P. Klein, C. Stein, and É. Tardos. Leighton-Rao might be practical: faster approximation algorithms for concurrent flow with uniform capacities. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 310–321, 1990.

[119] D. Kravets, T. Leighton, and C. E. Leiserson. Lecture Notes for *18.435/6.848 Theory of Parallel and VLSI Computation (Fall 1989).* MIT LCS Research Seminar Series 8, MIT Laboratory for Computer Science, May 1990.

[120] D. Kravets and J. K. Park. Selection and sorting in totally monotone arrays. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 494–502, 1990.

[121] T. Leighton. A $2d - 1$ lower bound for 2-layer knock-knee channel routing. *SIAM Journal on Discrete Mathematics*, 1990. To appear.

[122] T. Leighton. Average case analysis of greedy routing algorithms on arrays. In *Proceedings of the Second Annual ACM Symposium on Parallel Algorithms and Architectures*, 1990. To appear.

[123] T. Leighton, D. Lyzinski, and B. Maggs. Empirical analysis of randomly-wired multistage networks. In *Proceedings IEEE International Conference on Computer Design*, 1990. To appear.

[124] T. Leighton and B. Maggs. Expanders might be practical: Fast algorithms for routing around faults in multibutterflies. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 384–389, IEEE, 1989.

[125] T. Leighton, B. Maggs, A. Ranade, and S. Rao. Randomized algorithms for routing and sorting in fixed-connection networks. Submitted to *Journal of Algorithms*, October 1989.

[126] T. Leighton, F. Makedone, and D. Pathria. Efficient reconfiguration on WSI arrays. In *Proceedings First ACM-IEEE International Conference on System Integration*, 1990. To appear.

[127] T. Leighton, F. Makedone, and S. Tragoudas. Approximation algorithms for VLSI partition problems. In *Proceedings of the First ACM-IEEE International Conference on Circuits and Systems*, 1990. To appear.

[128] T. Leighton and C. G. Plaxton. A (fairly) simple circuit that (usually) sorts. Unpublished manuscript, April 1990.

[129] T. Leighton and P. Shor. Tight bounds for minimax grid matching, with applications to the average case analysis of algorithms. *Combinatorica*, 9(2):161–187, 1989.

[130] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 1990. To appear.

[131] F. Makedone, T. Leighton, and I. Tollis. A $2N - 2$ step algorithm for routing on an $N \times N$ array with constant size queues. *J. Algorithmica*, 1990. To appear.

[132] Y. Mansour. Learning *via* Fourier transform. Unpublished manuscript, 1990.

[133] Y. Mansour, N. Nisan, and P. Tiwari. The computational complexity of universal hashing. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 235–243, 1990.

[134] Y. Mansour and L. Schulman. Sorting on a ring of processors. *Journal of Algorithms*, 1989. Accepted for publication.

[135] Y. Mansour and L. Schulman. Sorting on a ring of processors. *Journal of Algorithms*, 1990. To appear.

[136] A. R. Meyer, J. C. Mitchell, E. Moggi, and R. Statman. Empty types in polymorphic λ-calculus. In *Conference Record of the 14th Annual ACM Symposium on Principles of Programming Languages*, pages 253–262, 1987. Reprinted in G. Huet, editor, *Logical Foundations of Functional Programming,* chapter 11, pages 273–284. University of Texas at Austin, Year of Programming Series, Addison-Wesley, 1990.

[137] A. R. Meyer and M. A. Taitslin, editors. *Logic at Botic,'89: Proceedings of a Symposium on Logical Foundations of Computer Science*, Volume 363 of Lecture Notes in Computer Science. Springer-Verlag, 1989.

[138] S. Micali. Perfect pseudo-random generation. In *Proceedings of IFIP 11th World Computer Congress*, pages 121–126, San Francisco, CA, August 1989.

[139] S. Micali and T. Rabin. Collective coin tossing without assumptions nor broadcasting. In *Advances in Cryptology—CRYPTO '90*, Lecture Notes in Computer Science, Springer-Verlag, 1990.

[140] S. Micali, editor. *Randomness and Computation*, Volume of the series "Advances in Computing Research." JAI Press, December 1989.

[141] S. Micali and C. Schnorr. Super efficient, perfect pseudo-random number generators. *Journal of Cryptology*, 1990. Accepted, minor revisions already made.

[142] N. Nisan. Pseudorandom generators for space-bounded computation. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 204–212, 1990.

[143] C. Norton, S. Plotkin, and E. Tardos. Using separation algorithms in fixed dimension. Technical Report 866, School of Operations Research and Industrial Engineering, Cornell University, October 1989.

[144] C. Norton, S. Plotkin, and E. Tardos. Using separation algorithms in fixed dimension. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 377–387, 1990.

[145] R. Ostrovsky, R. Venkatesan, and M. Yung. On the complexity of asymmetric games. Unpublished manuscript, December 1989.

[146] R. Ostrovsky. Efficient computation on oblivious RAMs. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 514–523, 1990.

[147] C. G. Plaxton. On the network complexity of selection. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 396–401, IEEE, 1989.

[148] C. N. Potts, D. B. Shmoys, and D. P. Williamson. Permutation vs. non-permutation flow shop schedules. Unpublished manuscript.

[149] M. B. Reinhold. Typechecking is undecidable when 'type' is a type. Technical Report MIT/LCS/TR-458, MIT Laboratory for Computer Science, December 1989.

[150] J. G. Riecke. A complete and decidable proof system for call-by-value equalities (preliminary report). In *Automata, Languages and Programming: 17th International Colloquium*, Lecture Notes in Computer Science, Springer-Verlag, 1990. To appear.

[151] R. L. Rivest. Cryptography. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North-Holland, 1990. To appear.

[152] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, 1990. Winner of STOC '90 Best Student Paper Award.

[153] R. E. Schapire. The strength of weak learnability. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 28–33, IEEE, 1989. Abstract also appeared in *Proceedings of the Second Annual Workshop on Computational Learning Theory*, July 1989. Also to appear in *Machine Learning.*

[154] R. E. Schapire. Pattern languages are not learnable, April 1990. Submitted for publication.

[155] E. J. Schwabe. Normal hypercube algorithms can be simulated on a butterfly with only constant slowdown. Submitted to *IPL*, 1989.

[156] E. J. Schwabe. On the computational equivalence of hypercube-derived networks. In *Proceedings of the Second ACM Symposium on Parallel Algorithms and Architectures, 1990.*

[157] E. J. Schwabe. On the computational equivalence of hypercube-derived networks. In *Proceedings of the Second Annual ACM Symposium on Parallel Algorithms and Architectures*, 1990.

[158] D. B. Shmoys and D. P. Williamson. Analyzing the Held-Karp TSP bound: a monotonicity property with application. To appear in *Information Processing Letters.*

[159] The Committee on Family and Work, P. Elias, Chairman. Report of the MIT Committee on Family and Work: Summary and Recommendations (Preliminary). May 1990. To appear in *TechTalk.*

[160] The Committee on Family and Work, P. Elias, Chairman. Report of the MIT Committee on Family and Work: Analysis of Survey Findings (Preliminary). May 1990. To appear in *TechTalk.*

[161] J. Wein. Las Vegas RNC algorithms for unary weighted matching and T-join problems. Submitted to *Information Processing Letters.*

[162] J. Wein and S. Zenios. Massively parallel auction algorithms for the assignment problem. Submitted to the Third Symposium on the Frontiers of Massively Parallel Computation.

Theses Completed

[1] B. Berger. *Using Randomness to Design Efficient Deterministic Algorithms.* PhD thesis, MIT Department of Electrical Engineering and Computer, May 1990. Supervised by S. Micali.

[2] B. Bloom. *Ready Simulation, Bisimulation, and the Semantics of CCS-like Languages.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, August 1989. Supervised by A. R. Meyer.

[3] S. A. Goldman. *Learning Binary Relations, Total Orders, and Subclasses of Read-once Formulas.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, June 1990. In preparation. Supervised by R. Rivest. .

[4] L. A. Jategaonkar. *ML with Extended Pattern Matching and Subtypes.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, August 1989. Supervised by A.R. Meyer.

[5] A.F. Lent. *A lazy SECD Machine.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, January 1990. Supervised by A. R. Meyer.

[6] A. Rudich. *Finite Observation Semantics for Dataflow Networks.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990. Supervised by A. R. Meyer.

[7] C. Stein. *Using Cycles and Scaling in Parallel Algorithms.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, August 1989. Also appears as MIT/LCS/TR-457.

[8] D. P. Williamson. *Analysis of the Held-Karp Heuristic for the Traveling Salesman Problem.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

Talks

[1] B. Berger. Simulating $(\log^c n)$-wise independence in NC. Lecture given at 30^{th} Annual Symposium on Foundations of Computer Science, 1989.

[2] B. Berger. Approximation algorithms for the maximum acyclic subgraph problem. Lecture given at First Annual ACM-SIAM Symposium on Discrete Algorithms, 1990.

[3] A. Blum. Learning boolean functions in an infinite attribute space. Lecture given at 22nd Annual ACM Symposium on Theory of Computing, May 1990.

[4] A. Blum. On approximate 3-coloring. Lecture given at Coping with NP-Completeness, a Workshop at UCSD, January 1990.

[5] S. A. Goldman. Learning binary relations and total orders. Lecture given at Second Annual Workshop on Computational Learning Theory, 30th Annual Symposium on Foundations of Computer Science, 1989.

[6] S. A. Goldman. Learning binary relations and total orders. Lecture given at Bell Communications Research, Washington University, 1990.

[7] S. Goldwasser. Number theory and cryptography. Lecture given at AMS, Boulder, CO, August 1989 (Lecture notes will be assembled into a book by the American Mathematical Society).

[8] S. Goldwasser. Interactive proofs. Invited speaker at International Congress of Mathematicians (ICM), Japan, August 1990.

[9] S. Goldwasser. Invited speaker at British Colloquium on Computer Science Manchester, Britain, March 1990.

[10] S. Goldwasser. Lecture given at Oberwolfach Conference on Cryptography, Germany, October 1989.

[11] S. Goldwasser. On the structure of secret key exchange protocols. Lecture given at DIAMICS Workshop on Distributed Computing and Cryptography, October 1989.

[12] S. Goldwasser. Lecture given at Workshop on Circuit Complexity in Barbados, organized by McGill University, February 1990.

[13] L. Guibas. Epsilon geometry: building robust algorithms from imprecise computations. Lecture given at University of Tokyo, and Ricoh Software Research Center, Tokyo, Japan, November 1989.

[14] L. Guibas. Randomized incremental construction of Delaunay and Voronoi diagrams. Lecture given at IBM Tokyo Research Laboratory, Tokyo, Japan, November 1989.

[15] L. Guibas. Some combinatorial covering lemmas and their geometric applications. Lecture given at University of California, Berkeley, September 1989.

[16] L. Guibas. The use of arrangements in geometric computation. Lecture given at Asian Institute of Technology, Bangkok, Thailand (November 1989); NEC Central Laboratories, Kawasaki, Japan (March 1990); Harvard University (May 1990).

[17] L. Guibas. The uses of computational geometry in computer graphics and solid modeling. Lecture given at Istituto per la Matematica Applicata, Genova, Italy, July 1989.

[18] L. Guibas. The use of randomization in geometric algorithms. Lecture given at INRIA, Sophia-Antipolis, France, June 1990.

[19] R. Impagliazzo and L. Levin. Average case NP-completeness for all samplable distributions. Lecture given at Berkeley, April 1989.

[20] A. T. Ishii. A timing verification algorithm for level-clocked circuitry. Lecture given at University of California at Berkeley, and the Sixth MIT Conference on Advanced Research in VLSI, March 1990.

[21] M. Kearns. A polynomial-time algorithm for learning k-variable pattern languages from examples. Lecture given at the Workshop on Computational Learning Theory, August 1989.

[22] M. Kearns. Cryptographic limitations on learning. Lecture given at Columbia University (February); Also at Cornell University (March); Brown University (March), 1990.

[23] M. Kearns. Efficient distribution-free learning of probabilistic concepts. Lecture given at Bellcore (February 1990); International Computer Science Institute (February 1990); University of California at Santa Cruz (February 1990); University of California at San Diego (June 1990).

[24] S. Kipnis. Organization of systems with bussed interconnections. Lecture given at IBM Almaden Research Center, IBM Yorktown Research Center, Philips Laboratories, Digital Systems Research Center, AT&T Bell Laboratories, Columbia University, 1989.

[25] S. Kipnis. Priority arbitration with busses. Lecture given at DARPA Fall 1989 Microsystems Contractors Meeting, November 1989.

[26] T. Leighton. Fast fault-tolerant algorithms for routing on multibutterflies and nonblocking networks. Lecture given at the Third International Conference on Graph Theory, San Francisco (July 1989); DARPA Symposium, Washington, DC (August 1989); IBM Watson Computer Science Group (October 1989); CMU Computer Science Department (December 1989); NECUSE Annual Meeting at Amherst (January 1990); MIT LIDS (February 1990); DARPA contractors Meeting, Salt Lake City (March 1990); Thinking Machines Corporation (May 1990); MIT (May 1990); Workshop on Parallel Algorithms, Annapolis (May 1990).

[27] L. Levin. Pseudo-random bits: full security at last. Lecture given at Stanford (April 1989); NSF-CBMS Circuit Complexity Workshop; University of Chicago (August 1989); Distinguished Lecturer Series, SUNY at Stony Brook (November 1990).

[28] B. Maggs. Fault-tolerant routing algorithms for multibutterfly networks. Lecture given at Yale University (December 1989); Duke University (February 1990); Microelectronics Consortium of North Carolina (February 1990); Harvard University (April 1990).

[29] A. R. Meyer. Completeness for flat, simply typed lazy $\lambda\Omega$- calculus. Lecture given at MSRI, Berkeley (November 1989), Workshop on Logic from Computer Science.

[30] A. R. Meyer. Intereaving semantics of concurrent processes. Lecture given at Logic at Botic '89 Symposium, Pereslavl- Zalesskii, USSR, July 1989.

[31] A. R. Meyer. Theory of programming: A position statement. Lecture given at ACM/CRB Meeting on Strategic Directions in Computing, Washington, DC, October 1989.

[32] A. R. Meyer. Why standard semantics of lambda calculus does not model programming language behavior and what to do about it. Lecture given at Department of Computer Science, UC San Diego, March 1990. Distinguished Lecture Series.

[33] S. Micali. Digital signatures: The evolution of a fundamental primitive. Lecture given at CRYPTO 89, August 1989.

[34] S. Micali. How to collectively flip a coin. Lecture given at Scuola Normale Superiore (July); CRYPTO 89, Santa Barbara, CA (August), 1989.

[35] S. Micali. On-line/off-line digital signatures. Lecture given at CRYPTO 89, August 1989.

[36] S. Micali. An optimal algorithm for Byzantine agreement from scratch. Lecture given at 16th International Colloquium on Automata, Languages and Programming, July 1989. Also given at Technion (January); Boston University (spring); Carnegie Mellon University (October), 1989.

[37] S. Micali. Perfect pseudo-random number generation. Lecture given at 16th International Colloquium on Automata, Languages and Programming, July 1989.

[38] S. Micali. Perfect zero-knowledge, constant-round proof for graph isomorphism. Lecture given at University of Toronto, June 1989.

[39] S. Micali. Proving properties of physically hidden data. Lecture given at International Computer Science Institute (August); Workshop on Cryptography (September), 1989.

[40] S. Micali. Card games are universal. Lecture given at University of Rome, University of Toronto, January 1990.

[41] S. Micali. Interactive proofs. Lecture given at American Association for the Advancement of Science, February 1990.

[42] S. Micali. Probabilistic proofs and their applications. Lecture given at American Association for the Advancement of Science Meeting, New Orleans, LA, February 1990.

[43] S. Micali. Proofs, knowledge and computation. Lecture given at Yale University, February 1990.

[44] S. Micali. The round complexity of secure protocols. Lecture given at Princeton University, May 1990.

[45] S. Micali. Verifiable secret sharing. Lecture given at MIT Laboratory for Computer Science, May 1990.

[46] S. Micali. Zero-knowledge proofs and their applications. Lecture given at Harvard University, Yale University, April 1990.

[47] C. Norton. Using separation algorithms in fixed dimension. Lecture given at First Annual ACM-SIAM Symposium on Discrete Algorithms, 1990.

[48] L. Levin and O. Goldreich. A hard-core predicate for all one-way functions. Lecture given at Cryptography Conference at Oberwolfach, Germany, September 1989.

[49] L. Levin, R. Impagliazzo, and M. Luby. Pseudo-random number generation from any one-way function. Lecture given at Minimal Length Encoding Workshop, Stanford, March 1990.

[50] R. Ostrovsky. Comparison of bit-commitment and oblivious transfer protocols when players have different computing power. Lecture given at DIMACS Workshop, Princeton, NJ, October 1989.

[51] R. Ostrovsky. Efficient computation on oblivious rams. Lecture given at MIT Laboratory for Computer Science, December 1989.

[52] R. Ostrovsky. On protection of traffic-patterns in a distributed database. Lecture given at DIMACS workshop, Princeton, NJ, October 1989.

[53] R. Ostrovsky. Perfect zero-knowledge in constant rounds. Lecture given at Harvard University, November 1989.

[54] J. Park. The Monge array: An abstraction and its applications. Lecture given at the Ohio State University, Columbus, OH (February); Sandia National Laboratories, Albuquerque, NM (March), 1990.

[55] J. Park. Selection and sorting in totally monotone arrays. Lecture given at First Annual ACM-SIAM Symposium on Discrete Algorithms, 1990.

[56] C. G. Plaxton. Deterministic sorting in nearly logarithmic time on the hypercube and related computers. Lecture given at Cornell University (October 1989), Duke University (February 1990), Harvard University (March 1990).

[57] C. G. Plaxton. On the network complexity of selection. Lecture given at IBM Almaden Research Center, January 1990.

[58] J. G. Riecke. A complete and decidable proof system for call-by-value equalities. Lecture given at University of Pennsylvania, Philadelphia, PA, May 1990.

[59] R. L. Rivest. Recent developments in machine learning theory. Lecture given at Bar-Ilan University, and Foundations of Artificial Intelligence Conference, June 1989.

[60] R. L. Rivest. Two developments in machine learning theory. Lecture given at MIT LCS 25th Anniversary Symposium, November 1989.

[61] R. L. Rivest. Public-key cryptography. Lecture given at SECURICOM 90 Conference Paris, France, March 1990.

[62] R. L. Rivest. Two results in machine learning theory: on learning binary relations and on improving a learning algorithm. Lecture given at MIT, ARO Research Review, March 1990.

[63] P. Rogaway. Cryptographically secure distributed computation in a constant number of rounds. Lecture given at DIMACS workshop, Princeton, October 1989.

[64] J. Rompel. Efficient NC algorithms for set cover with applications to learning and geometry. Lecture given at 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, NC, October 1989.

[65] J. Rompel. One-way functions are necessary and sufficient for secure signatures. Lecture given at 22nd Annual ACM Symposium on Theory of Computing, Baltimore, MD (May 1990); University of California, Berkeley (November 1989).

[66] R. E Schapire. The strength of weak learnability. Lecture given at Second Annual Workshop on Computational Learning Theory, 30th Annual Symposium on Foundations of Computer Science, MIT Laboratory for Computer Science, 1989.

[67] R. E. Schapire. Unsupervised learning of deterministic environments. Lecture given at Harvard University Natural Information Processing Seminar series, 1989.

[68] L. Schulman. Optimal randomized algorithms for local sorting and set-maxima. Lecture given at the 22nd Annual ACM Symposium on Theory of Computing, May 1990.

[69] E. J. Schwabe. On the computational equivalence of hypercube-derived networks. Lecture given at Second Annual ACM Symposium on Parallel Algorithms and Architectures, July 1990.

[70] E. J. Schwabe. On the computational equivalence of hypercube-derived networks. Lecture given at Second Annual ACM Symposium on Parallel Algorithms and Architectures, July 1990.

[71] C. Stein. Leighton-Rao might be practical: faster approximation algorithms for concurrent flow with uniform capacities. Lecture given at 22nd Annual ACM Symposium on Theory of Computing, May 1990.

[72] C. Stein. A new parallel graph decomposition technique with applications to finding a cycle cover. Lecture given at Fifth SIAM Conference on Discrete Mathematics, June 1990.

[73] L. Stockmeyer. Bounds on the time to reach agreement in the presence of timing uncertainty. Lecture given at TDS Group Meeting, MIT LCS (February); IBM T.J. Watson Research Center (March); Yale University (April); Cornell University (April) 1990.

Theory of Distributed Systems

Academic Staff

N. Lynch, Group Leader

Research Staff

H. Attiya	A. Fekete
C. Dwork	

Graduate Students

K. Goldman	K. Streeter
J. Leo	G. Troxel
S. Ponzio	M. Tuttle
I. Saias	G. Varghese

Undergraduate Students

M. Gupta

Support Staff

A. Wiseman

14.1 Introduction

The Theory of Distributed Systems research group continued its work on algorithms and impossibility results for distributed problems, as well as its work on modeling, proof techniques and applications. Particular highlights this year include results on timing-based computing and the development of the Spectrum system for simulating distributed algorithms.

14.2 Faculty Reports

Nancy A. Lynch

This year, Nancy Lynch worked mainly on upper and lower time bound results for timing-based and asynchronous systems, as well as the development of models and proof techniques for timing-based systems. First, she completed work with Hagit Attiya, begun last year, on upper and lower bounds for the mutual exclusion problem in a timing-based setting [21]. Second, again working with Attiya, she developed a new mapping technique for proving correctness and timing properties of timing-based algorithms [206]. Third, she worked with Attiya, Dwork, and Stockmeyer to prove upper and lower time bounds for the problem of distributed consensus in the presence of processor faults [20]. Fourth, she worked with Attiya and Shavit to prove upper and lower time bounds for the problem of wait-free approximate agreement. The main point of this work is to show that wait-free algorithms are inherently more time consuming than non-wait-free algorithms, even in the "normal" case where no processors fail [22]. The last three of these sets of results are all discussed in the report of Attiya.

Lynch continued her work on the theory of atomic transactions. One major accomplishment, done jointly with Alan Fekete and Bill Weihl [111], is a new result showing how some of the standard techniques of the "classical" theory of database concurrency control can be used to help prove the stronger "user-view" notion of database correctness described in [110]. Another accomplishment is an almost-completed revision of work with Ken Goldman on modeling replicated data algorithms [122]. Also, she helped revise an earlier paper on modeling locking algorithms [109], for publication in a special conference issue of *JCSS*.

Lynch continued her work of last year on algorithms and impossibility results for data link behavior. The work presented in [207] on the impossibility of implementing reliable data link behavior in the presence of crashes was simplified and submitted for publication. With Fekete, Lynch obtained a new result showing the impossibility of transmitting any data without message headers [108]. Work in progress involves combining the remaining result of [207], showing impossibility of reliable data link message transmission in the presence of bounded headers, with a contrasting result of Hagit Attiya, Mike Fischer, Lenore Zuck and Da-Wei Wang in which such message delivery is achieved; the difference is that the impossibility result requires an assumption, violated by the algorithm, that the "best case" message delivery time be bounded by a constant. An interesting sidelight is that the correctness proof of this algorithm (which is not at all obvious) has been verified automatically by Tobias Nipkow, a visitor from Cambridge, England, using the Isabelle automatic theorem-prover.

Lynch also worked with Ken Goldman on a method of modeling asynchronous shared memory algorithms within the I/O automaton model [122].

Lynch began a consulting project with Digital Equipment Corporation on modeling, specification and verification for timing-dependent communication protocols, and one with Draper Labs on fault-diagnosis algorithms.

Lynch's professional service activities included the following:

1. Working on the committee to select the ACM thesis award winners.
2. Formulating (with Mike Fischer) a set of recommendations to the Computing Research Board for the formation of a new Committee on the Status of Women in Computer Science.
3. Serving on this year's panel to select the Presidential Young Investigators.
4. Conducting a review of the Computer Science Department at the University of Tennessee.
5. Planning this year's POCS colloquium series.

With Weihl, Butler Lampson, and John Guttag, Lynch also worked on developing a new course on "Principles of Computer Systems." In addition to supervising her own students, she also served as thesis reader for Bard Bloom.

14.3 Research Associate and Student Reports

Hagit Attiya

Hagit Attiya continued to work on timing properties of distributed systems. Together with Nancy Lynch, Hagit developed a new technique for proving timing properties for timing-based algorithms; it is an extension of the mapping techniques previously used in proofs of safety properties for asynchronous concurrent systems. The key to the method is a way of representing a system with timing constraints as an automaton whose state includes predictive timing information. Timing assumptions and timing requirements for the system are both represented in this way. A multivalued mapping from the "assumptions automaton" to the "requirements automaton" is then used to show that the given system satisfies the requirements. The technique is illustrated with two simple examples, a resource manager and a signal relay system, and a third, more complex example of a two-process race system. The technique is shown to be *complete*, that is, if some automaton with certain timing assumptions has certain timing behavior, than there exists a mapping from the "assumptions automaton" to the "requirements automaton." (These results appear in [206].)

Together with Cynthia Dwork, Nancy Lynch and Larry Stockmeyer, Hagit studied upper and lower bounds for the time complexity of the problem of reaching agreement in a distributed network, in the presence of process failures and uncertain information about time [20]. It is assumed that the amount of (real) time between any two consecutive steps of any nonfaulty process is at least c_1 and at most c_2; thus, $C = c_2/c_1$ is a measure of the timing uncertainty. It is also assumed that the time for message delivery is at most d. Processes are assumed to fail by stopping, so that process failures can be detected by timeouts. A straightforward adaptation of a $(t+1)$-round synchronous agreement algorithm takes time $(t+1)Cd$ if there are t faults, while a straightforward reduction from a timing-based algorithm to a synchronous algorithm yields a lower bound of $(t+1)d$. The main result is an agreement algorithm in which the uncertainty factor C is only incurred for *one* round, yielding a running time of approximately $2td + Cd$ in the worst case. A second result shows that any agreement algorithm must take time at least, approximately, $(t-1)d + Cd$ in the worst case.

Hagit also continued to study the issue of fault-tolerance in various asynchronous distributed systems.

Together with Yehuda Afek, Danny Dolev, Eli Gafni, Michael Merritt and Nir Shavit, Hagit developed a wait-free algorithm for obtaining atomic snapshots of shared memory, using only bounded amount of additional memory [2]. An *atomic snapshot memory* is a shared data structure allowing concurrent processes to store information in a collection of shared registers, all of which may be read in a single atomic *scan* operation. They present three wait-free implementations of atomic snapshot memory. Two constructions implement wait-free single-writer atomic snapshot memory from wait-free atomic single-writer, n-reader registers. A third construction implements a wait-free n-writer atomic snapshot memory from n-writer, n-reader registers. The first implementation uses *unbounded* (integer) fields in these registers, while the other implementations use only *bounded* registers. All operations require $\Theta(n^2)$ reads and writes to the component shared registers in the worst case.

In joint work with Amotz Bar-Noy and Danny Dolev, Hagit developed a method of emulating wait-free asynchronous algorithms that communicate via shared-memory in two different message-passing systems [19]. The two message-passing models considered are a complete network with processor failures and an arbitrary network with dynamic link failures. The emulations are achieved by implementing a wait-free, atomic, single-writer multi-reader register in unreliable, asynchronous networks. The overhead introduced by these emulations is polynomial in the number of processors in the systems. Any wait-free algorithm based on atomic, single-writer multi-reader registers can be automatically emulated in message-passing systems. Immediate new results are obtained by applying the emulators to known shared-memory algorithms. These include, among others, protocols to solve the following problems in the message-passing model in the presence of processor or link failures: multi-writer multi-reader atomic registers, concurrent time-stamp systems, ℓ-exclusion, atomic snapshots, randomized consensus, and implementation of a class of data structures.

Together with Nancy Lynch and Nir Shavit, Hagit explored the *time complexity* of wait-free algorithms for approximate agreement in "normal" executions, where no failures occur and processes operate at approximately the same speed. A lower bound of $\log n$ on the time complexity of any wait-free algorithm that achieves *approximate agreement* among n processes is proved. In contrast, there exists a (non-wait-free) algorithm that solves this problem in constant time. This implies an $\Omega(\log n)$ time separation between the wait-free and non-wait-free computation models. Two fast wait-free approximate agreement algorithms are presented, a constant time 2-process algorithm and an $O(\log n)$ time n-process algorithm; the complexity of the latter algorithm is within a small constant of the lower bound. (These results appear in [22].)

Cynthia Dwork

(On Sabbatical from IBM Almaden)

Since arriving at MIT in September 1989 Dwork has:

1. Improved lower bounds on connectivity necessary for perfectly secure message transmission in a general network in certain adversary models.
2. Identified and studied the effects of cooperation between two adversaries, the *disruptor* and the *listener* on secret computation and secure message transmission.
3. Defined a stronger notion of secrecy in the presence of Byzantine faults than the one generally studied in the literature, and obtained protocols for verifiable secret sharing and secret computation at no increase in processors, and at no significant increase in computation or communication costs.
4. Obtained almost tight upper and lower bounds on the time needed to reach Byzantine agreement in the presence of fail-stop faults in a model of distributed computation in which there are known upper bounds on message delivery time and known upper and lower bounds on process step time. This work, joint with Attiya, Lynch, and Stockmeyer, is discussed in the report of Attiya.
5. Served as Program Chairperson for the Ninth Annual ACM Symposium on Principles of Distributed Computing.
6. Revised four articles in response to referee's comments. Of these, "A Time Complexity Gap for 2-way Probabilistic Finite State Automata," written with Stockmeyer, has been accepted for publication in *SIAM Journal of Computing*; and "Shifting Gears: Changing Algorithms on the Fly to Expedite Byzantine Agreement," written with Bar-Noy, Dolev, and Strong, has been accepted for publication in *Information and Computation.*

Alan Fekete

Alan Fekete (University of Sydney) spent six weeks in July and August 1989 visiting the Theory of Distributed Systems research group. His research concentrated on reasoning about nested transaction systems with special attention given to the following topics: verifying replication management algorithms with weaker correctness conditions than external consistent serializability, understanding the assumptions made in conventional serializability theory, and optimistic locking techniques. He also was involved in research on the possibility and impossibility of communication protocols using unreliable communication media.

Ken Goldman

Kenneth Goldman is a Ph.D. student in the Theory of Distributed Systems research group. His thesis presents the Spectrum Simulation System, a new research tool for the design and study of distributed algorithms. Based on the formal Input/Output Automaton model of

Lynch and Tuttle, this research tool allows one to express distributed algorithms as collections of I/O automata and simulate them directly in terms of the semantics of that model. This permits integration of algorithm specification, design, debugging, analysis, and proof of correctness within a single formal framework that is natural for describing distributed algorithms. The research tool provides a language for expressing algorithms as I/O automata, a simulator for generating algorithm executions, and a graphics interface for constructing systems of automata and observing their executions.

Goldman has shown that the properties of the I/O automaton model provide a solid foundation for algorithm development tools. For example, using I/O automaton *composition*, Spectrum users may define composed types hierarchically, study simulations at varying levels of detail, and create specialized debugging and analysis devices. These devices, called *spectators*, are written in the Spectrum language just as any other system component, and can monitor algorithm executions for correctness and performance without interfering with the algorithm. Spectators are made possible only by the nonblocking, synchronous, multiparty communication provided in the Spectrum system. Also, since the message system is modeled explicitly as an automaton, users may study algorithms under different communications assumptions simply by substituting one automaton type for another. Techniques used to prove the correctness of an algorithm can also be used as debugging tools. For example, the system checks state invariants during execution and allows users to roll back an execution to discover the source of errors. Several researchers have successfully used the system to simulate and debug algorithms (for examples, see [138][191]).

Also in Goldman's thesis, the I/O automaton model is extended for both *shared memory* [122] and *superposition* [121]. Possible extensions of the Spectrum Simulation System for shared memory and superposition are discussed. In addition, an algorithm for distributing the simulation system is presented in [120].

Upon completing his degree, Goldman will join the faculty in the Computer Science Department at Washington University in St. Louis.

John Leo

John Leo has completed his Master's thesis "Dynamic Process Creation in a Static Model." The thesis shows that proofs of correctness of algorithms involving dynamic process creation and changing topologies can be handled rigorously within a static model, in particular, I/O automata [208]. It is also shown that Actors [9] can be modeled using I/O Automata. Additional proof techniques are developed and demonstrated.

Steve Ponzio

In addition to coursework and reading, Stephen continued his study of timing-based algorithms. He extended his work on the dining philosophers problem, and developed a model of bounded-capacity message links for which he showed a lower bound on the time required to detect stopping failures.

Isaac Saias

Isaac Saias joined the Theory of Distributed Systems group in September 1989. Isaac is currently studying the time complexity of Rabin's probabilistic algorithm for mutual exclusion.

In this work he derives upper bounds for the expected time of a round in presence of an adversarial scheduler.

Isaac also worked with Nancy Lynch and Stuart Adams on the modeling of a complex system built up of potentially failing processors. They investigate different scheduling of repairs when provided with partial information about the system in order to maximize the expected life time.

Ken Streeter

Kenneth Streeter is continuing work on his Master's thesis "A Partitioned Computation Machine." His thesis is being supervised by both Nancy Lynch and Paul Brown, his company advisor in the VI-A Program with General Electric Corporate Research and Development. The thesis develops a specification language which utilizes a pictoral representation language extending Harel's statecharts [140]. The model is closely related to I/O automata [209] and could be used as a visual specification language for I/O automata. A formal execution semantics and methods of *additive* and *multiplicative* composition of partitioned computation machines are developed and demonstrated.

Greg Troxel

Greg Troxel worked and completed his Master's thesis:

An algorithm for detecting and recovering from deadlock in a system using remote procedure calls is presented, along with a proof of correctness. The proof uses the I/O automata model of Lynch and Tuttle, described in [209] and [208]. First, correctness conditions for the problem are given in terms of I/O automata. Next, a high level graph-theoretic representation of the algorithm is shown to be correct. Then a lower level formulation of the algorithm, taking into account its distributed nature, is shown to be equivalent to the higher level representation, and thus correct.

In giving the correctness conditions, we introduce *client automata*, which model the behavior of the user's program, and allow almost all details of this user program to be suppressed at both specification and proof time.

To simplify the proof of the high level version of the algorithm, safety properties are proved with a simplified version of the algorithm. Then, the algorithm is transformed to the full version, and it is argued that the safety properties hold for the transformed version.

A new technique that can be used either for expanding the number of algorithms to which a proof applies or for simplifying the proof that a lower level algorithm solves the same problem as a higher level one is presented. This is effected by underspecifying the predicates used in the preconditions of the I/O automata. This captures the concept that whether or not the algorithm takes a certain action under an intermediate range of conditions, it does not impact its correctness. This lack of specification can be carried through to the low level algorithm, presumably making the job of the implementor easier or allowing more efficient code, since then at times arbitrary choices may be made. It can also be used to make showing that the low level algorithm solves the same problem as the high level one easier.

The proof of the liveness properties of the high level version of the algorithm makes use of a metric on states of the algorithm. Rather than the conventional technique of claiming that the set of values of the metric is well founded, we show that every subset of such values occurring in a particular execution of the algorithm is well founded. This enables us to allow the user program to request an arbitrary but finite number of remote procedure calls with arbitrary arguments.

We then present the low level version of the algorithm, along with specifications for the communications network, etc. used by it. A proof is presented showing that the low level version of the algorithm (together with the network, etc.) is equivalent (from the point of view of the user's program) to the high level version.

Mark Tuttle

Mark Tuttle finished his Ph.D. thesis "Knowledge and Distributed Computation" in September 1989. The topic of the thesis is the role of formal definitions of knowledge in the design and analysis of distributed algorithms. The thesis shows how reasoning in terms of standard definitions of knowledge can lead to fast solutions to problems like consensus and the distributed firing squad problems, and how to construct new definitions of knowledge that seem to be useful in cryptography and areas where bounds on processors' computational powers limit what they can know.

George Varghese

George Varghese joined TDS when he entered MIT as a full time graduate student in February 1990. He worked with Nancy Lynch, and Art Harvey and Radia Perlman (at DEC) on the models and proofs of various transport and routing layer protocols. Recently, he has been working on a randomized version of the two Generals problem to produce lower and upper bounds on probabilistic safety for different adversarial models.

Undergraduate

Aparna Gupta

Aparna worked with Nancy Lynch on her undergraduate thesis this past term. Three algorithms—the Hirshberg-Sinclair leader election, the Peterson leader election, and the Dijkstra shortest paths algorithm—were described formally using Lynch and Tuttle's I/O automaton model. These descriptions were then coded into Ken Goldman's Spectrum simulation in order to gain an intuitive feel for the working of the algorithms. The final thesis discussed three points. The first one is the ease of translating the algorithms into the I/O automaton model and in the Spectrum programming language, and what is gained by each description. The second one is possible changes to the Spectrum interface which would enhance its ease of use and utility. And the final one is recommendations for further studies facilitated by both methods of description.

14.4 Publications

[1] Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt, and N. Shavit. Atomic snapshots of shared memory. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, Quebec, Canada, August 1990. Also, Technical Memo MIT/LCS/TM-429, MIT Laboratory for Computer Science, May 1990. Submitted to *Journal of the ACM.*

[2] H. Attiya, A. Bar-Noy, and D. Dolev. Sharing memory robustly in message-passing systems. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, Quebec, Canada, August 1990. Expanded version: Technical Memo MIT/LCS/TM-423, MIT Laboratory for Computer Science, February 1990. Submitted to *Journal of the ACM.*

[3] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischu. Renaming in an asynchronous environment. *Journal of the ACM*, 37(3), July 1990. To appear.

[4] H. Attiya, C. Dwork, N. A. Lynch, and L. J. Stockmeyer. Bounds on the time to reach agreement in the presence of timing uncertainty. In preparation.

[5] H. Attiya, M. Fischer, D. Wang, and L. Zuck. Reliable communication over an unreliable channel. In progress.

[6] H. Attiya and N. Lynch. Time bounds for real-time process control in the presence of timing uncertainty. In *Proceedings of the Tenth IEEE Real-Time Systems Symposium*, Santa-Monica, CA, December 1989. Expanded version: Technical Memo MIT/LCS/TM-403, MIT Laboratory for Computer Science, July 1989. Submitted for publication.

[7] H. Attiya, N. Lynch, and N. Shavit. Are wait-free algorithms fast? Submitted for publication.

[8] H. Attiya and M. Snir. Better computing on the anonymous ring. *Journal of Algorithms.* To appear.

[9] J. Burns and N. Lynch. Mutual exclusion using indivisible reads and writes. In *Proceedings of 18th Annual Allerton Conference on Communications, Control, and Computing*, pages 833–842, 1980. Submitted for publication.

[10] A. Fekete, N. Lynch, Y. Mansour, and J Spinelli. *The Data Link Layer: The Impossibility of Implementation in Face of Crashes.* Technical Memo MIT/LCS/TM-355.b, MIT Laboratory for Computer Science, August 1989. Submitted for publication.

[11] A. Fekete, N. Lynch, M. Merritt, and W. Weihl. Commutativity-based locking for nested transactions. In *Proceedings of Third International Workshop on Persistent Object Systems*, pages 113–127, Newcastle, Australia, January 1989. Revised version to appear in *JCSS.*

[12] A. Fekete and N. Lynch. The need for headers: an impossibility result for communication over unreliable channels. Submitted for publication. Also Technical Memo MIT/LCS/TM-428, MIT Laboratory for Computer Science, May 1990. Also to appear in *CONCUR*, 1990.

[13] A. Fekete, N. Lynch, and W. Weihl. A serialization graph construction for nested transactions. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 94–108, Nashville, TN, April 1990.

[14] A. Fekete, N. Lynch, M. Merritt, and W. Weihl. *Atomic transactions.* Book in progress.

[15] K. Goldman. Highly concurrent logically synchronous multicast. In *Proceedings of the Third International Workshop on Distributed Algorithms*, September 1989. Longer version as Technical Memo MIT/LCS/TM-401, MIT Laboratory for Computer Science, July 1989. Also submitted to *Distributed Computing.*

[16] K. Goldman. *Paralation Views: Abstractions for Efficient Scientific Computing on the Connection Machine.* Technical Memo MIT/LCS/TM-398, MIT Laboratory for Computer Science, August 1989.

[17] K. Goldman. Superposition in the I/O automaton model. In progress.

[18] K. Goldman and A. Lynch. Modelling shared state in a shared action model. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, June 1990. Also Technical Memo MIT/LCS/TM-427, MIT Laboratory for Computer Science, March 1990.

[19] K. Goldman and K. Yelick. Hierarchical correctness proofs for shared object systems. In progress.

[20] J. Halpern and M. Tuttle. Knowledge, probability, and adversaries. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, pages 103–118, August 1989. Also IBM Research Report RJ 7045, September 1989.

[21] M. Herlihy, N. Lynch, M. Merritt, and W. Weihl. *On the Correctness of Orphan Management Algorithms.* Technical Memo MIT/LCS/TM-406, MIT Laboratory for Computer Science, August 1989. Submitted for publication.

[22] N. Lynch and H. Attiya. Using mappings to prove timing properties. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, Quebec, Canada, August 1990. Expanded version: Technical Memo MIT/LCS/TM-412.b, MIT Laboratory for Computer Science, December 1989. Submitted for publication.

[23] N. Lynch. A hundred impossibility proofs for distributed computing. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, Edmonton, Alberta, Canada, August 1989. Also Technical Memo MIT/LCS/TR-394, MIT Laboratory for Computer Science, 1989.

[24] N. Lynch. Multivalued possibilities mappings. In *Rex Workshop*, Springer-Verlag LNCS 430, Mook, The Netherlands, May 1990. Also Technical Memo MIT/LCS/TM-422, MIT Laboratory for Computer Science, 1990.

[25] J. Welch and N. Lynch. *Synthesis of Efficient Drinking Philosophers Algorithms.* Technical Memo MIT/LCS/TM-417, MIT Laboratory for Computer Science, November 1989.

Theses in Progress

[1] K. Goldman. *Distributed Algorithm Simulation using Input/Output Automata.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, 1990. Supervised by N. Lynch.

[2] S. Ponzio. *Real-time Analysis of Timing-based Distributed Algorithms.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1990. Supervised by N. Lynch.

[3] I. Saias. *Time Analysis of Probalistic Algorithm.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, 1990. Supervised by N. Lynch.

[4] K. Streeter. *A Partitioned Computation Machine.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1990. Supervised by N. Lynch.

[5] G. Varghese. *Dealing with Failure in Distributed Systems.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, 1990. Supervised by N. Lynch.

Theses Completed

[1] M. Gupta. *I/O Automaton based Simulation of Selected Distributed Algorithms.* Bachelor's thesis. MIT Department of Electrical Engineering and Computer Science. June 1990. Supervised by N. Lynch.

[2] J. Leo. *Dynamic Process Creation in a Static Model.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[3] G. Troxel. *A Hierarchical Proof of an Algorithm for Deadlock Recovery in a System using Remote Procedure Calls.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, January 1990. Also Technical Report, MIT/LCS/TR-474, MIT Laboratory for Computer Science, 1990. Supervised by N. Lynch.

[4] M. Tuttle. *Knowledge and Distributed Computation.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, September 1989. Also Technical Report MIT/LCS/TR-477, MIT Laboratory for Computer Science, 1990 Supervised by N. Lynch.

Talks

[1] H. Attiya. Bounded polynomial randomized consensus. Lecture given at IBM Almaden Research Center, San Jose, CA, August 1989.

[2] H. Attiya. Bounded polynomial randomized consensus. Lecture given at the Eighth ACM PODC, Edmonton, Canada, August 1989.

[3] H. Attiya. Time bounds for real-time process control in the presence of timing uncertainty. Lecture given at ONR workshop, October 1989.

[4] H. Attiya. Time bounds for real-time process control in the presence of timing uncertainty. Lecture given at University of North Carolina, Chapel Hill, NC, October 1989.

[5] H. Attiya. Time bounds for real-time process control in the presence of timing uncertainty. Lecture given at Tenth RTSS, Santa-Monica, December 1989.

[6] H. Attiya. Timing properties of distributed systems in the presence of timing uncertainty. Lecture given at Georgia Institute of Technology, Atlanta, GA, February 1990.

[7] H. Attiya. Timing properties of distributed systems in the presence of timing uncertainty. Lecture given at Harvard University, March 1990.

[8] C. Dwork. Secret message transmission. September 1989.

[9] C. Dwork. Secret message transmission. Lecture given at Yale University, December 1989.

[10] C. Dwork. Secret message transmission. Lecture given at IBM Thomas J. Watson Research Center, March 1990.

[11] C. Dwork. Secret message transmission. Lecture given at Northeastern University, June 1990.

[12] K. Goldman. Highly concurrent logically synchronous multicast. Lecture given at the Third International Workshop on Distributed Algorithms, Nice, France, September 1989.

[13] K. Goldman. Simulation and animation of distributed algorithms using the input/output automaton model. Lecture given at IBM, Yorktown Heights, NY, September 1989.

[14] K. Goldman. The spectrum simulation system: a formal approach to distributed algorithm development. Lecture given at GTE Labs, Waltham, February 1990.

[15] K. Goldman. The spectrum simulation system: a formal approach to distributed algorithm development. Lecture given at Washington University, St. Louis, MO, February 1990.

[16] N. Lynch. Multivalued possibilities mappings. Lecture given at REX Workshop on Using Abstraction Mappings for Program Verification. Mook, The Netherlands, May 1989.

[17] N. Lynch. Using mappings to prove timing properties. Lecture given at the Second ONR Workshop on Real-Time Computing, October 1989.

[18] N. Lynch. Three impossibility results for distributed computing. Lecture given at University of Virginia, October 1989.

[19] N. Lynch. New results in the theory of real-time computing. Lecture given at IBM, Hawthorne, NY, January, 1990.

[20] N. Lynch. New results in the theory of real-time computing. Lecture given at University of Pennsylvania, April 1990.

[21] N. Lynch. New results in the theory of real-time computing. Lecture given at University of California, San Diego, CA, January 1990.

[22] N. Lynch. New results in the theory of real-time computing. Lecture given at Bellcore, NJ, March 1990.

[23] M. Tuttle. Knowledge, probability, and adversaries. Lecture given at the Eighth ACM PODC, August 1989.

X Consortium

Research Staff

D. Converse	C. Peterson
J. Fulton	B. Scheifler, Group Leader
K. Packard	

Graduate Students

J. Chen C. Lindblad

Undergraduate Students

K. Bergenthal D. Matic

Support Staff

M. Leger

15.1 Introduction

The MIT X Consortium was formed in January 1988 to further the development of the X Window System. The major goal of the Consortium is to promote cooperation within the computer industry in the creation of standard software interfaces at all layers in the X Window System environment. MIT's role is to provide the vendor-neutral architectural and administrative leadership required to make this work. The Consortium is financially self-supporting, with membership open to any organization. At present, nearly 70 companies belong to the Consortium, as well as several universities. These members represent the bulk of the US computer industry, as well as a considerable segment of the international industry.

15.2 Release 4 of the X Window System

One of the primary tasks of the Consortium staff is the maintenance and evolution of a software distribution containing sample implementations of all interfaces defined by the Consortium, as well as numerous applications and utilities. In January 1990, Release 4 of this distribution, consisting of 50 megabytes of source code and documentation, was made available to the world, along with a companion collection of 90 megabytes of code and documentation of user-contributed software. The distribution is available using anonymous FTP from numerous Internet sites, and on magnetic tape from the MIT Software Center. Some of the major improvements in Release 4:

Keith Packard and Bob Scheifler implemented a significantly faster and smaller X server; more detail is provided in the next section.

Full font support was added for the X Consortium standard X Logical Font Description conventions, and a fairly rich set of fonts was added for both 75 and 100 dots per inch displays. Font donations came from Adobe Systems, Digital Equipment Corporation, Bigelow and Holmes, Sun Microsystems, and Sony Corporation.

A major revision of the Xt Intrinsics has consolidated several independent extensions undertaken by industry groups in support of product-quality toolkits and modern graphical user interfaces. The most significant addition is support for windowless widgets (called "gadgets"), as well as other resource-based non-windowed objects for general programming. In addition, varargs-style interfaces, better caching of resources, support for incremental selections, improved error reporting, support for passive device grabs, and a class extension mechanism were added.

Chris Peterson completely redesigned the Text widget in the Athena Widget Set using the new object mechanisms in the Xt Intrinsics, providing a clean interface between the text source and sink, and significantly improving the internal text and resource management. New functionality was added to provide previously missing features, most notably search and replace. Chris reimplemented the Simple Menu widget to use the new object mechanisms, providing a much simpler programming interface, as well as reducing the amount and complexity of the code. Several of our applications have been converted to use the new menu facilities. Chris also completely rewrote the reference manual for the Athena Widgets, making it much easier to use and making it reflect the true interface provided by the toolkit.

Ralph Swick of Project Athena and Keith Packard incorporated support for non-rectangular windows into the Athena Widget Set and several applications, using the new X Consortium standard SHAPE extension. Round clocks and oval buttons are two pleasant outcomes of this work.

Jim Fulton rewrote most of the "twm" window manager (the program that people use to manipulate windows on the screen), providing X with the first public user interface that implemented the guidelines described in the X Consortium standard Inter-client Communication Conventions Manual. The window manager was also revised to support non-rectangular application windows, "tab"-style title bars, and non-rectangular icons, all using the new SHAPE extension.

Keith Packard did a major overhaul of the "xdm" display manager daemon, cutting the number of processes used in half, improving the robustness and, most importantly, implementing the X Consortium standard X Display Manager Control Protocol (XDMCP). XDMCP is designed to make X terminals as easy to use as traditional character cell terminals, and to provide a vendor-neutral mechanism for reducing the administrative overhead involved in managing a large network of X terminals connected to central compute servers.

Jim Fulton enhanced the "xterm" terminal emulator to support 8-bit characters, allowing the full ISO Latin-1 character set to be used. Chris Peterson similarly enhanced the Athena Text Widget. Bob Scheifler added simple bilingual keyboard support to the Xlib and Xt Intrinsics libraries.

Donna Converse substantially reworked the user interface to the "xmh" mail handler program. Visually, the interface is simpler, and it is highly configurable. Functionally, xmh is now more powerful, making it easier for us to process many electronic mail messages each day. The new xmh makes use of nearly every widget in the Athena Widget Set and adheres to the Inter-client Communication Conventions.

Chris Peterson made the "xman" manual page browser considerably easier to use, by making use of the new menus, adding keyboard accelerators, and providing a simple search facility.

Donna Converse also completely reimplemented the interface to the "xcalc" calculator program, making it a model demonstration of the power of application default resource files in allowing end-user customization.

Keith Packard implemented an "xditview" application for displaying ditroff DVI files on an X display.

Bob Scheifler improved the "xwud" image display program to work with various visual types and with standard colormaps, and implemented a simple grayscale conversion algorithm for displaying color images on a monochrome screen.

Jim Fulton integrated client-side support for System V Release 3.2. This allows X applications to be run on personal computers running the System V operating system, and also allows X applications to be run on Cray supercomputers.

Donna Converse and Ralph Swick added support for using ANSI C function prototypes, and made the major C header files usable from C++ as well. Donna fixed the Xlib implementation to deal gracefully when memory allocation fails.

15.3 X Server Optimizations

Work on the X sample server over the past year has focused on performance issues. Bob Scheifler and Keith Packard completed work on new data structures for the major server resources (principally windows, regions, and graphics contexts), resulting in one-half to two-thirds reduction in total data space in a typical running server. This is an extremely important gain for low-end X terminals with limited memory.

Keith Packard and Bob Scheifler, along with Joel McCormack of Digital Equipment Corporation, collaborated in designing and implementing new algorithms for manipulating the window hierarchy. Common window operations, such as create, map, unmap, move, and resize were all sped up, by factors of between two and twenty. Data structure redesign contributed to the performance increases for window creation and destruction; windows were redesigned so that instead of being composed of many small, independently allocated pieces, a single allocation could be used to allocate the entire contents of the most common form of window.

Keith Packard rewrote most of the device-dependent code for 8-bit color frame buffers, resulting in dramatic performance improvements (up to two orders of magnitude, in some cases) for points, lines, filled areas, text, area copies, and scrolling. Bob Scheifler reimplemented zero-width arcs and filled arcs, using integer algorithms that run up to 500 times faster than the previous algorithms. Keith and Bob derived an efficient exact integer algorithm for scan-converting wide lines with correct pixelization (previous algorithms were sometimes incorrect); Keith's implementation resulted in an order of magnitude speedup.

One of the performance problems not solved by Release 4 was efficiently dealing (in both time and space) with all combinations of 16 logical raster operations, in conjunction with arbitrary plane masks. Either many copies of each piece of code must be compiled (wasteful in space) or a runtime switch on the operation must be made for each pixel (wasteful in time). Since Release 4, Keith Packard devised and implemented a raster operation reduction scheme, which converts an (operation, plane mask) pair into the following boolean equation:

$$(\text{dst } \textit{and } (\text{src } \textit{and } \text{a1 } \textit{xor } \text{x1}) \textit{ and } (\text{src } \textit{and } \text{a2 } \textit{xor } \text{x2}))$$

where a1, a2, x1, x2, and m are constants. Although this equation may look complex, it is substantially faster than switching on the operation for each pixel. This equation can also be reduced further in many common cases. For example, when the source is a constant, it reduces to:

$$\text{dst } \textit{and } \text{av } \textit{xor } \text{xv}$$

The general equation can be reduced for the most common cases, and these can be compiled individually. For modern RISC processors, it is usually possible to perform several register operations between memory writes without slowing down, so these algorithms can often still run at memory speeds.

15.4 Internationalization

Internationalizing a program means making it adaptable to the requirements of different native languages, local customs, and coded character sets, so that the program can be run in different locales without source code modification or recompilation. Typically, the program will contain some collection of data (e.g., various strings) which need to change for each locale; this data usually must be external to the executable, so that it can be tailored to and dynamically selected for the desired locale.

One of the major areas of work within the X Consortium over the past year has been internationalization. This work has focused on changes to Xlib and the Xt Intrinsics to support internationalization. The work divides into four main topics: keyboard input, text display, text interchange, and resource files. A high priority for the work is to keep the interfaces and specifications as simple as possible, so that they can be understood and used by the general programming population, not just multi-lingual programmers.

Although many people believe that X should be designed to support true multi-lingual environments (supporting multiple languages, mixed into a single textual context), it is also a goal of our work to harmonize with existing formal standards for internationalization. The most important standard at this time is the ANSI C standard, and its locale mechanism. Unfortunately, this mechanism is quite biased towards a mono-lingual environment, with a single, global locale affecting all internationalized operations.

For keyboard input, the most important piece of functionality is supporting input methods, where multiple keystrokes are used to produce a single character or string of characters. An example of a very simple input method is the use of diacritical marks; typically a user will type a base letter and then a diacritical mark (or vice versa) to produce a single character. Much more complicated input methods are used in Asia. For example, in Japan it is common to type in Romaji (composed of Latin letters) which is converted on the fly to Kana characters and displayed; once a complete word (or phrase or sentence) is entered, a conversion key is pressed and conversion to Kanji takes place. There may be several Kanji words for a given Kana representation, and the user may have to choose from a list of alternatives. This conversion process typically requires fairly complex linguistic mechanisms, and large dictionaries.

All of this "pre-editing" normally should be hidden entirely from the application, since it is rather complex and locale-specific. However, this desire conflicts with another desire—that of providing a smooth integration of pre-edit with normal text editing. While pre-edit is in progress, there are two basic choices for where to display the pre-edit text: directly inline with the normal text, called "on the spot," with things like justification (in a WYSIWYG editor) taking place on the fly, and "off the spot," with the pre-edit text displayed somewhere below or to the side of the main text window. On the spot pre-editing is generally the most desirable for end users, but this requires tight coupling between the input method and that part of the application which is concerned with the actual display of the text. Hence, some support for pre-edit must be provided by the application in order to support on the spot pre-edit, although we do not want to require all applications to provide such support.

There are two basic models for implementing input methods: frontend and backend. In the frontend model, the input method is actually a separate process from the application. While pre-edit is in progress, keystrokes go to the input method rather than to the application; once a complete string of characters has been converted, the resulting string is then sent by the input method to the application. In the backend method, the input method is generally linked into the application as a library facility. While pre-edit is in progress, keystrokes continue to go to the application, which passes them on to the input method for processing.

The frontend model has a number of advantages. For example, it is easy to select an input method at runtime. This can be important in Asia, because there are generally a number of different input styles in use for a given language, even within a single organization. It is also possible to share a single input method process among all applications on the display. This is important, since the input method typically has very large resident dictionaries, as well as user-specific dictionaries that can be edited dynamically in one window and automatically propagate to other windows. But the frontend model has disadvantages as well. For example, it is rather expensive to implement on the spot pre-editing in this model, since it requires inter-process communication with the application on every keystroke. There are also significant event synchronization issues that are rather difficult to solve, such as coordinating changes to the input focus. Both frontend and backend models are being used in Japan, and a goal of our keyboard input design is to support both models, transparent to the application.

The principal issue in displaying text is mapping from the string encoding used in the application to the glyph encoding used for fonts. These encodings easily can be different, particularly when the string encoding contains a mixture of single-byte and multi-byte characters, or state-dependent control sequences. The goal for internationalized text display routines is to insulate the application from the complexities of this mapping. As part of this mapping, it may be necessary to use more than one font to render a given language. For example, the set of characters used in Taiwan is often viewed as up to sixteen "planes," each arranged as a two dimensional array of characters, each plane containing several thousand characters. At most, two planes can be represented with a single X font, so multiple fonts must be used to display a full set of such Chinese characters. To deal with this, the notion of a "font group" is introduced, which internally masks the details of the number of fonts used and the manner in which they are used.

In addition to the encoding issue, there are several other difficult issues for text display. In some instances, it may be necessary to use multiple glyphs to represent a single character. A simple example would be to use a Latin font containing only base letters and individual diacritical marks; a given character would then be displayed by using the glyph for a base letter, "overstruck" by the appropriate diacritical mark. In some languages with script letter forms, glyphs representing fragments of letters may be used to compose sequences of characters with appropriate tie marks. Another problem is "nonlinear" display, in which the order of glyphs on the screen does not match the order of characters in a string. A major example is text with both right-to-left and left-to-right text sequences, as occurs in Arabic and Hebrew. Another example would be the treatment of vowels in Hebrew and various Arabic derivatives. A goal of the internationalization work is to mask these details from the application. However, some of them are rather difficult issues, and it remains to be seen how well this can work.

Text interchange is a somewhat easier problem to deal with. The X Consortium standard Compound Text can be used as an interchange format for a large number of common languages. However, Compound Text only provides the raw character information, and does not provide any indication of locale; this information is generally needed to process or display the information. Unfortunately, there are no standards for locale names or for the locale mechanism defined by ANSI C, and no organized registration mechanism in place to collect such locale names and attempt to avoid conflicting use of names. This makes true interchange in a heterogeneous network environment somewhat problematic.

When an X program has been internationalized, it will generally use resource files to obtain locale-specific data, such as text labels and other visual cues. Simple mechanisms are required to enable the programmer to select a resource file dynamically, based on the locale, and resource databases must support locale-specific string encodings.

15.5 Resource Management

As we gain experience with sophisticated toolkits and applications, several problems with the X Resource Manager facilities have come to light. Chris Peterson has been exploring these problems and developing possible solutions. Four problems being considered are: easier means for users to override application default resources; support for multiple-value resources; conditional resource matching based on dynamic attributes; and interactive resource editing.

The current set of matching rules does not take into account the fact that the resource database created for an application may have been written by many different people. The database created for a toolkit application is loaded from several sources, with parts supplied by the application programmer and parts supplied by the end user. The application programmer often wants to be very specific with the resource definitions (e.g., a particular button should have a border width of 3). Users, on the other hand, often like to be as general as possible (e.g., all windows should be green). When these two specifications overlap the application default setting usually dominates since the more specific resource string will match.

A possible solution is to allow the resource database to be divided into sections. In essence, there would be several smaller databases that are searched in order, and if a match is found for a resource in an earlier section, entries in later sections would be ignored. Typically there would be two sections; application default resources would normally be loaded into the last section, and users could ensure an override by placing resources into the first section.

Another class of problem is exemplified by the translation table resources in the Xt Intrinsics. A text widget typically has a default set of translations built in to the widget. The application programmer often wants to modify a few of these translations, based on specific use of the widget within the application. Finally, the end user typically wants to make a few more modifications, to rebind commands to their liking. Unfortunately, given the entire-value replacement semantics of the resource manager, the application programmer's modifications will be lost when the end user tries to make modifications; the only way around this is for the end users to be aware of the application modifications and to incorporate them directly as part of their own modifications. This is very undesirable, because later changes by the application programmer will not propagate automatically.

A possible solution is allow a resource query to return a list of values, rather than just a single value. A new resource specification could be introduced to indicate that it augments a base resource specification, rather than replacing the base. A new lookup function could be provided to return the base value plus all augmentations. The translation table parsing in the Intrinsics could then be modified to use this new function, and use the resulting list of values to produce the desired composite translations.

A growing problem for both developers and users is the inability to specify that a given resource should be used only on a specific screen, or with a specific visual. For example, if a server supports two screens, one color and one monochrome, there is no reasonable mechanism for users to choose different colors depending on which screen an application is on. For developers, there is no way to make such choices within application default resource files, so applications are typically configured by default to only use black and white, even when they are displayed on a color screen.

A possible solution is to add a preprocessor-style language to the resource specification syntax, one that allows the application to make conditional comparisons to check attributes dynamically, at resource lookup time. Simply preprocessing the resource file when it is loaded is not an adequate solution, since the attributes to be matched may vary within an application as different objects search for their resources. For example, a given application might place windows on more than one screen of a given server. When querying a resource from the database, the application would supply a set of attributes, and these would be used to evaluate the conditional expressions to produce a (logically) modified instance of the database to resolve the query against. Typical attributes might be which screen, the visual class of the window, the number of bits per pixel being displayed, the particular host where the application is executing, and the particular server where the application is displaying.

A final problem that many users experience is mapping between printed documentation about resources for an application and the actual visual pieces of the application affected by those resources. Often, it would be easier to work in an inverse manner, namely pointing at an object on the screen and asking what resources are associated with it. Chris Peterson has been prototyping a possible extension to the Xt Intrinsics to support this, in combination with a resource editor program. The user can click on any application window, and the editor will graphically display a tree showing the widgets used by the application. The user can pan over this tree, select individual widgets, and dynamically modify their resources. The user can also select a node in the tree, and have the corresponding widget in the actual application highlighted. At present, to learn the complete set of resources supported by a widget, a companion application written by Jim Fulton can be used. This program graphically displays a tree showing the class hierarchy of the Athena Widgets. Individual classes can be selected, and their resources can be displayed. Eventually, the resource editor will be enhanced to query the application directly for the set of supported resources.

15.6 User Interface Monitoring

Good graphical user interfaces are difficult and time consuming to create. An interactive design process is often desirable, involving the construction and evaluation of prototypes.

Evaluation requires an effective means of recording and analyzing interactions between the user and the application.

Jolly Chen examined how monitoring mechanisms can be added to a user interface architecture to provide intrinsic support for recording the human-computer dialogue. Previous approaches to monitoring have generally captured information either at a very high level, requiring modification of the application, or at a very low level, recording keystrokes and other raw device events. Both of these levels have significant drawbacks, either requiring significant modification to the application or requiring an extremely detailed knowledge of the application in order to map low level events into meaningful semantic actions.

The key to intrinsic monitoring is inspection of the communication channels between the application objects and the interaction objects within a program. The information communicated at this level does not capture the full semantic intent of the user, but it does offer a higher level view of the user's actions than that offered by recording raw device events. For example, instead of recording only that a mouse button was pressed at location (100, 300), it is possible to record that the second item on a particular menu was selected. If the user interface architecture satisfies a few straightforward requirements, intrinsic monitoring can be easily added, without requiring support from the application programmer.

Jolly implemented such a monitoring mechanism with a few small changes to the Xt Intrinsics. The Xt Intrinsics architecture provides two main communication channels between the application and interaction objects: callbacks and actions. Both of these channels are easy to monitor. Actions can also be used for communication within and between interaction objects, but this level of detail is often of interest to the user interface evaluator as well, particularly when evaluating the design of new interaction objects. Using the standard object naming conventions of the Intrinsics, resources can be defined to enable and disable recording of individual callbacks and actions, or groups of them.

The monitoring mechanism is unobtrusive with respect to the user interface, and does not appear to impose a significant performance penalty. Jolly also implemented an analysis tool to process the resulting data. The tool is essentially a database application with a graphical user interface, designed specifically for storing and querying the monitored incidents. The tool provides various means for filtering, sorting, and summing the incidents. For example, one can query for all incidents that occurred in a Text widget and had a duration of greater than half a second, or count the number of sequences of three incidents that begin with a Help action and end with an Abort action.

15.7 Test Suite

The X Testing Consortium was a loosely bound group of approximately one dozen companies, working together on comprehensive test software for the X protocol and the Xlib C language interface to it. The Testing Consortium produced an Alpha Release of the test suite in August 1989, its last official act. It is generally agreed that the test suite is still a long way from being complete, and requires a fairly careful review before substantial new work is performed. Bob Scheifler put together a Request For Proposals for further development of the test suite. Nine bids were received, and they were reviewed by a multi-vendor committee headed by Bob Scheifler. A single firm has been chosen as subcontractor on the work, which is expected to take two years to complete, although several releases of the suite will be produced in the interim for evaluation by the X Consortium. A major goal of the work is to make the suite usable for regression testing, for validation of systems claimed to conform to Federal Information Processing Standard (FIPS) 158 on the X Window System, and for use by industry organizations in branding systems as compliant with their standards.

15.8 X Conference

In January 1990, we hosted the Fourth Annual X Technical Conference. The purpose of the conference is to present and discuss leading edge research and development in the X environment from both academia and industry. Having outgrown MIT facilities, this year the conference was held at the Boston Marriott Copley Place. The conference consisted of seven tutorials, 26 talks, and 23 informal "birds of a feather" sessions, spread over three days. Major themes were object-oriented toolkits and user interface management systems, X server performance issues, multithreaded clients and servers, input synthesis for regression testing, and novel window managers. Bob Scheifler and Donna Converse handled most of the details for the technical program. Donna Converse and Michelle Leger handled the bulk of the organizational details, including scheduling, catering, conference proceedings, and video tape coordination. MIT Conference Services handled registration. The conference was attended by approximately 1200 people, and was very well received.

15.9 Publications

[1] *Bitmap Distribution Format, Version 2.1.*

[2] J. Chen. *Providing Intrinsic Support for User Interface Monitoring.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1990.

[3] J. Flowers. *X Logical Font Description Conventions, Version 1.3.*

[4] *Fourth Annual X Technical Conference.* January 1990, video tapes.

[5] J. Friedberg, L. Seiler, and J. Vroom. *Extending X for Double-buffering, Multi-buffering, and Stereo, Version 3.2.*

[6] J. Gettys, R. W. Scheifler, and R. Newman. *Xlib - C Language X Interface, X Version 11, Release 4.*

[7] J. McCormack, P. Asents, and R. R.Swick. *X Toolkit Intrinsics - C Language Interface, X Version 11, Release 4.*

[8] K. Packard. *X Display Manager Control Protocol, Version 1.0.*

[9] K. Packard. *X11 Nonrectangular Window Shape Extension, Version 1.0.*

[10] M. Patrick and G. Sachs. *X11 Input Extension Library Specification, Public Review Draft.*

[11] C. Peterson. *Athena Widget Set - C Language Interface, X Version 11, Release 4.*

[12] D.S.H. Rosenthal. *Inter-client Communication Conventions Manual, Version 1.0.*

[13] R.J. Rost, editor. *PEX Protocol Specification, Version 4.0P.*

[14] G. Sachs and M. Patrick. *X11 Input Extension Protocol Specification, Public Review Draft.*

[15] R.W. Scheifler. *X Window System Protocol, X Version 11, Release 4.*

[16] R.W. Scheifler. *Compound Text Encoding, Version 1.1.*

Publications

Technical Memos[1]

TM-10 Jackson, J.N.
Interactive Design Coordination for the Building Industry, June 1970; AD 708400

TM-11 Ward, P.W.
Description and Flow Chart of the PDP-7/9 Communications Package, July 1970; AD 711379

TM-12 Graham, R.M.
File Management and Related Topics, June 12, 1970, September 1970; AD 712068

TM-13 Graham, R.M.
Use of High Level Languages for Systems Programming, September 1970; AD 711965

TM-14 Vogt, C.M.
Suspension of Processes in a Multi-processing Computer System, September 1970; AD 713989

TM-15 Zilles, S.N.
An Expansion of the Data Structuring Capabilities of PAL, October 1970; AD 720761

TM-16 Bruere-Dawson, G.
Pseudo-Random Sequences, October 1970; AD 713852

TM-17 Goodman, L.I.
Complexity Measures for Programming Languages, September 1971; AD 729011

TM-18 Reprinted as TR-85

TM-19 Fenichel, R.R.
A New List-Tracing Algorithm, October 1970; AD 714522

TM-20 Jones, T.L.
A Computer Model of Simple Forms of Learning, January 1971; AD 720337

TM-21 Goldstein, R.C.
The Substantive Use of Computers For Intellectual Activities, April 1971; AD 721618

TM-22 Wells, D.M.
Transmission of Information Between A Man-Machine Decision System and its Environment, April 1971; AD 722837

TM-23 Strnad , A.J.
The Relational Approach to the Management of Data Bases, April 1971; AD 721619

TM-24 Goldstein, R.C. and Strnad, A.J.
The MacAIMS Data Management System, April 1971; AD 721620

[1]TMs 1-9 were never issued.

TM-25 Goldstein, R.C.
Helping People Think, April 1971; AD 721998

TM-26 Iazeolla, G.G.
Modeling and Decomposition of Information Systems for Performance Evaluation, June 1971; AD 733965

TM-27 Bagchi, A.
Economy of Descriptions and Minimal Indices, January 1972, AD 736960

TM-28 Wong, R.
Construction Heuristics for Geometry and a Vector Algebra Representation of Geometry, June 1972, AD 743487

TM-29 Hossley, R. and Rackoff, C.
The Emptiness Problem for Automata on Infinite Trees, June 1972; AD 747250

TM-30 McCray, W.M.
SIM360: A S/360 Simulator, October 1972; AD 749365

TM-31 Bonneau. R.J.
A Class of Finite Computation Structures Supporting the Fast Fourier Transform, March 1973; AD 757787

TM-32 Moll, R.
An Operator Embedding Theorem for Complexity Classes of Recursive Functions, May 1973; AD 759999

TM-33 Ferrante, J. and Rackoff, C.
A Decision Procedure for the First Order Theory of Real Addition with Order, May 1973; AD 760000

TM-34 Bonneau, R.J.
Polynomial Exponentiation: The Fast Fourier Transform Revisited, June 1973; PB 221742

TM-35 Bonneau, R.J.
An Interactive Implementation of the Todd-Coxeter Algorithm, December 1973; AD 770565

TM-36 Geiger, S.P.
A User's Guide to the Macro Control Language, December 1973; AD 771435

TM-37 Schonhage, A.
Real-Time Simulation of Multidimensional Turing Machines by Storage Modification Machines, December 1973; PB 226103/AS

TM-38 Meyer, A.R.
Weak Monadic Second Order Theory of Successor Is Not Elementary-Recursive, December 1973; PB 226514/AS

TM-39 Meyer, A.R.
Discrete Computation: Theory and Open Problems, January 1974; PB 226836/AS

TM-40 Paterson, M.S., Fischer, M.J. and Meyer, A.R.
An Improved Overlap Argument for On-Line Multiplication, January 1974; AD 773137

TM-41 Fischer, M.J. and Paterson, M.S.
String-Matching and Other Products, January 1974; AD 773138

TM-42 Rackoff, C.
On the Complexity of the Theories of Weak Direct Products, January 1974; PB 228459/AS

TM-43 Fischer, M.J. and Rabin, M.O.
Super-Exponential Complexity of Presburger Arithmetic, February 1974; AD 775004

TM-44 Pless, V.
Symmetry Codes and their Invariant Subcodes, May 1974; AD 780243

TM-45 Fischer, M.J. and Stockmeyer, L.J.
Fast On-Line Integer Multiplication, May 1974; AD 779889

TM-46 Kedem, Z.M.
Combining Dimensionality and Rate of Growth Arguments for Establishing Lower Bounds on the Number of Multiplications, June 1974; PB 232969/AS

TM-47 Pless, V.
Mathematical Foundations of Flip-Flops, June 1974; AD 780901

TM-48 Kedem, Z.M.
The Reduction Method for Establishing Lower Bounds on the Number of Additions, June 1974; PB 233538/AS

TM-49 Pless, V.
Complete Classification of (24,12) and (22,11) Self-Dual Codes, June 1974; AD 781335

TM-50 Benedict, G.G.
An Enciphering Module for Multics, S.B. Thesis, EE Department, July 1974; AD 782658

TM-51 Aiello, J.M.
An Investigation of Current Language Support for the Data Requirements of Structured Programming, S.M. & E.E. Thesis, EE Department, September 1974; PB 236815/AS

TM-52 Lind, J.C.
Computing in Logarithmic Space, September 1974; PB 236167/AS

TM-53 Bengelloun, S.A.
MDC-Programmer: A Muddle-to-Datalanguage Translator for Information Retrieval, S.B. Thesis, EE Department, October 1974; AD 786754

TM-54 Meyer, A.R.
The Inherent Computation Complexity of Theories of Ordered Sets: A Brief Survey, October 1974; PB 237200/AS

TM-55 Hsieh, W.N., Harper, L.H. and Savage, J.E.
A Class of Boolean Functions with Linear Combinatorial Complexity, October 1974; PB 237206/AS

TM-56 Gorry, G.A.
Research on Expert Systems, December 1974

TM-57 Levin, M.
On Bateson's Logical Levels of Learning, February 1975

TM-58 Qualitz, J.E.
Decidability of Equivalence for a Class of Data Flow Schemas, March 1975; PB 237033/AS

TM-59 Hack, M.
Decision Problems for Petri Nets and Vector Addition Systems, March 1975; PB 231916/AS

TM-60 Weiss, R.B.
CAMAC: Group Manipulation System, March 1975; PB 240495/AS

TM-61 Dennis, J.B.
First Version of a Data Flow Procedure Language, May 1975

TM-62 Patil, S.S.
An Asynchronous Logic Array, May 1975

TM-63 Pless, V.
Encryption Schemes for Computer Confidentiality, May 1975; A010217

TM-64 Weiss, R.B.
Finding Isomorph Classes for Combinatorial Structures, S.M. Thesis, EE Department, June 1975

TM-65 Fischer, M.J.
The Complexity Negation-Limited Networks - A Brief Survey, June 1975

TM-66 Leung, C.K.C.
Formal Properties of Well-Formed Data, June 1975.

TM-67 Cardoza, E.E.
Computational Complexity of the World Problem for Commutative Semigroups, S.M. Thesis, EE & CS Department, October 1975

TM-68 Weng, K-S.
Stream-Oriented Computation in Recursive Data Flow Schemas, S.M. Thesis, EE & CS Department, October 1975

TM-69 Bayer, P.J.
Improved Bounds on the Costs of Optimal and Balanced Binary Search Trees, S.M. Thesis, EE & CS Department, November 1975

TM-70 Ruth, G.R.
Automatic Design of Data Processing Systems, February 1976; A023451

TM-71 Rivest, R.
On the Worst-Case of Behavior of String-Searching Algorithms, April 1976

TM-72 Ruth, G.R.
Protosystem I: An Automatic Programming System Prototype, July 1976; A026912

TM-73 Rivest, R.
Optimal Arrangement of Keys in a Hash Table, July 1976

TM-74 Malvania, N.
The Design of a Modular Laboratory for Control Robotics, S.M. Thesis, EE & CS Department, September 1976; A030418

TM-75 Yao, A.C. and Rivest, R.
K+1 Heads are Better than K, September 1976; A030008

TM-76 Bloniarz, P.A., Fischer, M.J. and Meyer, A.R.
A Note on the Average Time to Compute Transitive Closures, September 1976

TM-77 Mok, A.K.
Task Scheduling in the Control Robotics Environment, S.M. Thesis, EE & CS Department, September 1976; A030402

TM-78 Benjamin, A.J.
Improving Information Storage Reliability Using a Data Network, S.M. Thesis, EE & CS Department, October 1976; A033394

TM-79 Brown, G.P.
A System to Process Dialogue: A Progress Report, October 1976; A033276

TM-80 Even, S.
The Max Flow Algorithm of Dinic and Karzanov: An Exposition, December 1976

TM-81 Gifford, D.
Hardware Estimation of a Process' Primary Memory Requirements, S.B. Thesis, EE & CS Department, January 1977

TM-82 Rivest, R.L., Shamir, A. and Adelman, L.M.
A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, April 1977; A039036

TM-83 Baratz, A.E.
Construction and Analysis of Network Flow Problem which Forces Karzanov Algorithm to $O(n^3)$ Running Time, April 1977

TM-84 Rivest, R.L. and Pratt, V.R.
The Mutual Exclusion Problem for Unreliable Processes, April 1977

TM-85 Shamir, A.
Finding Minimum Cutsets in Reducible Graphs, June 1977; A040698

TM-86 Szolovits, P., Hawkinson, L.B. and Martin, W.A.
An Overview of OWL, A Language for Knowledge Representation, June 1977; A041372

TM-87 Clark, D., Editor
Ancillary Reports: Kernel Design Project, June 1977

TM-88 Lloyd, E.L.
On Triangulations of a Set of Points in the Plane, S.M. Thesis, EE & CS Department, July 1977

TM-89 Rodriguez, H.
Measuring User Characteristics on the Multics System, S.B. Thesis, EE & CS Department, August 1977

TM-90 d'Oliveira, C.R.
An Analysis of Computer Decentralization, S.B. Thesis, EE & CS Department, October 1977; A045526

TM-91 Shamir, A.
Factoring Numbers in $O(\log n)$ Arithmetic Steps, November 1977; A047709

TM-92 Misunas, D.P.
Report on the Workshop on Data Flow Computer and Program Organization, November 1977

TM-93 Amikura, K.
A Logic Design for the Cell Block of a Data-Flow Processor, S.M. Thesis, EE & CS Department, December 1977

TM-94 Berez, J.M.
A Dynamic Debugging System for MDL, S.B. Thesis, EE & CS Department, January 1978; A050191

TM-95 Harel, D.
Characterizing Second Order Logic With First Order Quantifiers, March 1977

TM-96 Harel, D., Pnueli, A. and Stavi, J.
A Complete Axiomatic System for Proving Deductions about Recursive Programs, February 1978

TM-97 Harel, D., Meyer, A.R. and Pratt, V.R.
Computability and Completeness in Logics of Programs, February 1978

TM-98 Harel, D. and Pratt, V.R.
Nondeterminism in Logics of Programs, February 1978

TM-99 LaPaugh, A.S.
The Subgraph Homeomorphism Problem, S.M. Thesis, EE & CS Department, February 1978

TM-100 Misunas, D.P.
A Computer Architecture for Data-Flow Computation, S.M. Thesis, EE & CS Department, March 1978; A052538

TM-101 Martin, W.A.
Descriptions and the Specialization of Concepts, March 1978; A052773

TM-102 Abelson, H.
Lower Bounds on Information Transfer in Distributed Computations, April 1978

TM-103 Harel, D.
Arithmetical Completeness in Logics of Programs, April 1978

TM-104 Jaffe, J.
The Use of Queues in the Parallel Data Flow Evaluation of "If-Then-While" Programs, May 1978

TM-105 Masek, W.J. and Paterson, M.S.
A Faster Algorithm Computing String Edit Distances, May 1978

TM-106 Parikh, R.
A Completeness Result for a Propositional Dynamic Logic, July 1978

TM-107 Shamir, A.
A Fast Signature Scheme, July 1978; A057152

TM-108 Baratz, A.E.
An Analysis of the Solovay and Strassen Test for Primality, July 1978

TM-109 Parikh, R.
Effectiveness, July 1978

TM-110 Jaffe, J.M.
An Analysis of Preemptive Multiprocessor Job Scheduling, September 1978

TM-111 Jaffe, J.M.
Bounds on the Scheduling of Typed Task Systems, September 1978

TM-112 Parikh, R.
A Decidability Result for a Second Order Process Logic, September 1978

TM-113 Pratt, V.R.
A Near-optimal Method for Reasoning about Action, September 1978

TM-114 Dennis, J.B., Fuller, S.H., Ackerman, W.B., Swan, R.J. and Weng, K-S.
Research Directions in Computer Architecture, September 1978; A061222

TM-115 Bryant, R.E. and Dennis, J.B.
Concurrent Programming, October 1978; A061180

TM-116 Pratt, V.R.
Applications of Modal Logic to Programming, December 1978

TM-117 Pratt, V.R.
Six Lectures on Dynamic Logic, December 1978

TM-118 Borkin, S.A.
Data Model Equivalence, December 1978; A062753

TM-119 Shamir, A. and Zippel, R.E.
On the Security of the Merkle-Hellman Cryptographic Scheme, December 1978; A063104

TM-120 Brock, J.D.
Operational Semantics of a Data Flow Language, S.M. Thesis, EE & CS Department, December 1978; A062997

TM-121 Jaffe, J.
The Equivalence of R.E. Programs and Data Flow Schemes, January 1979

TM-122 Jaffe, J.
Efficient Scheduling of Tasks Without Full Use of Processor Resources, January 1979

TM-123 Perry, H.M.
An Improved Proof of the Rabin-Hartmanis-Stearns Conjecture, S.M. & E.E. Thesis, EE & CS Department, January 1979

TM-124 Toffoli, T.
Bicontinuous Extensions of Invertible Combinatorial Functions, January 1979; A063886

TM-125 Shamir, A., Rivest, R.L. and Adelman, L.M.
Mental Poker, February 1979; A066331

TM-126 Meyer, A.R. and Paterson, M.S.
With What Frequency Are Apparently Intractable Problems Difficult?, February 1979

TM-127 Strazdas, R.J.
A Network Traffic Generator for DECNET, S.B. & S.M. Thesis, EE & CS Department, March 1979

TM-128 Loui, M.C.
Minimum Register Allocation is Complete in Polynomial Space, March 1979

TM-129 Shamir, A.
On the Cryptocomplexity of Knapsack Systems, April 1979; A067972

TM-130 Greif, I. and Meyer, A.R.
Specifying the Semantics of While-Programs: A Tutorial and Critique of a Paper by Hoare and Lauer, April 1979; A068967

TM-131 Adelman, L.M.
Time, Space and Randomness, April 1979

TM-132 Patil, R.S.
Design of a Program for Expert Diagnosis of Acid Base and Electrolyte Disturbances, May 1979

TM-133 Loui, M.C.
The Space Complexity of Two Pebble Games on Trees, May 1979

TM-134 Shamir, A.
How to Share a Secret, May 1979; A069397

TM-135 Wyleczuk, R.H.
Timestamps and Capability-Based Protection in a Distributed Computer Facility, S.B. & S.M. Thesis, EE & CS Department, June 1979

TM-136 Misunas, D.P.
Report on the Second Workshop on Data Flow Computer and Program Organization, June 1979

TM-137 Davis, E. and Jaffe, J.M.
Algorithms for Scheduling Tasks on Unrelated Processors, June 1979

TM-138 Pratt, V.R.
Dynamic Algebras: Examples, Constructions, Applications, July 1979

TM-139 Martin, W.A.
Roles, Co-Descriptors, and the Formal Representation of Quantified English Expressions, September 1979; A074625

TM-140 Szolovits, P.
Artificial Intelligence and Clinical Problem Solving, September 1979

TM-141 Hammer, M.M. and McLeod, D.
On Data Base Management System Architecture, October 1979; A076417

TM-142 Lipski, W.
On Data Bases with Incomplete Information, October 1979

TM-143 Leth, J.W.
An Intermediate Form for Data Flow Programs, S.M. Thesis, EE & CS Department, November 1979

TM-144 Takagi, A.
Concurrent and Reliable Updates of Distributed Databases, November 1979

TM-145 Loui, M.C.
A Space Bound for One-Tape Multidimensional Turing Machines, November 1979

TM-146 Aoki, D.J.
A Machine Language Instruction Set for a Data Flow Processor, S.M. Thesis, EE & CS Department, December 1979

TM-147 Schroeppel, R. and Shamir, A.
$AT + 0(2^{n/2}), S = 0(2^{n/4})$ Algorithm for Certain NP-Complete Problems, January 1980; A080385

TM-148 Adelman, L.M. and Loui, M.C.
Space-Bounded Simulation of Multitape Turing Machines, January 1980

TM-149 Pallottino, S. and Toffoli, T.
An Efficient Algorithm for Determining the Length of the Longest Dead Path in an "Lifo" Branchand-Bound Exploration Schema, January 1980; A079912

TM-150 Meyer, A.R.
Ten Thousand and One Logics of Programming, February 1980

TM-151 Toffoli, T.
Reversible Computing, February 1980; A082021

TM-152 Papadimitriou, C.H.
On the Complexity of Integer Programming, February 1980

TM-153 Papadimitriou, C.H.
Worst-Case and Probabilistic Analysis of a Geometric Location Problem, February 1980

TM-154 Karp, R.M. and Papadimitriou, C.H.
On Linear Characterizations of Combinatorial Optimization Problems, February 1980

TM-155 Itai, A., Lipton, R.J., Papadimitriou, C.H. and Rodeh, M.
Covering Graphs by Simple Circuits, February 1980

TM-156 Meyer, A.R. and Parikh, R.
Definability in Dynamic Logic, February 1980

TM-157 Meyer, A.R. and Winklmann, K.
On the Expressive Power of Dynamic Logic, February 1980

TM-158 Stark, E.W.
Semaphore Primitives and Starvation-Free Mutual Exclusion, S.M. Thesis, EE & CS Department, March 1980

TM-159 Pratt, V.R.
Dynamic Algebras and the Nature of Induction, March 1980

TM-160 Kanellakis, P.C.
On the Computational Complexity of Cardinality Constraints in Relational Databases March 1980

TM-161 Lloyd, E.L.
Critical Path Scheduling of Task Systems with Resource and Processor Constraints, March 1980

TM-162 Marcum, A.M.
A Manager for Named, Permanent Objects, S.B. & S.M. Thesis, EE & CS Department, April 1980; A083491

TM-163 Meyer, A.R. and Halpern, J.Y.
Axiomatic Definitions of Programming Languages: A Theoretical Assessment, April 1980

TM-164 Shamir, A.
The Cryptographic Security of Compact Knapsacks—Preliminary Report, April 1980; A084456

TM-165 Finseth, C.A.
Theory and Practice of Text Editors or A Cookbook for an Emacs, S.B. Thesis, EE & CS Department, May 1980

TM-166 Bryant, R.E.
Report on the Workshop on Self-Timed Systems, May 1980

TM-167 Pavelle, R. and Wester, M.
Computer Programs for Research in Gravitation and Differential Geometry, June 1980

TM-168 Greif, I.
Programs for Distributed Computing: The Calendar Application, July 1980; A087357

TM-169 Burke, G. and Moon, D.
LOOP Iteration Macro, July 1980; A087372

TM-170 Ehrenfeucht, A., Parikh, R. and Rozenberg, G.
Pumping Lemmas for Regular Sets, August 1980

TM-171 Meyer, A.R.
What is a Model of the Lambda Calculus?, August 1980

TM-172 Paseman, W.G.
Some New Methods of Music Synthesis, S.M. Thesis, EE & CS Department, August 1980; A090130

TM-173 Hawkinson, L.B.
XLMS: A Linguistic Memory System, September 1980; A090033

TM-174 Arvind, Kathail, V. and Pingali, K.
A Dataflow Architecture with Tagged Tokens, September 1980

TM-175 Meyer, A.R., Weise, D. and Loui, M.C.
On Time Versus Space III, September 1980

TM-176 Seaquist, C.R.
A Semantics of Synchronization, S.M. Thesis, EE & CS Department, September 1980; A091015

TM-177 Sinha, M.K.
TIMEPAD - A Performance Improving Synchronization Mechanism for Distributed Systems, September 1980

TM-178 Arvind and Thomas, R.E.
I-Structures: An Efficient Data Type for Functional Languages, September 1980

TM-179 Halpern, J.Y. and Meyer, A.R.
Axiomatic Definitions of Programming Languages, II, October 1980

TM-180 Papadimitriou, C.H.
A Theorem in Data Base Concurrency Control, October 1980

TM-181 Lipski, W. and Papadimitriou, C.H.
A Fast Algorithm for Testing for Safety and Detecting Deadlocks in Locked Transaction Systems, October 1980

TM-182 Itai, A., Papadimitriou, C.H. and Szwarefiter, J.L.
Hamilton Paths in Grid Graphs, October 1980

TM-183 Meyer, A.R.
A Note on the Length of Craig's Interpolants, October 1980

TM-184 Lieberman, H. and Hewitt, C.
A Real Time Garbage Collector that can Recover Temporary Storage Quickly, October 1980

TM-185 Kung, H-T. and Papadimitriou, C.H.
An Optimality Theory of Concurrency Control for Databases, November 1980; A092625

TM-186 Szolovits, P. and Martin, W.A.
BRAND X Manual, November 1980; A093041

TM-187 Fischer, M.J., Meyer, A.R. and Paterson, M.S.
$W(nlogn)$ Lower Bounds on Length of Boolean Formulas, November 1980

TM-188 Mayr, E.W.
An Effective Representation of the Reachability Set of Persistent Petri Nets, January 1981

TM-189 Mayr, E.W.
Persistence of Vector Replacement Systems is Decidable, January 1981

TM-190 Ben-Ari, M., Halpern, J.Y. and Pnueli, A.
Deterministic Propositional Dynamic Logic: Finite Models, Complexity, and Completeness, January 1981

TM-191 Parikh, R.
Propositional Dynamic Logics of Programs: A Survey, January 1981

TM-192 Meyer, A.R., Streett, R.S. and Mirkowska, G.
The Deducibility Problem in Propositional Dynamic Logic, February 1981

TM-193 Yannakakis, M. and Papadimitriou, C.H.
Algebraic Dependencies, February 1981

TM-194 Barendregt, H. and Longo, G.
Recursion Theoretic Operators and Morphisms on Numbered Sets, February 1981

TM-195 Barber, G.R.
Record of the Workshop on Research in Office Semantics, February 1981

TM-196 Bhatt, S.N.
On Concentration and Connection Networks, S.M. Thesis, EE & CS Department, March 1981

TM-197 Fredkin, E. and Toffoli, T.
Conservative Logic, May 1981

TM-198 Halpern, J.Y. and Reif, J.H.
The Propositional Dynamic Logic of Deterministic, Well-Structured Programs, March 1981

TM-199 Mayr, E.W. and Meyer, A.R.
The Complexity of the Word Problems for Commutative Semigroups and Polynomial Ideals, June 1981

TM-200 Burke, G.S.
LSB Manual, June 1981

TM-201 Meyer, A.R.
What is a Model of the Lambda Calculus? Expanded Version, July 1981

TM-202 Saltzer, J.H.
Communication Ring Initialization without Central Control, December 1981

TM-203 Bawden, A., Burke, G. and Hoffman, C.W.
MACLISP Extensions, July 1981

TM-204 Halpern, J.Y.
On the Expressive Power of Dynamic Logic, II, August 1981

TM-205 Kannon, R.
Circuit-Size Lower Bounds and Non-Reducibility to Sparce Sets, October 1981

TM-206 Leiserson, C.E. and Pinter, R.Y.
Optimal Placement for River Routing, October 1981

TM-207 Longo, G.
Power Set Models For Lambda-Calculus: Theories, Expansions, Isomorphisms, November 1981

TM-208 Cosmadakis, S. and Papadimitriou, C.H.
The Traveling Salesman Problem with Many Visits to Few Cities, November 1981

TM-209 Johnson, D. and Papadimitriou, C.H.
Computational Complexity and the Traveling Salesman Problem, December 1981

TM-210 Greif, I.
Software for the 'Roles' People Play, February 1982

TM-211 Meyer, A.R. and Tiuryn, J.
A Note on Equivalences Among Logics of Programs, December 1981

TM-212 Elias, P.
Minimax Optimal Universal Codeword Sets, January 1982

TM-213 Greif, I.
PCAL: A Personal Calendar, January 1982

TM-214 Meyer, A.R. and Mitchell, J.C.
Terminations for Recursive Programs: Completeness and Axiomatic Definability, March 1982

TM-215 Leiserson, C.E. and Saxe, J.B.
Optimizing Synchronous Systems, March 1982

TM-216 Church, K.W. and Patil, R.S.
Coping with Syntactic Ambiguity or How to Put the Block in the Box on the Table, April 1982

TM-217 Wright, K.D.
A File Transfer Program for a Personal Computer, April 1982

TM-218 Greif, I.
Cooperative Office Work, Teleconferencing and Calendar Management: A Collection of Papers, May 1982

TM-219 Jouannaud, J-P., Lescanne, P. and Reinig, F.
Recursive Decomposition Ordering and Multiset Orderings, June 1982

TM-220 Chu, T-A.
Circuit Analysis of Self-Timed Elements for NMOS VLSI Systems, May 1982

TM-221 Leighton, F.T., Lepley, M. and Miller, G.L.
Layouts for the Shuffle-Exchange Graph Based on the Complex Plane Diagram, June 1982

TM-222 Meier zu Sieker, F.
A Telex Gateway for the Internet, S.B. Thesis, Electrical Engineering Department, May 1982

TM-223 diSessa, A.A.
A Principled Design for an Integrated Computation Environment, July 1982

TM-224 Barber, G.
Supporting Organizational Problem Solving with a Workstation, July 1982

TM-225 Barber, G. and Hewitt, C.
Foundations for Office Semantics, July 1982

TM-226 Bergstra, J., Chmielinska, A. and Tiuryn, J.
Hoares' Logic Not Complete When it Could Be, August 1982

TM-227 Leighton, F.T.
New Lower Bound Techniques for VLSI, August 1982

TM-228 Papadimitriou, C.H. and Zachos, S.K.
Two Remarks on the Power of Counting, August 1982

TM-229 Cosmadakis, S.S.
The Complexity of Evaluation Relational Queries, August 1982

TM-230 Shamir, A.
Embedding Cryptographic Trapdoors in Arbitrary Knapsack Systems, September 1982

TM-231 Kleitman, D., Leighton, F.T., Lepley, M. and Miller G.L.
An Asymptotically Optimal Layout for the Shuffle-Exchange Graph, October 1982

TM-232 Yeh, A.
PLY: A System of Plausibility Inference with a Probabilistic Basis, December 1982

TM-233 Konopelski, L.J.
Implementing Internet Remote Login on a Personal Computer, S.B. Thesis, Electrical Engineering Department, December 1982

TM-234 Rivest, R. and Sherman, A.T.
Randomized Encryption Techniques, January 1983

TM-235 Mitchell, J.C.
The Implementation of Problem for Functional and Inclusion Dependencies, February 1983

TM-236 Leighton, F.T. and Leiserson, C.E.
Wafter-Scale Integration of Systolic Arrays, February 1983

TM-237 Dolev, D., Leighton, F.T. and Trickey, H.
Planar Embedding of Planar Graphs, February 1983

TM-238 Baker, B.S., Bhatt, S.N. and Leighton, F.T.
An Approximation Algorithm for Manhattan Routing, February 1983

TM-239 Sutherland, J.B. and Sirbu, M.
Evaluation of an Office Analysis Methodology, March 1983

TM-240 Bromley, H.
A Program for Therapy of Acid-Base and Electrolyte Disorders, S.B. Thesis, Electrical Engineering Department, June 1983

TM-241 Arvind and Iannucci, R.A.
Two Fundamental Issues in Multiprocessing: The Dataflow Solution, September 1983; A134239

TM-242 Pingali, K. and Arvind.
Efficient Demand-driven Evaluation (I), September 1983; A133477

TM-243 Pingali, K. and Arvind.
Efficient Demand-driven Evaluation (II), September 1983; A133879

TM-244 Goldreich, O., Goldwasser, S. and Micali, S.
How to Construct Random Functions, November 1983

TM-245 Meyer, A.R.
Understanding Algol: The View of the Recent Convert to Denotational Semantics, October 1983

TM-246 Trakhtenbrot, B.A., Halpern, J.Y. and Meyer, A.R.
From Denotational to Operational and Axiomatic Semantics for Algol-Like Languages: An Overview, October 1983

TM-247 Leighton, T. and Lepley, M.
Probabilistic Searching in Sorted Linked Lists, November 1983

TM-248 Leighton, F.T. and Rivest, R.L.
Estimating a Probability Using Finite Memory, November 1983

TM-249 Leighton, F.T. and Rivest, R.L.
The Markov Chain Tree Theorem, December 1983

TM-250 Goldreich, O.
On Concurrent Identification Protocols, December 1983

TM-251 Dolev, D., Lynch, N.A., Pinter, S. Stark, E. and Weihl, W.
Reaching Approximate Agreement in the Presence of Faults, December 1983

TM-252 Zachos, S. and Heller, H.
On BPP, December 1983

TM-253 Chor, B., Leiserson, C., Rivest, R. and Shearer, J.
An Application of Number Theory to the Organization of Raster Graphics Memory, April 1984; A140638

TM-254 Feldmeier, D.C.
Empirical Analysis of a Token Ring Network, January 1984; A138905

TM-255 Bhatt, S. and Leiserson, C.
How to Assemble Tree Machines, March 1984; A139963

TM-256 Goldreich, O.
On the Number of Close-and Equal Pairs of Bits in a String (With Implications on the Security of RSA's L.S.B.), March 1984

TM-257 Dwork, C., Kanellakis, P.C. and Mitchell, J.C.
On the Sequential Nature of Unification, March 1984; A139939

TM-258 Halpern, J.Y., Meyer A.R. and Trakhtenbrot, B.A.
The Semantics of Local Storage, or What Makes the Free-List Free?, April 1984

TM-259 Lynch, N.A. and Fredrickson, G.N.
The Impact of Synchronous Communication on the Problem of Selecting a Leader in a Ring, April 1984

TM-260 Chor, B. and Goldreich, O.
RSA/Rabin Least Significant Bits are 1/2 + 1/poly(logN) Secure, May 1984

TM-261 Zaks, S.
Optimal Distributed Algorithms for Sorting and Ranking, May 1984

TM-262 Leighton, T. and Rosenberg, A.
Three-dimensional Circuit Layouts, June 1984; A143430

TM-263 Sirbu, M.S. and Sutherland, J.B.
Naming and Directory Issues in Message Transfer Systems, July 1984; A154727

TM-264 Sarin, S.K. and Greif, I.
Software for Interactive On-Line Conferences, July 1984; A154742

TM-265 Lundelius, J. and Lynch, N.
A New Fault-Tolerant Algorithm for Clock Synchronization, July 1984; A154771

TM-266 Chor, B. and Coan, B.A.
A Simple and Efficient Randomized Byzantine Agreement Algorithm, August 1984; A153240

TM-267 Schooler, R. and Stamos, J.W.
Proposal for a Small Scheme Implementation, October 1984; A148707

TM-268 Awerbuch, B.
Complexity of Network Synchronization, January 1985

TM-269 Fischer, M., Lynch, N.A., Burns, J. and Borodin, A.
The Colored Ticket Algorithm, August 1983; A148696

TM-270 Dwork, C., Lynch, C. and Stockmeyer, L.
Consensus in the Presence of Partial Synchrony (Preliminary Version), July 1984; A154705

TM-271 Dershowitz, N. and Zaks, S.
Patterns in Trees, January 1985

TM-272 Leighton, T.
Tight Bounds on the Complexity of Parallel Sorting, April 1985; A154726

TM-273 Berman, F., Leighton, T., Shor, P.W. and Shor, L.
Generalized Planar Matching, April 1985; A154770

TM-274 Kuipers, B.
Qualitative Simulation of Mechanisms, April 1985

TM-275 Burns, J.E. and Lynch, N.A.
The Byzantine Firing Squad Problem, April 1985; A154809

TM276] Dolev, D., Lynch., N.A., Pinter, S.S., Stark, E.W. and Weihl, W.E. Reaching Approximate Agreement in the Presence of Faults, May 1985; A156541

TM-277 Frederickson, G.N. and Lynch, N.A.
A General Lower Bound for Electing a Leader in a Ring, March 1985; A156241

TM-278 Fisch, M.J., Griffeth, N.D., Guibas, L.J. and Lynch, N.A.
Probabilistic Analysis of a Network Resource Allocation Algorithm, June 1985; A157553

TM-279 Fischer, M.J., Lynch, N.A. and Merritt, M.
Easy Impossibility Proofs for Distributed Consensus Problems, June 1985; A157402

TM-280 Kuipers, B. and Kassirer, J.P.
Qualitative Simulation in Medical Physiology: A Progress Report, June 1985

TM-281 Hailperin, M.
What Price for Eliminating Expression Side-Effects?, June 1985

TM-282 Sarin, S. and Greif, I.
Computer Based Real-Time Conferences, July 1985

TM-283 Chor, B. and Goldreich, O.
Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity, September 1986

TM-284 Leiserson, C.E. and Saxe, J.B.
A Mixed-Integer Linear Programming Problems Which Is Efficiently Solvable, July 1985; A159496

TM-285 Kilian, J.J
Two Undecidability Results in Probabilistic Automata Theory, S.B. Thesis, EE & CS Department, June 1985

TM-286 Hastad, J.
Improvements of Yao's Results on Parity Circuits, September 1985

TM-287 Rivest, R.L.
Network Control by Bayesian Broadcast, July 1985

TM-288 Chung, J.C.
Dscribe: A Scribe Server, May 1985

TM-289 Toffoli, T. and Margolus, N.
The CAM-7 Multiprocessor: A Cellular Automata Machine, December 1985

TM-290 Fischer, M.J., Lynch, N.A., Burns, J.E. and Borodin, A.
Distributed FIFO Allocation of Identical Resources Using Small Shared Space, June 1985

TM-291 Goldberg, A.V.
A New Max-Flow Algorithm, November 1985

TM-292 Jain, R. and Routhier, S.
Packet Trains: Measurements and a New Model for Computer Network Traffic, November 1985

TM-293 Barrington, D.A.
Width-3 Permutation Branching Programs, December 1985

TM-294 Arvind and Culler, D.E.
Dataflow Architectures, February 1986; A166235

TM-295 Greif, I., Selinger, R. and Weihl, W.
Atomic Data Abstractions in a Distributed Collaborative Editing System (Extended Abstract), November 1985

TM-296 Margolus, N., Toffoli, T. and Vichniac, G.
Cellular Automata Supercomputers for Fluid Dynamics Modeling, December 1985

TM-297 Bentley, J.L., Leighton, F.T., Lepley, M., Stanat, D.F. and Steele, J.M.
A Randomized Data Structure for Ordered Sets, May 1986

TM-298 Leighton, T. and Shor, P.
Tight Bounds for Minimax Grid Matching, with Applications to the Average Case Analysis of Algorithms, May 1986

TM-299 Gifford, D.K., Lucassen, J.M. and Berlin, S.T,
The Application of Digital Broadcast Communication to Large Scale Information Systems, April 1986

TM-300 Dwork, C. and Moses, Y.
Knowledge and Common Knowledge in a Byzantine Environment: Crash Failures, July 1986

TM-301 Elias, P.
Interval and Regency-Rank Source Coding: Two On-Line Adaptive Variable-Length Schemes, April 1986

TM-302 Leighton, T. and Leiserson, C.E.
A Survey of Algorithms for Integrating Wafer-Scale Systolic Arrays, May 1986; A171623

TM-303 Li, M., Longpre, L. and Vitanyi, P.M.B.
The Power of the Queue, April 1986

TM-304 Kranakis, E. and Vitanyi, P.M.B.
Distributed Control in Computer Networks and Cross-Sections of Colored Multidimensional Bodies, April 1986; A172224

TM-305 Sacks, E.
Representing Change, May 1986

TM-306 Vitanyi, P.M.B.
Nonsequential Computation and Laws of Nature, May 1986; A171505

TM-307 Greenberg, R.I. and Leiserson, C.E.
Randomized Routing on Fat-Trees, April 1986

TM-308 Meyer, A.R.
Floyd-Hoare Logic Defines Semantics, May 1986

TM-309 Leiserson, C.E. and Saxe, J.B.
Retiming Synchronous Circuitry, May 1986; A172144

TM-310 Szolovits, P. Kassirer, J.P., Long, W.J., Moskowitz, A.J., Pauker, S.G., Patil, R.S. and Wellman, M.P.
An Artificial Intelligence Approach to Clinical Decision Making, September 1986

TM-311 Rivest, R.L.
Game Tree Searching By Min/Max Approximation, September 1986

TM-312 Sacks, E.
Hierarchical Inequality Reasoning, February 1987

TM-313 Theory of Computation Research Group
Theory of Computation Research Group Summary, 1985-86, August 1986

TM-314 Vitanyi, P.M.B. and Awerbuch, B.
Atomic Shared Register Access By Asynchronous Hardware (Detailed Abstract), October 1986; A178301

TM-315 Goldreich, O.
Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme, September 1986

TM-316 Greif, I. and Sarin, S.
Data Sharing in Group Work, October 1986

TM-317 Bennett, C.H., Toffoli, T. and Wolfram, S.
Cellular Automata '86 Conference, December 1986

TM-318 Leiserson, C.E. and Maggs, B.M.
Communication-Efficient Parallel Graph Algorithms, December 1986

TM-319 Leong, T-Y.
Murmur Clinic: An Auscultation Expert System, January 1987

TM-320 Goldberg, A.V. and Plotkin, S.A.
Efficient Parallel Algorithms for $(\Delta + 1)$-Coloring and Maximal Independent Set Problems, February 1987

TM-321 Cormen, T.H. and Leiserson, C.E.
A Hyperconcentrator Switch for Routing Bit-Serial Messages, February 1987; A17840

TM-322 Cormen, T.H.
Efficient Multichip Partial Concentrator Switches, February 1987; A178334

TM-323 Leiserson, C.E. and Phillips, C.A.
A Space-Efficient Algorithm for Finding the Connected Components of Rectangles in the Plane, February 1987

TM-324 Fekete, A. Lynch, N., Merritt, M. and Weihl, W.
Nested Transactions and Read/Write Locking, April 1987; A191981

TM-325 Awerbuch, B., Goldreich, O. and Vainish, R.
On the Message Complexity of Broadcast: A Basic Lower Bound (Extended Abstract), May 1987

TM-326 Awerbuch, B.
Controlling Worst-Case Performance of a Communication Protocol and Dynamic Resource Management, May 1987

TM-327 Awerbuch, B.
Adapting Communication Protocols to Dynamic Input and Network Topology: Research Summary, May 1987

TM-328 Awerbuch, B. and Plotkin, S.A.
Approximating the Size of a Dynamically Growing Asynchronous Distributed Network, April 1987

TM-329 Herlihy, M., Lynch, N., Merritt, M. and Weihl, W.
On the Correctness of Orphan Elimination Algorithms; May 1987; A182175

TM-330 Arvind and Iannucci, R.A.
Two Fundamental Issues in Multiprocessing (replaces TM-241), October 1987

TM-331 Russ, T.A.
Temporal Control Structure Reference Manual, June 1987

TM-332 Wellman, M.P.
Formulation of Tradeoffs in Planning Under Uncertainty, June 1987

TM-333 Goldberg, A.V. and Tarjan, R.E.
Finding Minimum-Cost Circulations by Successive Approximation, July 1987

TM-334 Goldberg, A.V. and Tarjan, R.E.
Finding Minimum-Cost Circulations by Canceling Negative Cycles, July 1987

TM-335 Bruce, K.B. and Riecke, J.G.
The Semantics of Miranda's Algebraic Types, August 1987

TM-336 Beame, P.
Lower Bounds for Recognizing Small Cliques on CROW PRAMS, August 1987

TM-337 Theory of Computation Research Group
Theory of Computation Group Research Summary June 1986-87, August 1987

TM-338 Fujita, T.
Multithreaded Processor Architecture for Parallel Symbolic Computation, September 1987

TM-339 Quinlan, J.R. and Rivest, R.L.
Inferring Decision Trees Using the Minimum Description Length Principle, September 1987

TM-340 Fekete, A., Lynch, N., Merritt, M. and Weihl, W.
Nested Transactions, Conflict-Based Locking and Dynamic Atomicity, September 1987

TM-341 Fekete, A., Lynch, N. and Shrira, L.
A Modular Proof of Correctness for a Network Synchronizer, September 1987

TM-342 Fekete, A.
Approximate Agreement, September 1987

TM-343 Leiserson, C.E. and Saxe, J.B.
A Mixed-Integer Linear Programming Problem, October 1987

TM-344 Meyer, A.R. and Sieber, K.
Towards Fully Abstract Semantics for Local Variables: Preliminary Report, November 1987

TM-345 Bloom, B., Istrail, S. and Meyer, A.R.
Bisimulation Can't Be Traced: Preliminary Report, November 1987

TM-346 Awerbuch, B.
Space Efficient Dynamic Protocols: Part I, December 1987

TM-347 Awerbuch, B.
Space Efficient Dynamic Protocols: Part II, December 1987

TM-348 Haimowitz, I.J.
Using NIKL in Large Medical Knowledge Base, December 1987

TM-349 Lynch, N.A. and Stark, E.W.
A proof of the Kahn principle for Input/Output Automata, January 1988

TM-350 Awerbuch, B.
Linear Time Algorithm for Minimum Network Partition, March 1988

TM-351 Lynch, N.
I/O Automata: A Model for Discrete Event Systems, March 1988

TM-352 Feldmeier, D.C.
Estimated Performance of a Gateway Routing Table Cache, March 1988

TM-353 Meyer, A.R.
Semantical Paradigms: Notes for an Invited Lecture, July 1988

TM-354 Awerbuch, B., Bar-Noy, A., Linial, N., and Peleg, D.
Improved Routing Strategies with Succinct Tables, April 1988

TM-355 Lynch, N., Mansour, Y. and Fekete, A.
The Data Link Layer: Two Impossibility Results, May 1988

TM-356 Goldberg, A.V., Plotkin, S.A. and Shannon, G.E.
Parallei Symmetry-Breaking in Sparse Graphs, May 1988

TM-357 Goldberg, A.V., Plotkin, S.A. and Vaidya, P.
Sublinear-Time Parallel Algorithms for Matching and Related Problems, July 1988

TM-358 Goldberg, A.V., Plotkin, S.A. and Tardos, E.
Combinatorial Algorithms for the Generalized Circulation Problem, May 1988

TM-359 Welch, J.L.
Simulating Synchronous Processors, June 1988

TM-360 Coan, B.A. and Welch, J.L.
Transaction Commit in a Realistic Timing Model, June 1988

TM-361 Welch, J.L., Lamport, L. and Lynch, N.
A Lattice-Structured Proof Technique Applied to a Minimum Spanning Tree Algorithm, June 1988

TM-362 Lynch, N., Merritt, M., Weihl, W.E. and Fekete, A.
A Theory of Atomic Transaction, June 1988

TM-363 Fogg, D.C.
Assisting Design Given Multiple Performance Criteria, August 1988

TM-364 Schaffer, R. and Bloom, B. (editor)
On the Correctness of Atomic Multi-Writer Registers, June 1988

TM-365 Awerbuch, B., Goldreich, O., Peleg, D. and Vainish, R.
A Tradeoff Between Information and Communication in Broadcast Protocols, July 1988

TM-366 Goldman, S.A.
A Space Efficient Greedy Triangulation Algorithm, July 1988

TM-367 Weihl, W.E.
Commutivity-Based Concurrency Control for Abstract Data Types, August 1988

TM-368 Herlihy, M.P. and Weihl, W.E.
Hybrid Concurrency Control For Abstract Data Types, August 1988

TM-369 Awerbuch, B., Bar-Noy, A., Linial, N. and Peleg, D.
Memory-Balanced Routing Strategies, August 1988

TM-370 Fekete, A.,Lynch, N., Merritt, M. and Weihl, B.
Commutativity-Based Locking for Nesting Transactions, August 1988

TM-371 Gifford, D.
Natural Random Numbers, September 1988

TM-372 Leiserson, C.E. and Saxe, J.B.
Retiming Synchronous Circuitry, October 1988

TM-373 Lynch, N.A. and Tuttle, M.R.
An Introduction to Input/Output Automata, November 1988

TM-374 Grigni, M. and Peleg, D.
Tight Bounds on Minimum Broadcast Network, October 1988

TM-375 Awerbuch, B.
On the Effects of Feedback in Dynamic Network Protocols, December 1988

TM-376 Awerbuch, B., Bar-Noy, A., Linial, N. and Peleg, D.
Improved Routing Strategies with Succinct Tables, December 1988

TM-377 TOC
Theory of Computation Group Research Summary June 1987-June 1988, January 1989

TM-378 Jouvelot, P. and Gifford, D.K.
Reasoning about Continuations with Control Effects, January 1989

TM-379 Kilian, J., Kipnis, S. and Leiserson, C.E.
The Organization of Permutation Architecture with Bussed Interconnections, January 1989

TM-380 O'Toole, J.W. and Gifford, D.K.
Type Reconstruction with First Class Polymorphic Values, May 1989

TM-381 Elias, P.
Error-Correcting Code for List Decoding, February 1989

TM-382 Weihl, W.
The Impact of Recovery on Concurrency Control, February 1989

TM-383 Wang, P.
A Special Case of Second-Order Strictness Analysis, February 1989

TM-384 Lamport, L. and Lynch, N.A.
Chapter on Distributed Computing, February 1989

TM-385 Sloan, R.
All Zero-Knowledge Proofs are Proofs of Language Membership, February 1989

TM-386 Jouvelot, P. and Gifford, D.K.
Communication Effects for Message-Based Concurrency, February 1989

TM-387 Doyle, J. and Patil, R.S.
Language Restrictions, Taxonomic Classification, and the Utility of Representation Services, May 1989

TM-388 Kipnis, S.
Three Methods for Range Queries in Computational Geometry, March 1989

TM-389 Afek, Y., Awerbuch, B. and Moriel, H.
A Complexity Preserving Reset Procedure, May 1989

TM-391 Awerbuch, B., Mansour, Y. and Shavit, N.
Polynomial End-to-End Communication, May 1989, A213 776

TM-392 Attiya, H., Dolev, D. and Shavit, N.
Bounded Polynomial Randomized Consensus, June 1989, A213 808

TM-393 Dolev, D. and Shavit, N.
Bounded Concurrent Time-Stamp Systems are Constructible, June 1989, A213 053

TM-394 Lynch, N.
A Hundred Impossibility Proofs for Distributed Computing, August 1989

TM-395 Owicki, S. and Anant, A.
Evaluating the Performance of Software Cache Coherence, June 1989

TM-396 Agarwal, A. and Cherian, M.
Adaptive Backoff Synchronization Techniques, June 1989, A213 966

TM-397 Agarwal, A. and Gupta, A.
Temporal, Processor, and Spatial Locality in Multiprocessor Memory References, June 1989, A213 790

TM-398 Goldman, K.J.
Paralation Views: Abstractions for Efficient Scientific Computing on the Connection Machine, June 1989

TM-399 Colby, C.P.
Correctness Proofs of the Peterson-Fischer Mutual Exclusion Algorithm, S.B. Thesis, June 1989

TM-400 Nour, M.F.
An Automata-Theoretic Model for Unity, S.B. Thesis, June 1989

TM-401 Goldman, K.J.
Highly Concurrent Logically Synchronous Multicast, July 1989; A213747

TM-402 Leiserson, C.E.
VLSI Theory and Parallel Supercomputing, May 1989; A214035

TM-403 Attiya, H. and Lynch, N. A.
Time Bounds for Real-time Process Control in the Presence of Timing Uncertainty, July 1989; A213791

TM-404 Theory of Computation Group Theory of Computation Group Research Summary—June 1988-July 1989, July 1989

TM-405 Boppana, R. B. and Sipser, M.
The Complexity of Finite Functions, August 1989

TM-406 Herlihy, M., Lynch, N., Merritt, M. and Weihl, W.
On the Correctness of Orphan Management Algorithms, August 1989; A213777

TM-408 Kipnis, S.
Priority Arbitration With Busses, October 1989

TM-409 Smith, M. A.
Nuclear Fusion Through Dimensional Confinement, August 1989

TM-410 Awerbuch, B. and Peleg, D.
Online Tracking of Mobile Users, August 1989

TM-411 Awerbuch, B. and Peleg, D.
Routing with Polynomial Communication-space Tradeoff, September 1989

TM-412 Lynch, N. and Attiya, H.
Using Mappings to Prove Timing Properties, December 1989; A213967

TM-412.b Lynch, N. and Hagit, A.
Using Mappings to Prove Timing Properties, December 1989

TM-413 Goldman, S., Rivest, R. and Schapire, R.
Learning Binary Relations and Total Orders, May 1990; A2221543

TM-415 Schapire, R.E.
The Strength of Weak Learnability, October 1989; A219843

TM-416 Doyle, J. and Wellman, M.
Impediments to Universal Preference-based Default Theories, October 1989

TM-417 Welch, J. and Lynch, N.
Synthesis of Efficient Drinking Philosophers Algorithms, November 1989

TM-418 Doyle, J. and Sacks, E.
Stochastic an Analysis of Qualitative Dynamics, December 1989

TM-419 Gifford, D.K.
Notes on Community Information Systems, December 1989

TM-421 Fekete, A., Lynch, N. and Weihl, W.
A Serialization Graph Construction for Nested Transactions, February 1990; A222697

TM-423 Attiya, H., Bar-Noy, A. and Dolev, D.
Sharing Memory Robustly in Message-passing Systems, February 1990

TM-424 Davis, R. and Swick, R. Workstation Services and Kerberos Authentication at Project Athena, February 1990

TM-425 Feldman, P. and Micali, S.
An Optimal Algorithm for Synchronous Byzantine Agreement, June 1990

TM-426 Awerbuch, B. and Peleg, D.
Non-obtrusive Synchronizersm, April 1990

TM-427 Goldman, K. and Lynch, N.A.
Modelling Shared State in a Shared Action Model, March 1990

TM-428 Fekete, A. and Lynch, N.A.
The Need for Headers: An Impossibility Result for Communication over Unreliable Channels, March 1990; A222823

TM-429 Afek, Y., Attiya, H., Dolve, D., Gafni, E., Merritt, M., and Shavit, N.
Atomic Snapshots of Shared Memory, May 1990; A222765

TM-430 Blum, M., De Santis, A., Micali, S. and Persiano, G.
Non-interactive Zero Knowledge, May 1990; A222698

TM-431 Schapire, R. E.
The Emerging Theory of Average-case Complexity, June 1990; A222821

TM-432 Bellare, M. and Micali, S.
How to Sign Given Any Trapdoor Permutation, June 1990

Technical Reports[2]

TR-1 Bobrow, D.G.
Natural Language Input for a Computer Problem Solving System, Ph.D. Dissertation, Math. Department, September 1964; AD 604730

TR-2 Raphael, B.
SIR: A Computer Program for Semantic Information Retrieval, Ph.D. Dissertation, Math. Department, June 1964; AD 608499

TR-3 Corbato, F.J.
System Requirements for Multiple-Access, Time-Shared Computers, May 1964; AD 608501

TR-4 Ross, D.T. and Feldman, C.G.
Verbal and Graphical Language for the AED System: A Progress Report, May 1964; AD 604678

TR-6 Biggs, J.M. and Logcher, R.D.
STRESS: A Problem-Oriented Language for Structural Engineering, May 1964; AD 604679

TR-7 Weizenbaum, J.
OPL-1: An Open Ended Programming System within CTSS, April 1964; AD 604680

TR-8 Greenberger, M.
The OPS-1 Manual, May 1964; AD 604681

TR-11 Dennis, J.B.
Program Structure in a Multi-Access Computer, May 1964; AD 604500

TR-12 Fano, R.M.
The MAC System: A Progress Report, October 1964; AD 609296

TR-13 Greenberger, M.
A New Methodology for Computer Simulation, October 1964; AD 609288

TR-14 Roos, D.
Use of CTSS in a Teaching Environment, November 1964; AD 661807

TR-16 Saltzer, J.H.
CTSS Technical Notes, March 1965; AD 612702

TR-17 Samuel, A.L.
Time-Sharing on a Multiconsole Computer, March 1965; AD 462158

TR-18 Scherr, A.L.
An Analysis of Time-Shared Computer Systems, Ph.D. Dissertation, EE Department, June 1965; AD 470715

[2]TRs 5, 9, 10, 15 were never issued

TR-19 Russo, F.J.
A Heuristic Approach to Alternate Routing in a Job Shop, S.B. & S.M. Thesis, Sloan School, June 1965; AD 474018

TR-20 Wantman, M.E.
CALCULAID: An On-Line System for Algebraic Computation and Analysis, S.M. Thesis, Sloan School, September 1965; AD 474019

TR-21 Denning, P.J.
Queueing Models for File Memory Operation, S.M. Thesis, EE Department, October 1965; AD 624943

TR-22 Greenberger, M.
The Priority Problem, November 1965; AD 625728

TR-23 Dennis, J.B. and Van Horn, E.C.
Programming Semantics for Multi-programmed Computations, December 1965; AD 627537

TR-24 Kaplow, R., Strong, S. and Brackett, J.
MAP: A System for On-Line Mathematical Analysis, January 1966; AD 476443

TR-25 Stratton, W.D.
Investigation of an Analog Technique to Decrease Pen-Tracking Time in Computer Display, S.M. Thesis, EE Department, March 1966; AD 631396

TR-26 Cheek, T.B.
Design of a Low-Cost Character Generator for Remote Computer Displays, S.M. Thesis, EE Department, March 1966; AD 631269

TR-27 Edwards, D.J.
OCAS - On-Line Cryptanalytic Aid System, S.M. Thesis, EE Department, May 1966; AD 633678

TR-28 Smith, A.A.
Input/Output in Time-Shared, Segmented, Multiprocessor Systems, S.M. Thesis, EE Department, June 1966; AD 637215

TR-29 Ivie, E.L.
Search Procedures Based on Measures of Relatedness between Documents, Ph.D. Dissertation, EE Department, June 1966; AD 636275

TR-30 Saltzer, J.H.
Traffic Control in a Multiplexed Computer System, Sc.D. Thesis, EE Department, July 1966; AD 635966

TR-31 Smith, D.L.
Models and Data Structures for Digital Logic Simulation, S.M. Thesis, EE Department, August 1966; AD 637192

TR-32 Teitelman, W.
PILOT: A Step Toward Man-Computer Symbiosis, Ph.D. Dissertation, Math. Department, September 1966; AD 638446

TR-33 Norton, L.M.
ADEPT - A Heuristic Program for Proving Theorems of Group Theory, Ph.D. Dissertation, Math. Department, October 1966; AD 645660

TR-34 Van Horn, E.C.
Computer Design for Asynchronously Reproducible Multiprocessing, Ph.D. Dissertation, EE Department, November 1966; AD 650407

TR-35 Fenichel, R.R.
An On-Line System for Algebraic Manipulation, Ph.D. Dissertation, Applied Mathematics (Harvard), December 1966; AD 657282

TR-36 Martin, W.A.
Symbolic Mathematical Laboratory, Ph.D. Dissertation, EE Department, January 1967; AD 657283

TR-37 Guzman-Arenas, A.
Some Aspects of Pattern Recognition by Computer, S.M. Thesis, EE Department, February 1967; AD 656041

TR-38 Rosenberg, R.C., Kennedy, D.W. and Humphrey, R.A.
A Low-Cost Output Terminal For Time-Shared Computers, March 1967; AD 662027

TR-39 Forte, A.
Syntax-Based Analytic Reading of Musical Scores, April 1967; AD 661806

TR-40 Miller, J.R.
On-Line Analysis for Social Scientists, May 1967; AD 668009

TR-41 Coons, S.A.
Surfaces for Computer-Aided Design of Space Forms, June 1967; AD 663504

TR-42 Liu, C.L., Chang, G.D. and Marks, R.E.
Design and Implementation of a Table-Driven Compiler System, July 1967; AD 668960

TR-43 Wilde, D.U.
Program Analysis by Digital Computer, Ph.D. Dissertation, EE Department, August 1967; AD 662224

TR-44 Gorry, G.A.
A System for Computer-Aided Diagnosis, Ph.D. Dissertation, Sloan School, September 1967; AD 662665

TR-45 Leal-Cantu, N.
On the Simulation of Dynamic Systems with Lumped Parameters and Time Delays, S.M. Thesis, ME Department, October 1967; AD 663502

TR-46 Alsop, J.W.
A Canonic Translator, S.B. Thesis, EE Department, November 1967; AD 663503

TR-47 Moses, J.
Symbolic Integration, Ph.D. Dissertation, Math. Department, December 1967; AD 662666

TR-48 Jones, M.M.
Incremental Simulation on a Time-Shared Computer, Ph.D. Dissertation, Sloan School, January 1968; AD 662225

TR-49 Luconi, F.L.
Asynchronous Computational Structures, Ph.D. Dissertation, EE Department, February 1968; AD 667602

TR-50 Denning, P.J.
Resource Allocation in Multiprocess Computer Systems, Ph.D. Dissertation, EE Department, May 1968; AD 675554

TR-51 Charniak, E.
CARPS, A Program which Solves Calculus Word Problems, S.M. Thesis, EE Department, July 1968; AD 673670

TR-52 Deitel, H.M.
Absentee Computations in a Multiple-Access Computer System, S.M. Thesis, EE Department, August 1968; AD 684738

TR-53 Slutz, D.R.
The Flow Graph Schemata Model of Parallel Computation, Ph.D. Dissertation, EE Department, September 1968; AD 683393

TR-54 Grochow, J.M.
The Graphic Display as an Aid in the Monitoring of a Time-Shared Computer System, S.M. Thesis, EE Department, October 1968; AD 689468

TR-55 Rappaport, R.L.
Implementing Multi-Process Primitives in a Multiplexed Computer System, S.M. Thesis, EE Department, November 1968; AD 689469

TR-56 Thornhill, D., Stotz, R.H., Ross, D.T. and Ward, J.E.
An Integrated Hardware-Software System for Computer Graphics in Time-Sharing, December 1968; AD 685202

TR-57 Morris, J.H.
Lambda Calculus Models of Programming Languages, Ph.D. Dissertation, Sloan School, December 1968; AD 683394

TR-58 Greenbaum, H.J.
A Simulator of Multiple Interactive Users to Drive a Time-Shared Computer System, S.M. Thesis, EE Department, January 1969; AD 686988

TR-59 Guzman-Arenas, A.
Computer Recognition of Three-Dimensional Objects in a Visual Scene, Ph.D. Dissertation, EE Department, December 1968; AD 692200

TR-60 Ledgard, H.F.
A Formal System for Defining the Syntax and Semantics of Computer Languages, Ph.D. Dissertation, EE Department, April 1969; AD 689305

TR-61 Baecker, R.M.
Interactive Computer-Mediated Animation, Ph.D. Dissertation, EE Department, June 1969; AD 690887

TR-62 Tillman, C.C.
EPS: An Interactive System for Solving Elliptic Boundary-Value Problems with Facilities for Data Manipulation and General-Purpose Computation, June 1969; AD 692462

TR-63 Brackett, J., Hammer, M.M. and Thornhill, D.
Case Study in Interactive Graphics Programming: A Circuit Drawing and Editing Program for Use with a Storage-Tube Display Terminal, October 1969; AD 699930

TR-64 Rodrigues, J.E.
A Graph Model for Parallel Computations, Sc.D. Thesis, EE Department, September 1969; AD 697759

TR-65 Deremer, F.L.
Practical Translators for LR(k) Languages, Ph.D. Dissertation, EE Department, October 1969; AD 699501

TR-66 Beyer, W.T.
Recognition of Topological Invariants by Iterative Arrays, Ph.D. Dissertation, Math. Department, October 1969; AD 699502

TR-67 Vanderbilt, D.H.
Controlled Information Sharing in a Computer Utility, Ph.D. Dissertation, EE Department, October 1969; AD 699503

TR-68 Selwyn, L.
Economies of Scale in Computer Use: Initial Tests and Implications for The Computer Utility, Ph.D. Dissertation, Sloan School, June 1970; AD 710001

TR-69 Gertz, J.L.
Hierarchical Associative Memories for Parallel Computation, Ph.D. Dissertation, EE Department, June 1970; AD 711091

TR-70 Fillat, A.I. and Kraning, L.A.
Generalized Organization of Large Data Bases: A Set-Theoretic Approach to Relations, S.B. & S.M. Thesis, EE Department, June 1970; AD 711060

TR-71 Fiasconaro, J.G.
A Computer-Controlled Graphical Display Processor, S.M. Thesis, EE Department, June 1970; AD 710479

TR-72 Patil, S.S.
Coordination of Asynchronous Events, Sc.D. Thesis, EE Department, June 1970; AD 711763

TR-73 Griffith, A.K.
Computer Recognition of Prismatic Solids, Ph.D. Dissertation, Math. Department, August 1970; AD 712069

TR-74 Edelberg, M.
Integral Convex Polyhedra and an Approach to Integralization, Ph.D. Dissertation, EE Department, August 1970; AD 712070

TR-75 Hebalkar, P.G.
Deadlock-Free Sharing of Resources in Asynchronous Systems, Sc.D. Thesis, EE Department, September 1970; AD 713139

TR-76 Winston, P.H.
Learning Structural Description ¿from Examples, Ph.D. Dissertation, EE Department, September 1970; AD 713988

TR-77 Haggerty, J.P.
Complexity Measures for Language Recognition by Canonic Systems, S.M. Thesis, EE Department, October 1970; AD 715134

TR-78 Madnick, S.E.
Design Strategies for File Systems, S.M. Thesis, EE Department & Sloan School, October 1970; AD 714269

TR-79 Horn, B.K.P.
Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View, Ph.D. Dissertation, EE Department, November 1970; AD 717336

TR-80 Clark, D.D., Graham, R.M., Saltzer, J.H. and Schroeder, M.D.
The Classroom Information and Computing Service, January 1971; AD 717857

TR-81 Banks, E.R.
Information Processing and Transmission in Cellular Automata, Ph.D. Dissertation, ME Department, January 1971; AD 717951 w

TR-82 Krakauer, L.J.
Computer Analysis of Visual Properties of Curved Objects, Ph.D. Dissertation, EE Department, May 1971; AD 723647

TR-83 Lewin, D.
In-Process Manufacturing Quality Control, Ph.D. Dissertation, Sloan School, January 1971; AD 720098

TR-84 Winograd, T.
Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, Ph.D. Dissertation, Math. Department, February 1971; AD 721399

TR-85 Miller, P.L.
Automatic Creation of a Code Generator ¿from a Machine Description, E.E. Thesis, EE Department, May 1971; AD 724730

TR-86 Schell, R.R.
Dynamic Reconfiguration in a Modular Computer System, Ph.D. Dissertation, EE Department, June 1971; AD 725859

TR-87 Thomas, R.H.
A Model for Process Representation and Synthesis, Ph.D. Dissertation, EE Department, June 1971; AD 726049

TR-88 Welch, T.A.
Bounds on Information Retrieval Efficiency in Static File Structures, Ph.D. Dissertation, EE Department, June 1971; AD 725429

TR-89 Owens, R.C.
Primary Access Control in Large-Scale Time-Shared Decision Systems, S.M. Thesis, Sloan School, July 1971; AD 728036

TR-90 Lester, B.P.
Cost Analysis of Debugging Systems, S.B. & S.M. Thesis, EE Department, September 1971; AD 730521

TR-91 Smoliar, S.W.
A Parallel Processing Model of Musical Structures, Ph.D. Dissertation, Math. Department, September 1971; AD 731690

TR-92 Wang, P.S.
Evaluation of Definite Integrals by Symbolic Manipulation, Ph.D. Dissertation, Math. Department, October 1971; AD 732005

TR-93 Greif, I.
Induction in Proofs about Programs, S.M. Thesis, EE Department, February 1972; AD 737701

TR-94 Hack, M.
Analysis of Production Schemata by Petri Nets, S.M. Thesis, EE Department, February 1972; AD 740320

TR-95 Fateman, R.J.
Essays in Algebraic Simplification (A revision of a Harvard Ph.D. Dissertation), April 1972; AD 740132

TR-96 Manning, F.
Autonomous, Synchronous Counters Constructed Only of J-K Flip-Flops, S.M. Thesis, EE Department, May 1972; AD 744030

TR-97 Vilfan, B.
The Complexity of Finite Functions, Ph.D. Dissertation, EE Department, March 1972; AD 739678

TR-98 Stockmeyer, L.J.
Bounds on Polynomial Evaluation Algorithms, S.M. Thesis, EE Department, April 1972; AD 740328

TR-99 Lynch, N.A.
Relativization of the Theory of Computational Complexity, Ph.D. Dissertation, Math. Department, June 1972; AD 744032

TR-100 Mandl, R.
Further Results on Hierarchies of Canonic Systems, S.M. Thesis, EE Department, June 1972; AD 744206

TR-101 Dennis, J.B.
On the Design and Specification of a Common Base Language, June 1972; AD 744207

TR-102 Hossley, R.F.
Finite Tree Automata and Ω-Automata, S.M. Thesis, EE Department, September 1972; AD 749367

TR-103 Sekino, A.
Performance Evaluation of Multiprogrammed Time-Shared Computer Systems, Ph.D Dissertation, EE Department, September 1972; AD 749949

TR-104 Schroeder, M.D.
Cooperation of Mutually Suspicious Subsystems in a Computer Utility, Ph.D. Dissertation, EE Department, September 1972; AD 750173

TR-105 Smith, B.J.
An Analysis of Sorting Networks, Sc.D. Thesis, EE Department, October 1972; AD 751614

TR-106 Rackoff, C.
The Emptiness and Complementation Problems for Automata on Infinite Trees, S.M. Thesis, EE Department, January 1973; AD 756248

TR-107 Madnick, S.E.
Storage Hierarchy Systems, Ph.D. Dissertation, EE Department, April 1973; AD 760001

TR-108 Wand, M.
Mathematical Foundations of Formal Language Theory, Ph.D. Dissertation, Math. Department, December 1973.

TR-109 Johnson, D.S.
Near-Optimal Bin Packing Algorithms, Ph.D Dissertation, Math. Department, June 1973, PB 222-090

TR-110 Moll, R.
Complexity Classes of Recursive Functions, Ph.D. Dissertation, Math. Department, June 1973; AD 767730

TR-111 Linderman, J.P.
Productivity in Parallel Computation Schemata, Ph.D. Dissertation, EE Department, December 1973, PB 226-159/AS

TR-112 Hawryszkiewycz, I.T.
Semantics of Data Base Systems, Ph.D. Dissertation, EE Department, December 1973, PB 226-061/AS

TR-113 Herrmann, P.P.
On Reducibility Among Combinatorial Problems, S.M. Thesis, Math. Department, December 1973, PB 226-157/AS

TR-114 Metcalfe, R.M.
Packet Communication, Ph.D. Dissertation, Applied Math., Harvard University, December 1973; AD 771430

TR-115 Rotenberg, L.
Making Computers Keep Secrets, Ph.D. Dissertation, EE Department, February 1974, PB 229-352/AS

TR-116 Stern, J.A.
Backup and Recovery of On-Line Information in a Computer Utility, S.M. & E.E. Thesis, EE Department, January 1974; AD 774141

TR-117 Clark, D.D.
An Input/Output Architecture for Virtual Memory Computer Systems, Ph.D. Dissertation, EE Department, January 1974; AD 774738

TR-118 Briabrin, V.
An Abstract Model of a Research Institute: Simple Automatic Programming Approach, March 1974, PB 231-505/AS

TR-119 Hammer, M.M.
A New Grammatical Transformation into Deterministic Top-Down Form, Ph.D. Dissertation, EE Department, February 1974; AD 775545

TR-120 Ramchandani, C.
Analysis of Asynchronous Concurrent Systems by Timed Petri Nets, Ph.D. Dissertation, EE Department, February 1974; AD 775618

TR-121 Yao, F.F.
On Lower Bounds for Selection Problems, Ph.D. Dissertation, Math. Department, March 1974, PB 230-950/AS

TR-122 Scherf, J.A.
Computer and Data Security: A Comprehensive Annotated Bibliography, S.M. Thesis, Sloan School, January 1974; AD 775546

TR-123 Saltzer, et al.
Introduction to Multics, February 1974; AD 918562

TR-124 Laventhal, M.S.
Verification of Programs Operating on Structured Data, S.B. & S.M. Thesis, EE Department, March 1974, PB 231-365/AS

TR-125 Mark, W.S.
A Model-Debugging System, S.B. & S.M. Thesis, EE Department, April 1974; AD 778688

TR-126 Altman, V.E.
A Language Implementation System, S.B. & S.M. Thesis, Sloan School, May 1974; AD 780672

TR-127 Greenberg, B.
An Experimental Analysis of Program Reference Patterns in the Multics Virtual Memory, S.M. Thesis, EE Department, May 1974; AD 780407

TR-128 Frankston, R.M.
The Computer Utility as a Marketplace for Computer Services, S.M. & E.E. Thesis, EE Department, May 1974; AD 780436

TR-129 Weissberg, R.
Using Interactive Graphics in Simulating the Hospital Emergency Room, S.M. Thesis, EE Department, May 1974; AD 780437

TR-130 Ruth, G.R.
Analysis of Algorithm Implementations, Ph.D. Dissertation, EE Department, May 1974; AD 780408

TR-131 Levin, M.
Mathematical Logic for Computer Scientists, June 1974.

TR-132 Janson, P.A.
Removing the Dynamic Linker from the Security Kernel of a Computing Utility, S.M. Thesis, EE Department, June 1974; AD 781305

TR-133 Stockmeyer, L.J.
The Complexity of Decision Problems in Automata Theory and Logic, Ph.D. Dissertation, EE Department, July 1974, PB 235-283/AS

TR-134 Ellis, D.J.
Semantics of Data Structures and References, S.M. & E.E. Thesis, EE Department, August 1974, PB 236-594/AS

TR-135 Pfister, G.F.
The Computer Control of Changing Pictures, Ph.D. Dissertation, EE Department, September 1974; AD 787795

TR-136 Ward, S.
Functional Domains of Applicative Languages, Ph.D. Dissertation, EE Department, September 1974; AD 787796

TR-137 Seiferas, J.
Nondeterministic Time and Space Complexity Classes, Ph.D. Dissertation, Math. Department, September 1974; PB 236777/AS

TR-138 Yun, David Y.Y.
The Hensel Lemma in Algebraic Manipulation, Ph.D. Dissertation, Math. Department, November 1974; AD 02737

TR-139 Ferrante, J.
Some Upper and Lower Bounds on Decision Procedures in Logic, Ph.D. Dissertation, Math. Department, November 1974. PB 238121/AS

TR-140 Redell, D.D.
Naming and Protection in Extendable Operating Systems, Ph.D. Dissertation, EE Department, November 1974; AD 01721

TR-141 Richards, M., Evans, A. Jr. and Mabee, R.F.
The BCPL Reference Manual, December 1974; AD 03599

TR-142 Brown, G.P.
Some Problems in German to English Machine Translation, S.M. & E.E. Thesis, EE Department, December 1974; AD 03002

TR-143 Silverman, H.
A Digitalis Therapy Advisor, S.M. Thesis, EE Department, January 1975.

TR-144 Rackoff, C.
The Computational Complexity of Some Logical Theories, Ph.D. Dissertation, EE&CS Department, February 1975.

TR-145 Henderson, D.A.
The Binding Model: A Semantic Base for Modular Programming Systems, Ph.D. Dissertation, EE&CS Department, February 1975; AD A006961

TR-146 Malhotra, A.
Design Criteria for a Knowledge-Based English Language System for Management: An Experimental Analysis, Ph.D. Dissertation, EE&CS Department, February 1975.

TR-147 Van De Vanter, M.L.
A Formalization and Correctness Proof of the CGOL Language System, S.M. Thesis, EE&CS Department, March 1975.

TR-148 Johnson, J.
Program Restructuring for Virtual Memory Systems, Ph.D. Dissertation, EE&CS Department, March 1975; AD 09218

TR-149 Snyder, A.
A Portable Compiler for the Language C, S.B. & S.M. Thesis, EE&CS Department, May 1975; AD 10218

TR-150 Rumbaugh, J.E.
A Parallel Asynchronous Computer Architecture for Data Flow Programs, Ph.D. Dissertation, EE&CS Department, May 1975; AD 10918

TR-151 Manning, F.
Automatic Test, Configuration and Repair of Cellular Arrays, Ph.D. Dissertation, EE&CS Department, June 1975; AD 12822

TR-152 Qualitz, J.E.
Equivalence Problems for Monadic Schemas, Ph.D. Dissertation, EE&CS Department, June 1975; AD 12823

TR-153 Miller, P.B.
Strategy Selection in Medical Diagnosis, S.M. Thesis, EE & CS Department, September 1975.

TR-154 Greif, I.
Semantics of Communicating Parallel Processes, Ph.D. Dissertation, EE & CS Department, September 1975; AD 16302

TR-155 Kahn, K.M.
Mechanization of Temporal Knowledge, S.M. Thesis, EE & CS Department, September 1975.

TR-156 Bratt, R.G.
Minimizing the Naming Facilities Requiring Protection in a Computing Utility, S.M. Thesis, EE & CS Department, September 1975.

TR-157 Meldman, J.A.
A Preliminary Study in Computer-Aided Legal Analysis, Ph.D. Dissertation, EE & CS Department, November 1975; AD 18997

TR-158 Grossman, R.W.
Some Data Base Applications of Constraint Expressions, S.M. Thesis, EE & CS Department, February 1976; AD 24149

TR-159 Hack, M.
Petri Net Languages, March 1976.

TR-160 Bosyj, M.
A Program for the Design of Procurement Systems, S.M. Thesis, EE & CS Department, May 1976; AD 26688

TR-161 Hack, M.
Decidability Questions for Petri Nets, Ph.D. Dissertation, EE & CS Department, June 1976.

TR-162 Kent, S.
Encryption-Based Protection Protocols for Interactive User-Computer Communication, S.M. Thesis, EE & CS Department, June 1976; AD 26911

TR-163 Montgomery, W.A.
A Secure and Flexible Model of Process Initiation for a Computer Utility, S.M. & E.E. Thesis, EE & CS Department, June 1976.

TR-164 Reed, D.P.
Processor Multiplexing in a Layered Operating System, S.M. Thesis, EE & CS Department, July 1976.

TR-165 McLeod, D.
High Level Expression of Semantic Integrity Specifications in a Relational Data Base System, S.M. Thesis, EE & CS Department, September 1976; AD 34184

TR-166 Chan, A.Y.
Index Selection in a Self-Adaptive Relational Data Base Management System, S.M. Thesis, EE & CS Department, September 1976; AD 34185

TR-167 Janson, P.A.
Using Type Extension to Organize Virtual Memory Mechanisms, Ph.D. Dissertation, EE & CS Department, September 1976.

TR-168 Pratt, V.R.
Semantical Considerations on Floyd-Hoare Logic, September 1976.

TR-169 Safran, C., Desforges, J.F. and Tsichlis, P.N.
Diagnostic Planning and Cancer Management, September 1976.

TR-170 Furtek, F.C.
The Logic of Systems, Ph.D. Dissertation, EE & CS Department, December 1976.

TR-171 Huber, A.H.
A Multi-Process Design of a Paging System, S.M. & E.E. Thesis, EE & CS Department, December 1976.

TR-172 Mark, W.S.
The Reformulation Model of Expertise, Ph.D. Dissertation, EE & CS Department, December 1976; AD 35397

TR-173 Goodman, N.
Coordination of Parallel Processes in the Actor Model of Computation, S.M. Thesis, EE & CS Department, December 1976.

TR-174 Hunt, D.H.
A Case Study of Intermodule Dependencies in a Virtual Memory Subsystem, S.M. & E.E. Thesis, EE & CS Department, December 1976.

TR-175 Goldberg, H.J.
A Robust Environment for Program Development, S.M. Thesis, EE & CS Department, February 1977.

TR-176 Swartout, W.R.
A Digitalis Therapy Advisor with Explanations, S.M. Thesis, EE & CS Department, February 1977.

TR-177 Mason, A.H.
A Layered Virtual Memory Manager, S.M. & E.E. Thesis, EE & CS Department, May 1977.

TR-178 Bishop, P.B.
Computer Systems with a Very Large Address Space and Garbage Collection, Ph.D. Dissertation, EE & CS Department, May 1977; AD 40601

TR-179 Karger, P.A.
Non-Discretionary Access Control for Decentralized Computing Systems, S.M. Thesis, EE & CS Department, May 1977; AD 40804

TR-180 Luniewski, A.
A Simple and Flexible System Initialization Mechanism, S.M. & E.E. Thesis, EE & CS Department, May 1977.

TR-181 Mayr, E.W.
The Complexity of the Finite Containment Problem for Petri Nets, S.M. Thesis, EE & CS Department, June 1977 .

TR-182 Brown, G.P.
A Framework for Processing Dialogue, June 1977; AD 42370

TR-183 Jaffe, J.M.
Semilinear Sets and Applications, S.M. Thesis, EE & CS Department, July 1977.

TR-184 Levine, P.H.
Facilitating Interprocess Communication in a Heterogeneous Network Environment, S.B. & S.M. Thesis, EE & CS Department, July 1977; AD 43901

TR-185 Goldman, B.
Deadlock Detection in Computer Networks, S.B. & S.M. Thesis, EE & CS Department, September 1977; AD 47025

TR-186 Ackerman, W.B.
A Structure Memory for Data Flow Computers, S.M. Thesis, EE & CS Department, August 1977; AD 47026

TR-187 Long, W.J.
A Program Writer, Ph.D. Dissertation, EE & CS Department, November 1977; AD 47595

TR-188 Bryant, R.E.
Simulation of Packet Communication Architecture Computer Systems, S.M. Thesis, EE & CS Department, November 1977; AD 48290

TR-189 Ellis, D.J.
Formal Specifications for Packet Communication Systems, Ph.D. Dissertation, EE & CS Department, November 1977; AD 48380

TR-190 Moss, E.B.
Abstract Data Types in Stack Based Languages, S.M. Thesis, EE & CS Department, February 1978; AD 52332

TR-191 Yonezawa, A.
Specification and Verification Techniques for Parallel Programs Based on Message Passing Semantics, Ph.D. Dissertation, EE & CS Department, January 1978; AD 51149

TR-192 Niamir, B.
Attribute Partitioning in a Self-Adaptive Relational Data Base System, S.M. Thesis, EE & CS Department, January 1978; AD 53292

TR-193 Schaffert, C.
A Formal Definition of CLU, S.M. Thesis, EE & CS Department, January 1978

TR-194 Hewitt, C. and Baker, H.G.
Actors and Continuous Functionals, February 1978; AD 52266

TR-195 Bruss, A.R.
On Time-Space Classes and Their Relation to the Theory of Real Addition, S.M. Thesis, EE & CS Department, March 1978

TR-196 Schroeder, M.D., Clark, D.D., Saltzer, J.H. and Wells, D.
Final Report of the Multics Kernel Design Project, March 1978

TR-197 Baker, H.G.
Actor Systems for Real-Time Computation, Ph.D. Dissertation, EE & CS Department, March 1978; AD 53328

TR-198 Halstead, R.H.
Multiple-Processor Implementations of Message-Passing Systems, S.M. Thesis, EE & CS Department, April 1978; AD 54009

TR-199 Terman, C.J.
The Specification of Code Generation Algorithms, S.M. Thesis, EE & CS Department, April 1978; AD 54301

TR-200 Harel, D.
Logics of Programs: Axiomatics and Descriptive Power, Ph.D. Dissertation, EE & CS Department, May 1978

TR-201 Scheifler, R.
A Denotational Semantics of CLU, S.M. Thesis, EE & CS Department, May 1978

TR-202 Principato, R.N.
A Formalization of the State Machine Specification Technique, S.M. & E.E. Thesis, EE & CS Department, July 1978

TR-203 Laventhal, M.S.
Synthesis of Synchronization Code for Data Abstractions, Ph.D. Dissertation, EE & CS Department, July 1978; AD 58232

TR-204 Teixeira, T.J.
Real-Time Control Structures for Block Diagram Schemata, S.M. Thesis, EE & CS Department, August 1978; AD 61122

TR-205 Reed, D.P.
Naming and Synchronization in a Decentralized Computer System, Ph.D. Dissertation, EE & CS Department, October 1978; AD 61407

TR-206 Borkin, S.A.
Equivalence Properties of Semantic Data Models for Data Base Systems, Ph.D. Dissertation, EE & CS Department, January 1979; AD 66386

TR-207 Montgomery, W.A.
Robust Concurrency Control for a Distributed Information System, Ph.D. Dissertation, EE & CS Department, January 1979; AD 66996

TR-208 Krizan, B.C.
A Minicomputer Network Simulation System, S.B. & S.M. Thesis, EE & CS Department, February 1979

TR-209 Snyder, A.
A Machine Architecture to Support an Object-Oriented Language, Ph.D. Dissertation, EE & CS Department, March 1979; AD 68111

TR-210 Papadimitriou, C.H.
Serializability of Concurrent Data Base Updates, March 1979

TR-211 Bloom, T.
Synchronization Mechanisms for Modular Programming Languages, S.M. Thesis, EE & CS Department, April 1979; AD 69819

TR-212 Rabin, M.O.
Digitalized Signatures and Public-Key Functions as Intractable as Factorization, January 1979

TR-213 Rabin, M.O.
Probabilistic Algorithms in Finite Fields, January 1979

TR-214 McLeod, D.
A Semantic Data Base Model and Its Associated Structured User Interface, Ph.D. Dissertation, EE & CS Department, March 1979; AD 68112

TR-215 Svobodova, L., Liskov, B. and Clark, D.D.
Distributed Computer Systems: Structure and Semantics, April 1979; AD 70286

TR-216 Myers, J.M.
Analysis of the SIMPLE Code for Data Flow Computation, May 1979

TR-217 Brown, D.
Storage and Access Costs for Implementations of Variable Length Lists, Ph.D. Dissertation, EE & CS Department, April 1979

TR-218 Ackerman, W.B. and Dennis, J.B.
VAL–A Value-Oriented Algorithmic Language, Preliminary Reference Manual, June 1979; AD 72394

TR-219 Sollins, K.R.
Copying Complex Structures in a Distributed System, S.M. Thesis, EE & CS Department, July 1979; AD 72441

TR-220 Kosinski, P.R.
Denotational Semantics of Determinate and Nondeterminate Data Flow Programs, Ph.D. Dissertation, EE & CS Department, July 1979

TR-221 Berzins, V.A.
Abstract Model Specifications for Data Abstractions, Ph.D. Dissertation, EE & CS Department, July 1979

TR-222 Halstead, R.H.
Reference Tree Networks: Virtual Machine and Implementation, Ph.D. Dissertation, EE & CS Department, September 1979; AD 76570

TR-223 Brown, G.P.
Toward a Computational Theory of Indirect Speech Acts, October 1979; AD 77065

TR-224 Isaman, D.L.
Data-Structuring Operations in Concurrent Computations, Ph.D. Dissertation, EE & CS Department, October 1979

TR-225 Liskov, B., Atkinson, R.R., Bloom, T., Moss, E.B., Schaffert, C., Scheifler, R. and Snyder, A.
CLU Reference Manual, October 1979; AD 77018

TR-226 Reuveni, A.
The Event Based Language and Its Multiple Processor Implementations, Ph.D. Dissertation, EE & CS Department, January 1980; AD 81950

TR-227 Rosenberg, R.L.
Incomprehensible Computer Systems: Knowledge Without Wisdom, S.M. Thesis, EE & CS Department, January 1980

TR-228 Weng, K.-S.
An Abstract Implementation for a Generalized Data Flow Language, Ph.D. Dissertation, EE & CS Department, January 1980

TR-229 Atkinson, R.R.
Automatic Verification of Serializes, Ph.D. Dissertation, EE & CS Department, March 1980; AD 82885

TR-230 Baratz, A.E.
The Complexity of the Maximum Network Flow Problem, S.M. Thesis, EE & CS Department, March 1980

TR-231 Jaffe, J.M.
Parallel Computation: Synchronization, Scheduling, and Schemes, Ph.D. Dissertation, EE & CS Department, March 1980

TR-232 Luniewski, A.
The Architecture of an Object Based Personal Computer, Ph.D. Dissertation, EE & CS Department, March 1980; AD 83433

TR-233 Kaiser, G.E.
Automatic Extension of an Augmented Transition Network Grammar for Morse Code Conversations, S.B. Thesis, EE & CS Department, April 1980; AD 84411

TR-234 Herlihy, M.P.
Transmitting Abstract Values in Messages, S.M. Thesis, EE & CS Department, May 1980; AD 86984

TR-235 Levin, L.A.
A Concept of Independence with Applications in Various Fields of Mathematics, May 1980

TR-236 Lloyd, E.L.
Scheduling Task Systems with Resources, Ph.D. Dissertation, EE & CS Department, May 1980

TR-237 Kapur, D.
Towards a Theory for Abstract Data Types, Ph.D. Dissertation, EE & CS Department, June 1980; AD 85877

TR-238 Bloniarz, P.A.
The Complexity of Monotone Boolean Functions and an Algorithm for Finding Shortest Paths on a Graph, Ph.D. Dissertation, EE & CS Department, June 1980

TR-239 Baker, C.M.
Artwork Analysis Tools for VLSI Circuits, S.M. & E.E. Thesis, EE & CS Department, June 1980; AD 87040

TR-240 Montz, L.B.
Safety and Optimization Transformations for Data Flow Programs, S.M. Thesis, EE & CS Department, July 1980

TR-241 Archer, R.F.
Representation and Analysis of Real-Time Control Structures, S.M. Thesis, EE & CS Department, August 1980; AD 89828

TR-242 Loui, M.C.
Simulations Among Multidimensional Turing Machines, Ph.D. Dissertation, EE & CS Department, August 1980

TR-243 Svobodova, L.
Management of Object Histories in the Swallow Repository, August 1980; AD 89836

TR-244 Ruth, G.R.
Data Driven Loops, August 1980

TR-245 Church, K.W.
On Memory Limitations in Natural Language Processing, S.M. Thesis, EE & CS Department, September 1980

TR-246 Tiuryn, J.
A Survey of the Logic of Effective Definitions, October 1980

TR-247 Weihl, W.E.
Interprocedural Data Flow Analysis in the Presence of Pointers, Procedure Variables, and Label Variables, S.B. & S.M. Thesis, EE & CS Department, October 1980

TR-248 LaPaugh, A.S.
Algorithms for Integrated Circuit Layout: An Analytic Approach, Ph.D. Dissertation, EE & CS Department, November 1980

TR-249 Turkle, S.
Computers and People: Personal Computation, December 1980

TR-250 Leung, C.K.C.
Fault Tolerance in Packet Communication Computer Architectures, Ph.D. Dissertation, EE & CS Department, December 1980

TR-251 Swartout, W.R.
Producing Explanations and Justifications of Expert Consulting Programs, Ph.D. Dissertation, EE & CS Department, January 1981

TR-252 Arens, G.C.
Recovery of the Swallow Repository, S.M. Thesis, EE & CS Department, January 1981; AD 96374

TR-253 Ilson, R.
An Integrated Approach to Formatted Document Production, S.M. Thesis, EE & CS Department, February 1981

TR-254 Ruth, G.R., Alter, S. and Martin, W.A.
A Very High Level Language for Business Data Processing, March 1981

TR-255 Kent, S.
Protecting Externally Supplied Software in Small Computers, Ph.D. Dissertation, EE & CS Department, March 1981

TR-256 Faust, G.G.
Semiautomatic Translation of COBOL into HIBOL, S.M. Thesis, EE & CS Department, April 1981

TR-257 Cesari, C.A.
Application of Data Flow Architecture to Computer Music Synthesis, S.B./S.M. Thesis, EE & CS Department, February 1981

TR-258 Singh, N.P.
A Design Methodology for Self-Timed Systems, S.M. Thesis, EE & CS Department, February 1981

TR-259 Bryant, R.E.
A Switch-Level Simulation Model for Integrated Logic Circuits, Ph.D. Dissertation, EE & CS Department, March 1981

TR-260 Moss, E.B.
Nested Transactions: An Approach to Reliable Distributed Computing, Ph.D. Dissertation, EE & CS Department, April 1981; A100754

TR-261 Martin, W.A., Church, K.W. and Patil, R.S.
Preliminary Analysis of a Breadth-First Parsing Algorithm: Theoretical and Experimental Results, EE & CS Department, June 1981

TR-262 Todd, K.W.
High Level VAL Constructs in a Static Data Flow Machine, S.M. Thesis, EE & CS Department, June 1981

TR-263 Street, R.S.
Propositional Dynamic Logic of Looping and Converse, Ph.D. Dissertation, EE & CS Department, May 1981

TR-264 Schiffenbauer, R.D.
Interactive Debugging in a Distributed Computational Environment, S.M. Thesis, EE & CS Department, August 1981

TR-265 Thomas, R.E.
A Data Flow Architecture with Improved Asymptotic Performance, Ph.D. Dissertation, EE & CS Department, April 1981

TR-266 Good, M.
An Ease of Use Evaluation of an Integrated Editor and Formatter, S.M. Thesis, EE & CS Department, August 1981

TR-267 Patil, R.S.
Causal Representation of Patient Illness for Electrolyte and Acid-Base Diagnosis, Ph.D. Dissertation, EE & CS Department, October 1981

TR-268 Guttag, J.V., Kapur, D., Musser, D.R.
Derived Pairs, Overlap Closures, and Rewrite Dominoes: New Tools for Analyzing Term Rewriting Systems, EE & CS Department, December 1981

TR-269 Kanellakis, P.C.
The Complexity of Concurrency Control for Distributed Data Bases, Ph.D. Dissertation, EE & CS Department, December 1981

TR-270 Singh, V.
The Design of a Routing Service for Campus-Wide Internet Transport, S.M. Thesis, EE & CS Department, January 1982

TR-271 Rutherford, C.J., Davies, B., Barnett, A.I. and Desforges, J.F.
A Computer System for Decision Analysis in Hodgkins Disease, EE & CS Department, February 1982

TR-272 Smith, B.C.
Reflection and Semantics in a Procedural Language, Ph.D. Dissertation, EE & CS Department, January 1982

TR-273 Estrin, D.L.
Data Communications via Cable Television Networks: Technical and Policy Considerations, S.M. Thesis, EE & CS Department, May 1982

TR-274 Leighton, F.T.
Layouts for the Shuffle-Exchange Graph and Lower Bound Techniques for VLSI, Ph.D. Dissertation, EE & CS Department, August 1982

TR-275 Kunin, J.S.
Analysis and Specification of Office Procedures, Ph.D. Dissertation, EE & CS Department, February 1982

TR-276 Srivas, M.K.
Automatic Synthesis of Implementations for Abstract Data Types from Algebraic Specifications, Ph.D. Dissertation, EE & CS Department, June 1982

TR-277 Johnson, M.G.
Efficient Modeling for Short Channel MOS Circuit Simulation, S.M. Thesis, EE & CS Department, August 1982

TR-278 Rosenstein, L.S.
Display Management in an Integrated Office, S.M. Thesis, EE & CS Department, January 1982

TR-279 Anderson, T.L.
The Design of a Multiprocessor Development System, S.M. Thesis, EE & CS Department, September 1982

TR-280 Guang-Rong, G.
An Implementation Scheme for Array Operations in Static Data Flow Computers, S.M. Thesis, EE & CS Department, May 1982

TR-281 Lynch, N.A.
Multilevel Atomicity - A New Correctness Criterion for Data Base Concurrency Control, EE & CS Department, August 1982

TR-282 Fischer, M.J., Lynch, N.A. and Paterson, M.S.
Impossibility of Distributed Consensus with One Faulty Process, EE & CS Department, September 1982

TR-283 Sherman, H.B.
A Comparative Study of Computer-Aided Clinical Diagnosis, S.M. Thesis, EE & CS Department, January 1981

TR-284 Cosmadakis, S.S.
Translating Updates of Relational Data Base Views, S.M. Thesis, EE & CS Department, February 1983

TR-285 Lynch, N.A.
Concurrency Control for Resilient Nested Transactions, EE & CS Department, February 1983

TR-286 Goree, J.A.
Internal Consistency of a Distributed Transaction System with Orphan Detection, S.M. Thesis, EE & CS Department, January 1983

TR-287 Bui, T.N.
On Bisecting Random Graphs, S.M. Thesis, EE & CS Department, March 1983

TR-288 Landau, S.E.
On Computing Galois Groups and its Application to Solvability by Radicals, Ph.D. Dissertation, EE & CS Department, March 1983

TR-289 Sirbu, M., Schoichet, S.R., Kunin, J.S., Hammer, M.M., Sutherland, J.B. and Zarmer, C.L.
Office Analysis: Methodology and Case Studies, EE & CS Department, March 1983

TR-290 Sutherland, J.B.
An Office Analysis and Diagnosis Methodology, S.M. Thesis, EE & CS Department, March 1983

TR-291 Pinter, R.Y.
The Impact of Layer Assignment Methods on Layout Algorithms for Integrated Circuits, Ph.D. Dissertation, EE & CS Department, August 1983

TR-292 Dornbrook, M. and Blank, M.
The MDL Programming Language Primer, EE & CS Department, June 1980

TR-293 Galley, S.W. and Pfister, G.
The MDL Programming Language, EE & CS Department, May 1979

TR-294 Lebling, P.D.
The MDL Programming Environment, EE & CS Department, May 1980

TR-295 Pitman, K.M.
The Revised Maclisp Manual, EE & CS Department, June 1983

TR-296 Church, K.W.
Phrase-Structure Parsing: A Method for Taking Advantage of Allophonic Constraints, Ph.D. Dissertation, EE & CS Department, June 1983

TR-297 Mok, A.
Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment, Ph.D. Dissertation, EE & CS Department, June 1983; A139996

TR-298 Krugler, K.
Video Games and Computer Aided Instruction, S.M. Thesis, EE & CS Department, June 1983

TR-299 Wing, J.M.
A Two-Tiered Approach to Specifying Programs, May 1983; A133949

TR-300 Cooper, G.H.
An Argument for Soft Layering of Protocols, S.M. Thesis, EE & CS Department, August 1983; A133948

TR-301 Valente, J.A.
Creating a Computer-Based Learning Environment for Physically Handicapped Children, Ph.D. Dissertation, EE & CS Department, September 1983

TR-302 Arvind, Dertouzos, M.L. and Iannucci, R.A.
A Multiprocessor Emulation Facility, October 1983; A136094

TR-303 Bloom T.
Dynamic Module Replacement in a Distributed Programming System, Ph.D. Dissertation, EE & CS Department, March 1983; A140619

TR-304 Terman, C.J.
Simulation Tools for Digital LSI Design, Ph.D. Dissertation, EE & CS Department, September 1983; A136116

TR-305 Bhatt, S.N. and Leighton, F.T.
A Framework for Solving VLSI Graph Layout Problems, October 1983; A136143

TR-306 Leung, K.C. and Lim, W. Y-P.
PADL – A Packet Architecture Description Language: A Preliminary Reference Manual, October 1983

TR-307 Guttag, J.V. and Horning, J.J.
Preliminary Report on the Larch Shared Language, October 1983; A136117

TR-308 Oki, B.M.
Reliable Object Storage to Support Atomic Actions, S.M. Thesis, EE & CS Department, May 1983; A136484

TR-309 Brock, J.D.
A Formal Model of Non-Determinate Dataflow Computation, Ph.D. Dissertation, EE & CS Department, November 1983

TR-310 Granville, R.
Cohesion in Computer Text Generation: Lexical Substitution, S.M. Thesis, EE & CS Department, May 1983; A148990

TR-311 Burke, G.S., Carrette, G.J. and Eliot, C.R.
NIL Reference Manual, January 1984

TR-312 Lancaster, J.N.
Naming in a Programming Support Environment, S.M. Thesis, EE & CS Department, August 1983; A142018

TR-313 Koile, K.
The Design and Implementation of an Online Directory Assistance System, S.M. Thesis, EE & CS Department, December 1983; A140821

TR-314 Weihl, W.E.
Specification and Implementation of Atomic Data Types, Ph.D. Dissertation, EE & CS Department, April 1984; A141823

TR-315 Coan, B.A. and Turpin, R.
Extending Binary Byzantine Agreement to Multivalued Byzantine Agreement, April 1984; A143424

TR-316 Comer, M.H.
Loose Consistency in a Personal Computer Mail System, S.B. & S.M. Thesis, EE & CS Department, May 1984

TR-317 Traub, K.R.
An Abstract Architecture for Parallel Graph Reduction, S.B. Thesis, EE & CS Department, May 1984

TR-318 Ashbell, I.J., M.D.
A Constraint Representation and Explanation Facility for Renal Physiology, S.M. Thesis, EE & CS Department, June 1984

TR-319 Herlihy, M.P.
Replication Methods for Abstract Data Types, Ph.D. Dissertation, EE & CS Department, May 1984; A153648

TR-320 Kornhauser, D.M.
Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups, and Applications, S.M. Thesis, EE & CS Department, May 1984

TR-321 Kuszmaul, B.C.
Type Checking in VIM VAL, Ph.D. Dissertation, EE & CS Department, June 1984

TR-322 Moulton, A.S.
Routing the Power and Ground Wires on a VLSI Chip, S.M. Thesis, EE & CS Department, May 1984; A145356

TR-323 Ackerman, W.B.
Efficient Implementation of Applicative Languages, Ph.D. Dissertation, EE & CS Department, April 1984

TR-324 Schooler, R.
Partial Evaluation as a Means of Language Extensibility, S.M. Thesis, EE & CS Department, August 1984; A148730

TR-325 Weiss, P.G.
Using Untyped Lambda Calculus to Compute With Atoms, S.M. Thesis, EE & CS Department, February 1984

TR-326 Walker, E.F.
Orphan Detection in the Argus System, S.M. Thesis, EE & CS Department, May 1984; A150562

TR-327 Chiu, S.Y.
Debugging Distributed Computations in a Nested Atomic Action System, Ph.D. Dissertation, EE & CS Department, December 1984; A153617

TR-328 Carnese, D.J.
Multiple Inheritance in Contemporary Programming Languages, September 1984

TR-329 Sacks, E.
Qualitative Mathematical Reasoning, November 1984

TR-330 Sarin, S.K.
Interactive On-Line Conferences, Ph.D. Dissertation, EE & CS Department, June 1984; A154723

TR-331 Sollins, K.R.
Distributed Name Management, Ph.D. Dissertation, EE & CS Department, February 1985; A154785

TR-332 Culler, D.E.
Resource Management for the Tagged Token Dataflow Architecture, S.M. Thesis, EE & CS Department, January 1985; A154773

TR-333 Arnold, J.M.
Parallel Simulation of Digital LSI Circuits, S.M. Thesis, EE & CS Department, February 1984; A154745

TR-334 Zarmer, C.L.
An Approach to Functional Office Automation, S.M. Thesis, EE & CS Department, April 1984

TR-335 Lundelius, J.
Synchronizing Clocks in a Distributed System, S.M. Thesis, EE & CS Department, August 1984

TR-336 Trilling, S.
Some Implications of Complexity Theory on Pseudo-Random Bit Generation, S.M. Thesis, EE & CS Department, January 1985

TR-337 Seiler, L.D.
A Hardware Assisted Methodology for VLSI Design Rule Checking, Ph.D. Dissertation, EE & CS Department, February 1985

TR-338 Koton, P.A.
Towards a Problem Solving System for Molecular Genetics, May 1985

TR-339 Soley, R.M.
Generic Software for Emulating Multiprocessor Architectures, May 1985; A157662

TR-340 Wellman, M.P.
Reasoning About Preference Models, S.M. Thesis, EE & CS Department, May 1985

TR-341 Boughton, G.A.
Routing Networks for Packet Communication Systems, Ph.D. Dissertation, EE & CS Department, August 1984

TR-342 Stark, E.W.
Foundations of a Theory of Specification for Distributed Systems, Ph.D. Dissertation, EE & CS Department, August 1984

TR-343 Forgaard, R.
A Program for Generating and Analyzing Term Rewriting Systems, S.M. Thesis, EE & CS Department, September 1984

TR-344 Yelick, K.A.
A Generalized Approach to Equational Unification, S.M. Thesis, EE & CS Department, August 1985; A163112

TR-345 Estrin, D.L.
Access to Inter-Organization Computer Networks, Ph.D. Dissertation, EE & CS Department, August 1985

TR-346 Cosmadakis, S.S.
Equational Theories and Database Constraints, Ph.D. Dissertation, EE & CS Department, August 1985

TR-347 Morecroft, L.E.
A Relative-Motion Microworld, S.M. Thesis, EE & CS Department, September 1985; A161856

TR-348 Yedwab, L.
On Playing Well in a Sum of Games, S.M. Thesis, EE & CS Department, August 1985

TR-349 Eisenberg, M.A.
Bochser: An Integrated Scheme Programming System, S.M. Thesis, EE & CS Department, August 1985

TR-350 Seliger, R.
Design and Implementation of a Distributed Program for Collaborative Editing, S.M. Thesis, EE & CS Department, September 1985

TR-351 Bhatt, S.N.
The Complexity of Graph Layout and Channel Routing for VLSI, Ph.D. Dissertation, EE & CS Department, February 1984

TR-352 Lucassen, J.M., Gifford, D.K., Berlin, S.T., and Burmaster, D.E.
Boston Community Information System User Manual (Version 6.0), April 1986; A171426

TR-353 Jagannathan, S.
Data Backup and Recovery in a Computer Architecture for Functional Programming, S.M. Thesis, EE & CS Department, October 1985

TR-354 Stamos, J.W.
Remote Evaluation, Ph.D. Dissertation, EE & CS Department, January 1986; A169739

TR-355 Guharoy, B.
Data Structure Management in a Data Flow Computer System, S.M. Thesis, EE & CS Department, May 1985

TR-356 Beckerle, M.J.
Logical Structures For Functional Languages, S.M. Thesis, EE & CS Department, February 1986; A169368

TR-357 Gibson, J.C.
Computation Management in a Single Address Space System, S.M. Thesis, EE & CS Department, January 1986

TR-358 Chaing, C.J.
Primitives for Real-Time Animation in Three Dimensions, S.M. Thesis, EE & CS Department, April 1986

TR-359 Feldmeier, D.C.
A CATV-Based High-Speed Packet-Switching Network Design, S.M. Thesis, EE & CS Department, April 1986; A171666

TR-360 Kunstaetter, R.
Intelligent Physiologic Modeling, S.M. Thesis, EE & CS Department, April 1986

TR-361 Barrington, D.A.
Bounded Width Branching Programs, Ph.D. Dissertation, EE & CS Department, June 1986

TR-362 Kuszmaul, B.C.
Simulating Applicative Architectures on the Connection Machine, S.M. Thesis, EE & CS Department, June 1986

TR-363 Marantz, J.D.
Exploiting Parallelism in VLSI CAD, S.M. Thesis, EE & CS Department, June 1986

TR-364 Lynch, N., Blaustein, B. and Siegel, M.
Correctness Conditions for Highly Available Replicated Databases, June 1986; A171427

TR-365 Morais, D.R.
ID World: An Environment for the Development of Dataflow Programs Written in ID, May 1986

TR-366 Younis, S.G.
The Clock Distribution System of the Multiprocessor Emulation Facility, S.B. Thesis, EE & CS Department, June 1986

TR-367 Lynch, N.A. and Merritt, M.
Introduction to the Theory of Nested Transactions, July 1986; A171428

TR-368 Scheifler, R.W. and Gettys, J.
The X Window System, October 1986; A176476

TR-369 Moses, Y. and Tuttle, M.R.
Programming Simultaneous Actions Using Common Knowledge, February 1987

TR-370 Traub, K.R.
A Compiler for the MIT Tagged-Token Dataflow Architecture, S.M. Thesis, EE & CS Department, August 1986

TR-371 Gao G.R.
A Pipelined Code Mapping Scheme for Static Data Flow Computers, Ph.D. Dissertation, EE & CS Department, August 1986

TR-372 Maley, F.M.
Compaction With Automatic Jog Introduction, S.M. Thesis, EE & CS Department, November 1986; A176525

TR-373 Segal, D.A., Gifford, D.K., Lucassen, J.M., Henderson, J.B., Berlin, G.T. and Burmaster, D.E.
Boston Community Information System User's Manual(Version 8.17), October 1987; A190380

TR-374 Goldberg, A.V.
Efficient Graph Algorithms for Sequential and Parallel Computers, Ph.D. Dissertation, EE & CS Department, February 1987; A178403

TR-375 Osborne, R.B.
Modeling the Performance of the Concert Multiprocessor, S.M. Thesis, EE & CS Department, May 1987; A183619

TR-376 Day, M.S.
Replication and Reconfiguration in a Distributed Mail Repository, S.M. Thesis, EE & CS Department, May 1987; A186967

TR-377 Ng, P.
Long Atomic Computations, Ph.D. Dissertation, EE & CS Department, October 1986; A174788

TR-378 Levitin, S.M.
MACE: A Multiprocessing Approach to Circuit Extraction, S.M. Thesis. EE & CS Department, October 1986

TR-379 Sloan, R.H.
The Notion of Security for Probabilistic Public Key Cryptosystems, S.M. Thesis EE & CS Department, October 1986

TR-380 Bradley, E.
Logic Simulation on a Multiprocessor, S.M. Thesis, EE & CS Department, October 1986; A175776

TR-381 Sherman, A.T.
Cryptology and VLSI (a two-part dissertation), Ph.D. Dissertation, EE & CS Department, October 1986; A175853

TR-382 Chien, A.A.
Congestion Control in Routing Networks, S.M. Thesis, EE & CS Department October 1986; A175785

TR-383 Strauss, M.M.
The Organization of Research in the Information Sciences, Case Studies in Japan and in the US, S.M. Thesis, EE & CS Department, December 1986

TR-384 Gifford, D. and Glasser, N.
Remote Pipes and Procedures for Efficient Distributed Communication, October 1986, Revised July 1987

TR-385 Dennis,J.
Data Flow Computer Architecture Final Report, October 1987

TR-386 St. Pierre, M.A.
A Simulation Environment for Schema, S.M. Thesis, EE & CS Department, December 1986

TR-387 Lynch, N.A. and Tuttle, M.
Hierarchical Correctness Proofs for Distributed Algorithms, S.M. Thesis, EE & CS Department, April 1987

TR-388 Hirsch, D.E.
An Expert System for Diagnosing Gait in Cerebral Palsy Patients, S.M. Thesis, EE & CS Department, May 1987

TR-389 Kohane, I.S.
Temporal Reasoning in Medical Expert Systems, Ph.D. Dissertation, EE & CS Department, May 1987

TR-390 Goldman, K.J.
Data Replication in Nested Transaction Systems, S.M. Thesis, EE & CS Department, May 1987; A182178

TR-391 Goldman, S.
Efficient Methods for Calculating Maximum Entropy Distributions, S.M. Thesis, EE & CS Department, May 1987

TR-392 Kolodney, L.K.
MAM: A Semi-Automatic Debugging Tool for Distributed Programs, S.M. Thesis, EE & CS Department, June 1987

TR-393 Chu, T.-A.
Synthesis of Self-Timed VLSI Circuits from Graph-Theoretic Specifications, Ph.D. Dissertation, EE & CS Department, June 1987

TR-394 Bodlaender, H.L.
Dynamic Programming on Graphs with Bounded Treewidth, June 1987

TR-395 Nuth, P.R.
Communication Patterns in a Symbolic Multiprocessor, S.M. Thesis, EE & CS Department, June 1987: A186-896

TR-396 Jang, Y.
KOLA: Knowledge Organization LAnguage, S.M. Thesis, EE & CS Department, October 1988

TR-397 Gifford, D.K., Heitmann, D., Segal, D.A., Cote, R.G., Tanacea, K. and Burmaster, D.E.
Boston Community Information System 1986 Experimental Test Results, August 1987; AD A187028

TR398 Gifford, D.K., Cote, R.G. and Segal, D.A.
Clipping Service User's Manual (Version 1.2), September 1987; AD 192543

TR-399 Gifford,D.K., Cote, R.G. and Segal, D.A.
Walter User's Manual (Version 1.0), September 1987; A192542

TR-400 Liskov, B., Day, M., Herlihy, M., Johnson, P. and Leavens, G.
Argus Reference Manual, November 1987; A190382

TR-401 Baldwin, R.
Rule Based Analysis of Computer Security Ph.D. Dissertation, EE & CS Department, March 1988; A195736

TR-402 Miller, J.
Multi-Scheme: A Parallel Processing System Base on MIT Scheme, Ph.D. Dissertation, EE & CS Department, September 1987; A190383

TR-403 Maley, F.M.
Single-Layer Wire Routing, Ph.D. Dissertation, EE & CS Department, August 1987; A186990

TR-404 Kolodner, E.K.
Recovery Using Virtual Memory, S.M. Thesis, EE & CS Department, July 1987; A187098

TR-405 Zachary, J.L.
A Framework for Incorporating Abstraction Mechanisms into the Logic Programming Paradigm, Ph.D. Dissertation, EE & CS Department, August 1987; A190381

TR-406 Muldrow, W.K.
Calvin: A Rule-Based Expert System for Improving Arrhythmia Detector Performance During Noisy ECGs, S.M. Thesis, EE & CS Department, September 1987.

TR-407 Gifford, D., Jouvelot, P., Lucassen, J. and Sheldon, M.
FX-87 Reference Manual, September 1987; A187031

TR-408 Lucassen, J.
Types and Effects-Towards the Integration of Functional and Imperative Programming, Ph.D. Dissertation, EE & CS Department, August 1987; A186930

TR-409 Ladin, R., Liskov, B., and Shrira, L.
A Technique for Constructing Highly-Available Services, January 1988; A192544

TR-410 Jing-Hwa Hwang, D.
Constructing a Highly-Available Location Service for a Distributed Environment, S.M. Thesis, EE & CS Department, January 1988; AD 192723

TR-411 Kaliski, B.
Elliptic Curves and Cryptography: A Pseudorandom Bit Generator and Other Tools, Ph.D. Dissertation, EE & CS Department, January 1988

TR-412 Berger, B. and Shor, P.W.
Approximation Algorithms for the Maximum Acyclic Subgraph Problem, September 1989

TR-413 Schapire, R.E.
Diversity-Based Infeience of Finite Automata, S.M. Thesis, EE & CS Department, May 1988

TR-414 Hicks, J.E.
A High-level Signal Processing Programming Language, S.M. Thesis, EE & CS Department, March 1988

TR-415 Margolus, N.H.
Physics and Computation, Ph.D. Dissertation, Physics Department, March 1988

TR-416 Sacks, E.P.
Automatic Qualitative Analysis of Ordinary Differential Equations Using Piecewise Linear Approximations, Ph.D. Dissertation, EE & CS Department, March 1988

TR-417 Traub, K.R.
Sequential Implementation of Lenient Programming Languages, Ph.D. Dissertation, EE & CS Department, October 1988; A200984

TR-418 Iannucci, R.A.
A Dataflow/Von Neumann Hybrid Architecture, Ph.D. Dissertation, EE & CS Department, July 1988

TR-419 Jackson, D.
Composing Data & Process Descriptions in the Design of Software Systems, S.M. Thesis, EE & CS Department, May 1988; A198096

TR-420 Gifford, D.K.
Polychannel Systems for Mass Digital Communication, July 1988

TR-421 Hammel, R.T. and Gifford, D.K.
FX-87 Performance Measurements: Dataflow Implementation, November 1988; A203150

TR-422 Gifford, D.K. and Segal, D.A.
Boston Community Information System 1987-1988 Experimental Text Results, May 1989

TR-423 Oki, B.M.
Viewstamped Replication for Highly Available Distributed Systems, Ph.D. Dissertation, EE & CS Department, May 1989

TR-424 Xu, A.S.
A Fault-Tolerant Network Kernel for Linda, S.M. Thesis, EE & CS Department, August 1988

TR-425 Maa, G.K.
Code-Mapping Policies for the Tagged-Token Dataflow Architecture, S.M. Thesis, EE & CS Department, May 1988; A198054

TR-426 Klein, P.N.
Efficient Parallel Algorithms for Planar, Chordal, and Interval Graphs, Ph.D. Dissertation, EE & CS Department, October 1988

TR-427 Wellman, M.P.
Formulation of Tradeoffs in Planning Under Uncertainty, Ph.D. Dissertation, EE & CS Department, August 1988

TR-428 Iyengar, A.K.
Parallel DNA Sequence Analysis, S.M. Thesis, October 1988

TR-429 Perlman, R.
Network Layer Protocols with Byzantine Robustness, Ph.D. Dissertation, EE & CS Department, October 1988

TR-430 Plotkin, S.A.
Graph-Theoretic Techniques for Parallel, Distributed, and Sequential Computation, Ph.D. Dissertation, EE & CS Department, September 1988; A200989

TR-431 Perl, S.E.
Distributed Commit Protocols for Nested Atomic Actions, S.M. Thesis, EE & CS Department, November 1988; A203900

TR-432 Papadopoulos, G.M.
Implementation of a General Purpose Dataflow Multiprocessor, Ph.D. Dissertation, EE & CS Department, December 1988; A207609

TR-433 Rosenberg, R.L.
Computer Literacy Education, Ph.D. Dissertation, EE & CS Department, January 1989; AD 209126

TR-434 Jagannathan, S.
A Programming Language Supporting First-Class Parallel Environments, Ph.D. Dissertation, EE & CS Department, January 1989; A207833

TR-435 Berger, B. and Rompel, J.
Simulating $(log^c n)$-wise Independence in NC, May 1989; A213974

TR-436 Berger, B.
Data-Structures for Removing Randomness, December 1988

TR-437 Kravets, D.
Finding Farthest Neighbors in a Convex Polygon and Related Problems, S.M. Thesis, EE & CS Department, January 1989; A210727

TR-438 Heller, S.K.
Efficient Lazy Data-Structures on a Dataflow Machine, February 1989; A209177

TR-439 Leavens, G.T.
Verifying Object-Oriented Programs that Use Subtypes, February 1989

TR-440 Aiello, W.A.
Proofs, Knowledge and Oracles Three Complexity Results for Interactive Proofs and Zero-Knowledge, Ph.D. Dissertation, EE & CS Department, February 1989

TR-441 Koton, P.A.
Using Experience in Learning and Problem Solving, Ph.D. Dissertation, EE & CS Department, March 1989

TR-443 Soley, R.M.
On the Efficient Exploitation of Speculation Under Dataflow Paradigms of Control, Ph.D. Dissertation, EE & CS Department, May 1989; A214112

TR-444 Berger, B., Rompel, J. and Shor, P.
Efficient NC Algorithms for Set Cover with Applications to Learning and Geometry, May 1989; AD 213975

TR-445 Blum, A.
On the Computation Complexity of Training Simple Neural Networks, S.M. Thesis, EE & CS Department, May 1989

TR-446 Culler, D.E.
Managing Parallelism and Resources in Scientific Dataflow Programs, Ph.D. Dissertation, EE & CS Department, March 1989

TR-447 Fortnow, L.J.
Complexity-Theoretic Aspects of Interactive Proof Systems, Ph.D. Dissertation, EE & CS Department, May 1989

TR-448 Sloan, R.H.
Computational Learning Theory: New Models and Algorithms, Ph.D. Dissertation, EE & CS Department, May 1989

TR-449 Onanian, J.S.
A Signal Processing Language for Coarse Grain Dataflow Multiprocessors, S.M. Thesis, EE & CS Department, June 1989; A213863

TR-450 MIT Computer Science Research Symposium
Copies of Speakers' Viewgraphs, Viewgraphs used for talks given at the Symposium celebrating the 25^{th} Anniversary of Project MAC, October 1988

TR-452 Cherian, M.M.
A Study of Backoff Barrier Synchronization, S.M. Thesis, EE & CS Department, June 1989

TR-453 Gruber, R.E.
Optimistic Concurrency Control for Nested Distributed Transactions, S.M. Thesis, EE & CS Department, June 1989; A213828

TR-454 Katz, M.
ParaTran: A Transparent, Transaction Based Runtime Mechanism for Parallel Execution of Scheme, S.M. Thesis, EE & CS Department, July 1989

TR-455 Zhang, L.
A New Architecture for Packet Switching Network Protocols, Ph.D. Dissertation, EE & CS Department, August 1989

TR-456 Greenberg, R.I.
Efficient Interconnection Schemes for VLSI and Parallel Computation, Ph.D. Dissertation, EE & CS Department, August 1989

TR-457 Stein, C.
Using Cycles and Scaling in Parallel Algorithms, S.M. Thesis, EE & CS Department, August 1989

TR-458 Reinhold, M.B.
Typechecking Is Undecidable When 'Type' Is a Type, S.M. Thesis, EE & CS Department, December 1989

TR-459 Riecke, J.G.
Should a Function Continue?, S.M. Thesis, EE & CS Department, September 1989

TR-460 Harris, N.L.
Probabilistic Reasoning in the Domain of Genetic Counseling, S.M. Thesis, EE & CS Department, October 1989

TR-461 Haimowitz, I.J.
Generating Empathetic Responses with Individual User Models, December 1989

TR-462 Phillips, C.A.
Theoretical and Experimental Analyses of Parallel Combinatorial Algorithms, Ph.D. Dissertation, EE & CS Department, October 1989

TR-463 Blelloch, G.E.
Scan Primitives and Parallel Vector Models, Ph.D. Dissertation, EE & CS Department, October 1989

TR-464 Osborne, R.B.
Speculative Computation in Multilisp, Ph.D. Dissertation, EE & CS Department, December 1989

TR-465 Hashem, E.S.
Analysis of Random Drop for Gateway Congestion Control, S.M. Thesis, EE & CS Department, November 1989

TR-466 Vuong-Adlerberg, I.
Cache for Multi-Threaded Processors on a Split-Transaction Bus, November 1989

TR-467 Morrison, J.D.
A Scalable Multiprocessor Architecture Using Cartesian Network-Relative Addressing, S.M. Thesis, EE & CS Department, December 1989

TR-468 Jategaonkar, L.A.
ML with Extended Pattern Matching and Subtypes, S.B. & S.M. Thesis, EE & CS Department, August 1989

TR-469 Maggs, B.M.
Locality in Parallel Computation, Ph.D. Dissertation, EE & CS Department, September 1989; A218733

TR-470 Heybey, A.T.
Rate-Based Congestion Control in Networks with Smart Links, S.B. Thesis, EE & CS Department, January 1990

TR-471 Steele, K.M.
Implementation of an I-Structure Memory Controller, Ph.D. Dissertation, EE & CS Department, March 1990; A221397

TR-473 Ghemawat, S.
Automatic Replication for Highly Available Services, S.M. Thesis, EE & CS Department, March 1990

TR-474 Troxel, G.D.
A Hierarchical Proof of an Algorithm for Deadlock Recovery in a System using Remote Procedure Calls, S.M. Thesis, EE & CS Department, January 1990; A222116

TR-475 Ben-Zvi, B.
Disconnected Actions: An Asynchronous Extension to a Nested Atomic Action System, S.M. Thesis, EE & CS Department, January 1990

TR-476 Liskov, B., Shrira, L. and Wroclawski, J.
Efficient At-Most-Once Messages Based on Synchronized Clocks, April 1990; A223004

TR-477 Tuttle, M.
Knowledge and Distributed Computation, Ph.D. Dissertation, EE & CS Department, May 1990; A223100

TR-478 Aghassi, D.S.
Evaluating Case-Based Reasoning for Heart Failure Diagnosis, S.M. Thesis, EE & CS Department, June 1990

TR-479 Williamson, D.P.
Analysis of the Held-Karp Heuristic for the Traveling Salesman Problem, S.M. Thesis, EE & CS Department, June 1990

Annual Reports

- Project MAC Progress Report I, to July 1964; AD 465088
- Project MAC Progress Report II, July 1964-July 1965; AD 629494
- Project MAC Progress Report III, July 1965-July 1966; AD 648346
- Project Mac Progress Report IV, July 1966-July 1967; AD 681342
- Project MAC Progress Report V, July 1967-July 1968; AD 687770
- Project MAC Progress Report VI, July 1968-July 1969; AD 705434
- Project MAC Progress Report VII, July 1969-July 1970; AD 732767
- Project MAC Progress Report VIII, July 1970-July 1971; AD 735148
- Project MAC Progress Report IX, July 1971-July 1972; AD 756689
- Project MAC Progress Report X, July 1972-July 1973; AD 771428
- Project MAC Progress Report XI, July 1973-July 1974; AD 04966
- Laboratory for Computer Science Progress Report XII, July 1974-July 1975; A024527
- Laboratory for Computer Science Progress Report XIII, July 1975-July 1976; A061246
- Laboratory for Computer Science Progress Report XIV, July 1976-July 1977; A061932
- Laboratory for Computer Science Progress Report 15, July 1977-July 1978; A073958
- Laboratory for Computer Science Progress Report 16, July 1978-July 1979; A088355
- Laboratory for Computer Science Progress Report 17, July 1979-July 1980; A093384
- Laboratory for Computer Science Progress Report 18, July 1980-June 1981; A127586
- Laboratory for Computer Science Progress Report 19, July 1981-June 1982; A143429
- Laboratory for Computer Science Progress Report 20, July 1982-June 1983; A145134
- Laboratory for Computer Science Progress Report 21, July 1983-June 1984; A154810
- Laboratory for Computer Science Progress Report 22, July 1984-June 1985
- Laboratory for Computer Science Progress Report 23, July 1985-June 1986
- Laboratory for Computer Science Progress Report 24, July 1986-June 1987
- Laboratory for Computer Science Progress Report 25, July 1987-June 1988
- Laboratory for Computer Science Progress Report 26, July 1988-June 1989

Copies of all reports with A; AD, or PB numbers listed in Publications may be secured from the National Technical Information Service, U.S. Department of Commerce, Reports Division, 5285 Port Royal Road, Springfield. Virginia 22161 (tel: 703-487-4650). Prices vary. The reference number must be supplied with the request.

References

[1] S. Abramsky. The lazy lambda calculus. In D. L. Turner, editor, *Declarative Programming*, Addison-Wesley, 1989.

[2] Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt, and N. Shavit. Atomic snapshots of shared memory. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, Quebec, Canada, August 1990. Also Technical Memo MIT/LCS/TM-429, MIT Laboratory for Computer Science, May 1990. Submitted to *Journal of the ACM.*

[3] Y. Afek and E. Gafni. End-to-end communication in unreliable networks. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing*, pages 131–148, 1988.

[4] A. Aggarwal, M. Hansen, and T. Leighton. Solving query-retrieval problems by compacting Voronoi diagrams. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 331–340, 1990.

[5] A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(2):195–208, 1987.

[6] A. Aggarwal, D. Kravets, J. Park, and S. Sen. Parallel searching in generalized Monge arrays with applications. In *Proceedings of the Second Annual ACM Symposium on Parallel Algorithms and Architectures*, 1990. To appear.

[7] A. Aggarwal and J. Park. Algorithms for economic lot-size problems with bounds on inventory and backlogged demand. 1990. Manuscript in preparation.

[8] A. Aggarwal and J. Park. *Improved Algorithms for Economic Lot-Size Problems.* Research Report RC 15626, IBM Research Division, T. J. Watson Research Center, March 1990.

[9] G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, 1986.

[10] A. Albano, L. Cardelli, and R. Orsini. A Strongly Typed Interactive Conceptual Language. *ACM Transactions on Database Systems*, 10(2):230–260, June 1985.

[11] E. Anagnostou, L. Guibas, and V. Polimenis. Topological sweeping in three dimensions. In *Proceedings of the First SIGAL Symposium*, 1990. To appear.

[12] D. Angluin. Finding patterns common to a set of strings. *JCSS*, 21(1):46–62, August 1980.

[13] Z. M. Ariola and Arvind. P-tac: a parallel intermediate language. In *Proceedings of the Conference on Functional Programming Languages and Computer Architecture*, September 1989. Also: CSG Memo 295, MIT Laboratory for Computer Science.

[14] B. Aronov, B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, and R. Wenger. Points and triangles in the plane and halving planes in space. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, ACM, 1990. To appear.

[15] T. Asano, L. Guibas, and T. Tokuyama. Walking on an arrangement topologically. In preparation.

[16] J. A. Aslam and A. Dhagat. Online algorithms for the Lovász local lemma via chip games. Submitted for publication.

[17] J. A. Aslam and R. L. Rivest. Minimum consistent inference of random walks. Submitted for publication.

[18] M. P. Atkinson, P. J. Bailey, K. J. Chisholm, P. W. Cockshott, and R. Morrison. An approach to persistent programming. *The Computer Journal*, 26(4):360–365, 1983.

[19] H. Attiya, A. Bar-Noy, and D. Dolev. Sharing memory robustly in message-passing systems. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, Quebec, Canada, August 1990. Expanded version: Technical Memo MIT/LCS/TM-423, MIT Laboratory for Computer Science, February 1990. Submitted to *Journal of the ACM.*

[20] H. Attiya, C. Dwork, N. A. Lynch, and L. J. Stockmeyer. Bounds on the time to reach agreement in the presence of timing uncertainty. In preparation.

[21] H. Attiya and N. Lynch. Time bounds for real-time process control in the presence of timing uncertainty. Submitted for publication.

[22] H. Attiya, N. Lynch, and N. Shavit. Are wait-free algorithms fast? Submitted for publication.

[23] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, 1989. To appear.

[24] B. Awerbuch, I. Cidon, I. Gopal, M. Kaplan, and S. Kutten. Distributed control for PARIS. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, 1989. To appear.

[25] B. Awerbuch, I. Cidon, and S. Kutten. Optimal maintenance of replicated information. April 1990. Unpublished manuscript.

[26] B. Awerbuch, A. Goldberg, M. Luby, and S. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of the 30^{th} Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[27] B. Awerbuch, O. Goldreich, and A. Herzberg. A quantitative approach to dynamic networks. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, 1989. To appear.

[28] B. Awerbuch, Y. Mansour, and N. Shavit. End-to-end communication with polynomial overhead. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[29] B. Awerbuch and D. Peleg. Online tracking of mobile users. August 1989. Unpublished manuscript.

[30] B. Awerbuch and D. Peleg. Routing with polynomial communication-space trade-off. August 1989. Unpublished manuscript.

[31] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, 1985.

[32] H. Bal. *Fault-tolerant Parallel Programming in Argus.* Report IR-214, Department of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, May 1990.

[33] E. Balas and N. Christofides. A restricted Lagrangean approach to the traveling salesman problem. *Mathematical Programming*, 21:19–46, 1981.

[34] R. A. Bankowitz, M. A. McNeil, S. M. Challinor, R. C. Parker, W. N. Kapoor, and R. A. Miller. A computer-assisted medical diagnostic consultation service. *Annals of Internal Medicine*, 110(10):824–832, May 1989.

[35] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Computer Conference*, 32, pages 307–314, 1968.

[36] G. M. Baudet. Asynchronous iterative methods for multiprocessors. *Journal of the ACM*, 25(2):226–244, April 1978.

[37] R. Bayer and M. Schkolnick. Concurrency of operations on b-trees. *Acta Informatica*, 9:1–22, 1977.

[38] D. Beaver, J. Feignebaum, J. Kilian, and P. Rogaway. Cryptographic applications of locally random reductions. 1989. Unpublished manuscript.

[39] D. Beaver and S. Goldwasser. Multi party fault tolerant computation with faulty majority based on oblivious transfer. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[40] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 503–513, 1990.

[41] M. Bellare, L. Cowen, and S. Goldwasser. On the structure of secret key exchange protocols. In J. Feigenbaum and M. Merritt, editors, *Proceedings of the DIMACS Workshop on Distributed Computing and Cryptography*, American Mathematical Society, Princeton, NJ, 1989.

[42] M. Bellare, S. Goldwasser, and J. Rompel. Randomness in interactive proofs. 1990. Submitted to 31st Annual Symposium on Foundations of Computer Science.

[43] M. Bellare and S. Micali. How to sign given any trapdoor function. *SIAM J. Computing*, 1989. Accepted, subject to minor revisions.

[44] M. Bellare, S. Micali, and R. Ostrovsky. From asyntotics to practice. In *Advances in Cryptology—CRYPTO '90*, Lecture Notes in Computer Science, Springer-Verlag, 1990.

[45] M. Bellare, S. Micali, and R. Ostrovsky. Perfect zero-knowledge in constant rounds. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 482–493, 1990.

[46] M. Bellare, S. Micali, and R. Ostrovsky. The (true) complexity of statistical zero knowledge. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 494–502, 1990.

[47] B. Berger. *Using Randomness to Design Efficient Deterministic Algorithms.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[48] B. Berger and L. Cowen. $\{<, \leq, =\}$*-Constrained Scheduling.* Technical Report CICS-P-236, MIT Center for Intelligent Control Systems, June 1990.

[49] B. Berger and J. Rompel. Simulating $(\log^c n)$-wise independence in NC. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 2–7, IEEE, 1989. Winner of 1989 Machtey Award. To appear in *Journal of the ACM.*

[50] B. Berger, J. Rompel, and P. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 54–59, IEEE, 1989. Submitted to *Journal of Algorithms* FOCS '89 special issue.

[51] B. Berger and P. W. Shor. Approximation algorithms for the maximum acyclic subgraph problem. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 236–243, 1990.

[52] G. Berry, P. Curien, and J. Lévy. Full abstraction for sequential languages: the state of the art. In M. Nivat and J. C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 89–132, Cambridge University Press, 1985.

[53] M. Biafore. *Two-dimensional universal automaton with electron-like tokens.* Technical Report MIT/LCS/TM-429, MIT Laboratory for Computer Science, June 1990.

[54] B. Bloom. Can LCF be topped? Flat lattice models of typed lambda calculus. In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, pages 282–295, IEEE, 1988.

[55] B. Bloom. *Partial Traces and the Semantics and Logic of CCS-like Languages.* Technical Report 89-1066, Cornell University, 1989.

[56] B. Bloom. *Ready Simulation, Bisimulation, and the Semantics of CCS-Like Languages.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, August 1989. Supervised by A.R. Meyer.

[57] B. Bloom and A. R. Meyer. A remark on the bisimulation of probabilistic processes. In A. R. Meyer and M. A. Taitslin, editors, *Logic at Botik '89: Symposium on Logical Foundations of Computer Science*, pages 26–40, Volume 363 of Lecture Notes in Computer Science, Springer-Verlag, 1989.

[58] B. Bloom and A. R. Meyer. Experimenting with process equivalence. In *Proceedings of the International BCS-FACS Workshop on Semantics for Concurrency*, Lecture Notes in Computer Science, Springer-Verlag, July 1990. To appear.

[59] B. Bloom and J. G. Riecke. LCF should be lifted. In T. Rus, editor, *Proceeding of the Conference on Algebraic Methodology and Software Technology*, pages 133–136, Department of Computer Science, University of Iowa, 1989.

[60] A. Blum. Learning boolean functions in an infinite attribute space. In *Proceedings of the 22^{nd} Annual ACM Symposium on Theory of Computing*, pages 64–72, 1990.

[61] A. Blum. Learning boolean functions in an infinite attribute space. Lecture given at Proceedings of the 22^{nd} Annual ACM Symposium on Theory of Computing, May 1990.

[62] A. Blum. Separating PAC and mistake-bound learning models over the boolean domain. April 1990. Submitted for publication.

[63] R. Boppana and M. Sipser. The complexity of finite functions. *Handbook of Theoretical Computer Science*, 1990. To appear.

[64] J. Bresnan, editor. *The Mental Representation of Grammatical Relations.* MIT Press, 1982.

[65] A. Buzacott. *The Development of Broadband Telecommunications Standards.* Master's thesis, Massachusetts Institute of Technology, June 1990.

[66] M. Carey, D. DeWitt, J. Richardson, and E. Sheikta. Object and file management in the EXODUS extensible database system. In *Proceedings of the 12^{th} International Conference on Very Large Databases*, August 1986.

[67] J. D. Case, J. R. Davin, M. S. Fedor, and M. L. Schoffstall. Internet Network Management Using the Simple Network Management Protocol. In *Proceedings of the 14^{th} IEEE Conference on Local Computer Networks*, Minneapolis, MN, October 1989.

[68] J. D. Case, M. S. Fedor, M. L. Schoffstall, and J. R. Davin. *A Simple Network Management Protocol.* Request for Comments 1157, DDN Network Information Center, SRI International, May 1990.

[69] L. Chalfin and B. S. Tsirelson. *Quantum/classical correspondence in the light of Bell's inequalities.* Technical Report MIT/LCS/TM-420, MIT Laboratory for Computer Science, 1990.

[70] B. Chazelle, H. Edelsbrunner, L. Guibas, J. Hershberger, R. Seidel, and M. Sharir. Slimming down by adding; selecting heavily covered points. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, ACM, 1990. To appear.

[71] A. Chien and W. Dally. Concurrent aggregates (CA). In *Proceedings of the Second ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Seattle, WA, March 1990.

[72] B. Chopard. Cellular automata model for the diffusion equation. *J. Stat. Phys.*, 1990. Submitted for publication.

[73] B. Chopard. A cellular automata model of large scale moving objects. *J. Phys. A*, 1990. To appear.

[74] I. Cidon, S. Kutten, Y. Mansour, and D. Peleg. Greedy packet scheduling. 1990. Unpublished manuscript.

[75] W. J. Clancey and E. H. Shortliffe, editors. *Readings in Medical Artificial Intelligence: The First Decade.* Addison-Wesley, 1984.

[76] J. L. Coburn. *Xil: A Scheme Embedded Hardware Description Language for Xilinx Chip Configuration.* Bachelor's thesis, Massachusetts Institute of Technology, June 1990.

[77] J. Cohen. *Distributed Atomic Stable Storage.* Master's thesis, Department of Electrical Engineering and Computer Science, 1989.

[78] F. Commoner. Synchronization graphs for quantum computation. To appear.

[79] G. F. Cooper. *The Computational Complexity of Probabilistic Inference Using Belief Networks.* Memo KSL-87-27, Knowledge Systems Laboratory, Stanford University, May 1988.

[80] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press/McGraw-Hill, 1990.

[81] S. S. Cosmadakis. Computing with recursive types. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science*, pages 24–38, IEEE, 1989.

[82] S. S. Cosmadakis, A. R. Meyer, and J. G. Riecke. Completeness for typed lazy inequalities. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 312–320, 1990.

[83] P. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming.* John Wiley and Sons, 1986.

[84] L. Damas and R. Milner. Principal type-schemes for functional programs. In *Proceedings of the Ninth Symposium on Principles of Programming Languages*, pages 207–212, January 1982.

[85] D. S. Daniels, A. Z. Spector, and D. S. Thompson. Distributed logging for transaction processing. In *In ACM Special Interest Group on Management of Data 1987 Annual Conference*, pages 82–96, ACM SIGMOD, May 1987.

[86] J. R. Davin, J. M. Galvin, and K. McCloghrie. *Administration of SNMP Communities.* Internet Draft, May 1990.

[87] N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Proceedings of Koninklijke Nederlandse Akademie van Wetenschappen, Series A, Mathematical Sciences*, 75:381–392, 1972.

[88] J. de Kleer. *Causal and Teleological Reasoning in Circuit Recognition.* AI-TR 529, MIT Artificial Intelligence Laboratory, September 1979.

[89] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings ACM SIGCOMM*, pages 1–12, September 1989.

[90] D. Detlefs, M. Herlihy, and J. Wing. Inheritance of synchronization and recovery properties in avalon/C++. *IEEE Computer*, 21(12), December 1988.

[91] J. Doyle. Reasoning, representation, and rational self-government. In Z. W. Ras, editor, *Methodologies for Intelligent Systems, 4*, pages 367–380, North-Holland, 1989.

[92] J. Doyle. The foundations of psychology: a logico-computational inquiry into the concept of mind. In R. Cummins and J. Pollock, editors, *Philosophy and AI: Essays at the Interface*, Bradford/MIT Press, 1990. To appear.

[93] J. Doyle. Rational belief revision. In K. Konolige, editor, *Proceedings of the Third International Workshop on Nonmonotonic Reasoning*, 1990. To appear.

[94] J. Doyle. Rational control of reasoning in artificial intelligence. In A. Fuhrmann, editor, *The Logic of Theory Change*, Springer-Verlag, 1990. To appear.

[95] J. Doyle. Rational self-government and universal default logics. In J. Le Moigne and P. Bourgine, editors, *Proceedings of the Second International Conference on Economics and Artificial Intelligence*, July 1990. To appear.

[96] J. Doyle. Rationality and its roles in reasoning (extended abstract). In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990. To appear.

[97] J. Doyle and R. S. Patil. *Two Dogmas of Knowledge Representation: Language Restrictions, Taxonomic Classifications, and the Utility of Representation Services.* MIT/LCS/TM-387.b, MIT Laboratory for Computer Science, September 1989.

[98] J. Doyle and E. P. Sacks. Stochastic analysis of qualitative dynamics. In N. S. Sridharan, editor, *Proceedings of the 11^{th} International Joint Conference on Artificial Intelligence*, pages 1187–1192, Morgan Kaufmann, 1989.

[99] J. Doyle and M. P. Wellman. Impediments to universal preference-based default theories. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 94–102, Morgan Kaufmann, May 1989.

[100] J. M. Draper. Compiling in Horizon. In *Proceedings of Supercomputing '88*, pages 51–52, IEEE, Orlando, FL, November 1988.

[101] H. Edelsbrunner and L. J. Guibas. Topologically sweeping an arrangement. *JCSS*, 38(1):165–194, 1989.

[102] A. El-Abbadi, D. Skeen, and F. Cristian. An efficient fault-tolerant protocol for replicated data management. In *Proceedings of the Fourth Symposium on Principles of Database Systems*, pages 215–229, ACM, 1985.

[103] A. El-Abbadi and S. Toueg. Maintaining availability in partitioned replicated databases. In *Proceedings of the Fifth Symposium on Principles of Database Systems*, pages 240–251, ACM, 1986.

[104] P. Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 1990. Accepted for publication.

[105] J. R. Ellis, K. Li, and A. W. Appel. *Real-time Concurrent Collection on Stock Multiprocessors.* Technical Report 25, Systems Research Center, Digital Equipment Corporation, February 1988.

[106] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. In *Advances in Cryptology—CRYPTO '89*, Lecture Notes in Computer Science, Springer-Verlag, 1989.

[107] E. Feigenbaum, P. McCorduck, and H. P. Nii. *The Rise of the Expert Company.* Time Books, 1988.

[108] A. Fekete and N. Lynch. The need for headers: an impossibility result for communication over unreliable channels. Also Technical Memo MIT/LCS/TM-428, MIT Laboratory for Computer Science May, 1990. Submitted for publication. Also to appear in *CONCUR 1990.*

[109] A. Fekete, N. Lynch, M. Merritt, and W. Weihl. Commutativity-based locking for nested transactions. In *Proceedings of Third International Workshop on Persistent Object Systems*, pages 113–127, Newcastle, Australia, January 1989. Revised version to appear in *Journal of Computer and System Sciences.*

[110] A. Fekete, N. Lynch, M. Merritt, and W. Weihl. *Atomic transactions.* Book in progress.

[111] A. Fekete, N. Lynch, and W. Weihl. A serialization graph construction for nested transactions. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 94–108, Nashville, TN, April 1990.

[112] A. Fekete, N. A. Lynch, M. Merritt, and W. E. Weihl. *Commutativity-based Locking for Nested Transactions.* Technical Memo MIT/LCS/TM-370.b, MIT Laboratory for Computer Science, July 1989. Supercedes MIT/LCS/TM-370.

[113] A. Fekete, N. A. Lynch, and W. E. Weihl. A serialization graph construction for nested transactions. In *Proceedings of the ACM Symposium on Principles of Database Systems*, Nashville, TN, April 1990. See also MIT/LCS/TM-421.

[114] L. Fortnow, H. Karloff, C. Lund, and N. Nisan. The polynomial time hierarchy has interactive proofs. 1990. Manuscript.

[115] J. M. Galvin, K. McCloghrie, and J. R. Davin. *Authentication and Privacy in the SNMP.* Internet Draft, May 1990.

[116] D. Gelernter, N. Carriero, S. Chandran, and S. Chang. Parallel programming in Linda. In *Proceedings of International Conference on Parallel Processing*, IEEE, 1985.

[117] S. Ghemawat. *Automatic Replication for Highly Available Services.* Technical Report MIT/LCS/TR-473, MIT Laboratory for Computer Science, 1990.

[118] J. R. Glass. *Finding Acoustic Regularities in Speech: Applications to Phonetic Recognition.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, May 1988.

[119] W. Goddard, V. King, and L. Schulman. Optimal randomized algorithms for local sorting and set-maxima. In *Proceedings of the 22^{nd} Annual ACM Symposium on Theory of Computing*, pages 45–53, 1990.

[120] K. Goldman. Highly concurrent logically synchronous multicast. In *Proceedings of the Third International Workshop on Distributed Algorithms*, September 1989. Longer version as Technical Memo MIT/LCS/TM-401, MIT Laboratory for Computer Science, July 1989. Also submitted to *Distributed Computing*

[121] K. Goldman. Superposition in the I/O automaton model. In progress.

[122] K. Goldman and A. Lynch. Modelling shared state in a shared action model. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, June 1990. Also Technical Memo MIT/LCS/TM-427, MIT Laboratory for Computer Science, March 1990.

[123] S. A. Goldman. *Learning Binary Relations, Total Orders, and Subclasses of Read-once Formulas.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, June 1990. In preparation.

[124] S. A. Goldman, M. J. Kearns, and R. E. Schapire. Exact identification of circuits using fixed points of amplification functions. April 1990. Submitted for publication.

[125] S. A. Goldman, M. J. Kearns, and R. E. Schapire. On the sample complexity of weak learning. April 1990. Submitted for publication.

[126] S. A. Goldman, R. L. Rivest, and R. E. Schapire. Learning binary relations and total orders. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 46–51, IEEE, 1989. Abstract also appeared in *Proceedings of the Second Annual Workshop on Computational Learning Theory*, July 1989.

[127] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[128] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *29th Annual Symposium on Foundations of Computer Science*, IEEE, 1988.

[129] S. Goldwasser and L. Levin. Computing general functions fairly in presence of immoral majority. In *Advances in Cryptology—CRYPTO '90*, Lecture Notes in Computer Science, Springer-Verlag, 1990.

[130] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18(1):186–208, February 1989. A preliminary version appeared in *STOC 85.*

[131] G. A. Gorry, H. Silverman, and S. G. Pauker. Capturing clinical expertise: a computer program that considers clinical responses to digitalis. *American Journal of Medicine*, 64:452–460, March 1978.

[132] J. N. Gray. *Notes on Database Operating Systems*, pages 393–481. Volume 60 of Lecture Notes in Computer Science, Springer-Verlag, 1978.

[133] M. Grigni and D. Peleg. Tight bounds on minimum broadcast networks. *SIAM Journal on Discrete Mathematics*, 1990. To appear. An earlier version appears in MIT/LCS/TM-374.

[134] M. Grigni and M. Sipser. On monotone complexity classes. In *Conference on the Complexity of Boolean Functions*, 1990. To appear.

[135] L. Guibas, J. Hershberger, and J. Snoeyink. Compact interval trees: a data structure for convex hulls. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 169–178, 1990.

[136] L. Guibas, D. Knuth, and M. Sharir. Randomized incremental construction of delaunay and voronoi diagrams. In *Automata, Languages and Programming: 17th International Colloquium.* Lecture Notes in Computer Science, Springer-Verlag, 1990. To appear.

[137] L. Guibas, D. Salesin, and J. Stolfi. Constructing strongly convex approximate hulls with inaccurate primitives. In *Proceedings of the First SIGAL Symposium*, 1990. To appear.

[138] M. Gupta. *I/O Automaton based Simulation of Selected Distributed Algorithms.* Bachelor's thesis. MIT Department of Electrical Engineering and Computer Science. June 1990. Supervised by N.A. Lynch.

[139] M. D. Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[140] D. Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[141] E. Hashem. *Analysis of Random Drop for Gateway Congestion Control.* Master's thesis, Massachusetts Institute of Technology, August 1989.

[142] D. Haussler. Generalizing the PAC model: sample size bounds from metric dimension-based uniform convergence results. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 40–45, IEEE, 1989.

[143] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[144] M. Herlihy and J. Wing. Axioms for concurrent objects. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, pages 13–26, January 1987.

[145] M. P. Herlihy and B. Liskov. A value transmission method for abstract data types. *ACM Transactions on Programming Languages and Systems*, 4(4):527–551, October 1982.

[146] D. H. Hickam, E. L. Shortliffe, M. B. Bischoff, A. C. Scott, and C. D. Jacobs. The treatment advice of a computer-based chemotherapy protocol advisor. *Annals of Internal Medicine*, 103(6):928–36, 1985.

[147] C. A. R. Hoare. Monitors: an operating system structuring concept. *Communications of the ACM*, 17(10):549–557, October 1974.

[148] H. Hrgovˇ cić. *The Local Interpretation of Quantum Mechanics.* PhD thesis, MIT, 1990. In progress.

[149] A. T. Ishii and C. E. Leiserson. A timing analysis of level-clocked circuitry. In *Advanced Research in VLSI: Proceedings of the Sixth MIT Conference*, pages 113–130, April 1990.

[150] G. Itkis and L. A. Levin. Power of fast VLSI models is insensitive to wires' thinness. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 402–407, IEEE, 1989.

[151] L. A. Jategaonkar. *ML with Extended Pattern Matching and Subtypes.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, August 1989.

[152] L. A. Jategaonkar. *ML with Extended Pattern Matching and Subtypes.* Technical Report MIT/LCS/TR-468, MIT Laboratory for Computer Science, August 1989.

[153] B. Jonsson. A fully abstract trace model for dataflow networks. In *Conference Record of the 16th Annual ACM Symposium on Principles of Programming Languages*, pages 155–165, 1989.

[154] A. Joseph and W. E. Weihl. Application-specific recovery: asynchronous iterative methods. June 1990. To appear.

[155] N. Kahale. New modular properties of the Bell numbers. *Journal of Combinatorial Theory Series A*, 1989. Accepted for publication.

[156] V. K. Kathail. *Optimal Interpreters for Lambda-calculus Based Functional Languages.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[157] M. Kearns and L. Pitt. A polynomial-time algorithm for learning k-variable pattern languages from examples. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 57–71, Morgan Kaufmann, August 1989.

[158] M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. April 1990. Submitted for publication.

[159] J. Kilian. Achieving zero-knowledge robustly. 1990. Unpublished manuscript.

[160] J. Kilian. Interactive proofs with provable security against passive adversaries. 1990. Unpublished manuscript.

[161] J. Kilian. On the power of finite cryptographic games. 1990. Unpublished manuscript.

[162] J. Kilian, S. Kipnis, and C. E. Leiserson. The organization of permutation architectures with bussed interconnections. *IEEE Transactions on Computers*, 1989. To appear. Also appeared as technical memo MIT/LCS/TM-379 and VLSI memo 89-500. Earlier version appeared in *Proceedings of the 28th IEEE Annual Symposium on Foundations of Computer Science*, pages 305–315, 1987.

[163] J. Kilian and S. Micali. Secret, physically secure computation. 1990. In preparation.

[164] J. Kilian, S. Micali, and R. Ostrovsky. Minimum resource zero-knowledge proofs. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 474–479, IEEE, 1989.

[165] J. Kilian, S. Micali, and R. Ostrovsky. Simple non-interactive zero-knowledge proofs. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[166] J. Kilian, S. Micali, and P. Rogaway. The notion of secure computation. In *Advances in Cryptology—CRYPTO '90*, Lecture Notes in Computer Science, Springer-Verlag, 1990.

[167] S. Kipnis. *Priority Arbitration with Busses.* Technical Memo MIT/LCS/TM-408, MIT Laboratory for Computer Science, October 1989. Also appeared in *Proceedings of the Sixth MIT Conference on Advanced Research in VLSI*, April 1990.

[168] S. Kipnis. *Three Methods for Range Queries in Computational Geometry.* Technical Memo MIT/LCS/TM-388, MIT Laboratory for Computer Science, March 1989.

[169] P. Klein and C. Stein. A parallel algorithm for approximating the minimum cycle cover. 1989. Submitted for publication.

[170] P. Klein and C. Stein. A parallel algorithm for eliminating cycles in undirected graphs. *Information Processing Letters*, 1990. To appear.

[171] R. Koch, T. Leighton, B. Maggs, S. Rao, and A. Rosenberg. Work-preserving emulations of fixed-connection networks. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 227–240, 1989.

[172] J. N. Kok. A fully abstract semantics for data flow nets. In J. W. de Bakker, A. J. Nijman, and P. C. Treleaven, editors, *PARLE: Parallel Architectures and Languages Europe, Volume II: Parallel Languages*, pages 351–368, Volume 259 of Lecture Notes in Computer Science, Springer-Verlag, 1987.

[173] E. Kolodner. *Design Bug In Argus Recovery System.* DSG Note 158, MIT Laboratory for Computer Science November 1989.

[174] E. Kolodner, B. Liskov, and W. Weihl. Atomic Garbage Collection: Managing a Stable Heap. In *Proceedings of the 1989 ACM SIGMOD International Conference on the Management of Data*, pages 15–25, June 1989.

[175] E. K. Kolodner. *Recovery Using Virtual Memory.* Technical Report MIT/LCS/TR-404, MIT Laboratory for Computer Science, July 1987.

[176] D. Kotz and C. S. Ellis. Evaluation of concurrent pools. In *Proceedings of the International Conference on Distributed Computing Systems*, June 1989.

[177] D. Kranz, R. Halstead, and E. Mohr. Mul-t: a high-performance parallel Lisp. In *Proceedings of SIGPLAN '89, Symposium on Programming Languages Design and Implementation*, 1989.

[178] D. Kravets and J. K. Park. Selection and sorting in totally monotone arrays. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 494–502, 1990.

[179] J. T. Kuehn and B. J. Smith. The Horizon supercomputing system: architecture and software. In *Proceedings Supercomputing '88*, pages 28–34, IEEE, Orlando, FL, November 1988.

[180] Y. S. Kwong and D. Wood. A new method for concurrency in b-trees. *IEEE Transactions on Software Engineering*, SE-8(3):211–222, May 1982.

[181] R. Ladin. *A Method for Constructing Highly Available Services and a Technique for Distributed Garbage Collection.* PhD thesis, MIT Department of Electrical Engineering and Computer Science, May 1989.

[182] R. Ladin, B. Liskov, and L. Shrira. Lazy replication: exploiting the semantics of distributed services. In *Proceedings of the Ninth ACM Symposium on Principles of Distributed Computing*, August 1990. To appear.

[183] L. Lamport. How to make a multiprocessor that correctly executes multiprocess programs. *IEEE Transactions on Computers*, C-28:690–691, 1979.

[184] B. W. Lampson and H. E. Sturgis. *Crash Recovery in a Distributed Data Storage System.* Technical Report, Xerox Research Center, Palo Alto, CA, 1979.

[185] P. J. Landin. The mechanical evaluation of expressions. *Computer Journal*, 6(4):308–320, 1963.

[186] K. Lee. *Automatic Speech Recognition: The Development of the Sphinx System Appendix I.* Kluwer Academic Publishers, 1989.

[187] P. L. Lehman and S. B. Yao. Efficient locking for concurrent operations on b-trees. *ACM Transactions on Database Systems*, 6(4):650–670, December 1981.

[188] C. E. Leiserson and J. B. Saxe. *Retiming Synchronous Circuitry.* Technical Memo MIT/LCS/TM-372, MIT Laboratory for Computer Science, October 1988.

[189] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 1990. To appear.

[190] A. F. Lent. *A Lazy SECD Machine.* Bachelor's thesis, MIT Department of Electrical Engineering and Computer Science, January 1990. Supervised by A. R. Meyer.

[191] J. Leo. *Dynamic Process Creation in a Static Model.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[192] H. Leung and V. Zue. Some phonetic recognition experiments using artificial neural nets. In *Proceedings from ICASSP*, pages 422–425, New York 1988.

[193] H. Leung and V. Zue. Phonetic classification using multi-layer perceptrons. In *Proceedings from ICASSP*, pages 525–528, Albuquerque, NM, 1990.

[194] L. Levin. Pseudo-random bits: full security at last. Lectures given at Stanford, April; NSF-CBMS Circuit Complexity Workshop, University of Chicago, August; and Distinguished Lecturer Series, SUNY at Stony Brook, November, 1989.

[195] J. Lévy. Optimal reductions in the Lambda-calculus. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 160–191, Academic Press, 1980.

[196] B. Liskov. *Overview of the Argus Language and System.* Programming Methodology Group Memo 40, MIT Laboratory for Computer Science, February 1984.

[197] B. Liskov. Distributed computing in Argus. *Communications of the ACM*, 31(3):300–312, March 1988.

[198] B. Liskov. Distributed programming in Argus. *Communications of the ACM*, 31(3):300–312, March 1988.

[199] B. Liskov, T. Bloom, D. Gifford, R. Scheifler, and W. Weihl. Communication in the Mercury system. In *Proceedings of the 21st Annual Hawaii Conference on System Sciences*, pages 178–187, IEEE, January 1988.

[200] B. Liskov, D. Curtis, P. Johnson, and R. Scheifler. Implementation of Argus. In *Proceedings of the 11th Symposium on Operating Systems Principles*, ACM, Austin, TX, November 1987.

[201] B. Liskov, P. Johnson, and R. Scheifler. Implementation of Argus. In *Proceedings of the 11th Symposium on Operating Systems Principles*, November 1987.

[202] B. Liskov, R. Scheifler, E. Walker, and W. Weihl. *Orphan Detection.* Programming Methodology Group Memo 53, MIT Laboratory for Computer Science, February 1987.

[203] B. H. Liskov and J. V. Guttag. *Abstraction and Specification in Program Development.* MIT Press, 1986.

[204] W. J. Long. Medical diagnosis using a probabilistic causal network. *Applied Artificial Intelligence*, 2(2-3):367–384, 1989.

[205] W. J. Long, S. Naimi, M. G. Criscitiello, and R. Jayes. The development and use of a causal model for reasoning about heart failure. In P. Miller, editor, *Selected Topics in Medical Artificial Intelligence*, chapter 4, Springer-Verlag, 1988.

[206] N. Lynch and H. Attiya. Using mappings to prove timing properties. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, Quebec, Canada, August 1990. Expanded version: Technical Memo MIT/LCS/TM-412.b, MIT Laboratory for Computer Science, December 1989. Submitted for publication.

[207] N. Lynch, Y. Mansour, and A. Fekete. The data link layer: two impossibility results. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing*, pages 149–170, Toronto, Canada, August 1988. Also Technical Memo MIT/LCS/TM-355, MIT Laboratory for Computer Science, May 1988. Revised Technical Memo MIT/LCS/TM-355.b.

[208] N. Lynch and M. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, August 1987.

[209] N. Lynch and M. Tuttle. An introduction to input/output automata. *CWI-Quarterly*, 2(3), 1989. Also as Technical Memo MIT/LCS/TM-373, MIT Laboratory for Computer Science, November 1988.

[210] N. A. Lynch and E. W. Stark. A proof of the Khan principles for input/output automata. *Information and Computation*, 82(1):81–92, July 1989.

[211] D. Maier, J. Stein, A. Otis, and A. Purdy. Development of an object-oriented DBMS. In *Proceedings of the Object-oriented Programming Systems, Languages and Applications*, pages 472–482, November 1986.

[212] U. Manber. On maintaining dynamic information in a concurrent environment. *SIAM Journal on Computing*, 1986.

[213] T. Mann, A. Hisgen, and G. Swart. *An Algorithm for Data Replication.* Report 46, DEC Systems Research Center, Palo Alto, CA, June 1989.

[214] C. Manning. *Acore: An Actor Core Language, Reference Manual.* MPSG Apiary Design Note 7, MIT Artificial Intelligence Laboratory, August 1987.

[215] Y. Mansour. Learning *via* Fourier transform. 1990. Unpublished manuscript.

[216] Y. Mansour, N. Nisan, and P. Tiwari. The computational complexity of universal hashing. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 235–243, 1990.

[217] Y. Mansour and L. Schulman. Sorting on a ring of processors. *Journal of Algorithms*, 1989. Accepted for publication.

[218] Y. Mansour and L. Schulman. Sorting on a ring of processors. *Journal of Algorithms*, 1990. To appear.

[219] N. Margolus. N. Margolus. Parallel quantum computation. In *Complexity, Entropy, and the Physics of Information,* Addison-Wesley, 1990. To appear.

[220] N. Margolus and T. Toffoli. Cellular automata machines. In *Lattice Gas Methods for Partial Differential Equations*, pages 219–248, Addison-Wesley, 1990.

[221] S. Markowitz. *Central-server-based Orphan Detection for Argus.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, 1990.

[222] K. McCloghrie, J. R. Davin, and J. M. Galvin. *Experimental Definitions of Managed Objects for Administration of SNMP Communities.* Internet Draft, April 1990.

[223] A. R. Meyer. Semantical paradigms: notes for an invited lecture, with two appendices by S.S. Cosmadakis. In *Third Annual Symposium on Logic in Computer Science*, pages 236–255, IEEE, 1988.

[224] A. R. Meyer and M. A. Taitslin, editors. *Logic at Botic, '89: Proceedings of a Symposium on Logical Foundations of Computer Science.* Volume 363 of Lecture Notes in Computer Science, Springer-Verlag, 1989.

[225] R. A. Miller, H. E. Pople, Jr., and J. D. Myers. Internist-1, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307:468–476, 1982.

[226] D. Mills. Network time protocol (version 1) specification and implementation. DARPA-Internet Report RFC-1059. July 1988.

[227] R. Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

[228] Y. Mond and Y. Raz. Concurrency control in b+ trees using preparatory operations. In *Proceedings of the 11th International Conference on Very Large Data Bases*, pages 331–334, Stockholm, August 1985.

[229] T. Nguyen. *Performance Measurement of Orphan Detection in the Argus System.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1988.

[230] R. S. Nikhil. Practical Polymorphism. In *Lecture notes in Computer Science*, Volume 201, Springer-Verlag, September 1985.

[231] N. Nisan. Pseudorandom generators for space-bounded computation. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 204–212, 1990.

[232] N. Nisan and A. Wigderson. Hardness vs. randomness. In *29th Annual Symposium on Foundations of Computer Science*, pages 2–11, IEEE, 1988.

[233] C. Norton. Using separation algorithms in fixed dimension. 1990.

[234] C. Norton, S. Plotkin, and E. Tardos. *Using Separation Algorithms in Fixed Dimension.* Technical Report 866, School of Operations Research and Industrial Engineering, Cornell University, October 1989.

[235] B. M. Oki. *Viewstamped Replication for Highly Available Distributed Systems.* Technical Report MIT/LCS/TR-423, MIT Laboratory for Computer Science, 1988.

[236] B. M. Oki and B. Liskov. Viewstamped replication: a new primary copy method to support highly-available distributed systems. In *Proceedings of the Seventh ACM Symposium on Principles of Distributed Computing*, 1988.

[237] C. L. Ong. Fully abstract models of the lazy lambda calculus. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 368–376, IEEE, 1988.

[238] C. L. Ong. *The Lazy Lambda Calculus: An Investigation into the Foundations of Functional Programming.* PhD thesis, Imperial College, University of London, 1988.

[239] R. Ostrovsky. Efficient computation on oblivious RAMs. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 514–523, 1990.

[240] R. Ostrovsky, R. Venkatesan, and M. Yung. On the complexity of asymmetric games. December 1989. Unpublished manuscript.

[241] D. Pallett. Benchmark tests for darpa resource management database performance evaluations. In *Proceedings of ICASSP-89*. May 1989.

[242] P. Panagaden and E. W. Stark. Computations, residuals and the power of indeterminacy. In T. Lepistö and A. Salomaa, editors, *Automata, Languages and Programming: 15th International Colloquium*, pages 439–454, Volume 317 of Lecture Notes in Computer Science, Springer-Verlag, 1988.

[243] J. Paris. Voting with witnesses: a consistency scheme for replicated files. In *Proceedings of the Sixth International Conference on Distributed Computer Systems*, pages 606–612, IEEE, 1986.

[244] S. G. Pauker, G. A. Gorry, J. P. Kassirer, and W. B. Schwartz. Towards the simulation of clinical cognition: taking a present illness by computer. *American Journal of Medicine*, 60:981–996, 1976.

[245] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

[246] M. Phillips. Automatic discovery of acoustic measurements for phonetic classification. *Journal of Acoustical Society of America*, 84(S216), 1988.

[247] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[248] G. D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[249] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–257, 1977.

[250] C. N. Potts, D. B. Shmoys, and D. P. Williamson. Permutation vs. non-permutation flow shop schedules. Unpublished manuscript.

[251] V. N. Rao and V. Kumar. *Concurrent Access of Priority Queues*. Technical Report, Department of Computer Sciences, The University of Texas at Austin, 1988.

[252] J. A. Reggia and S. Tuhrim, editors. *Computer-assisted Medical Decision Making*. Springer-Verlag, 1985.

[253] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–96, 1987.

[254] J. G. Riecke. A complete and decidable proof system for call-by-value equalities (preliminary report). In *Automata, Languages and Programming: 17th International Colloquium*, Lecture Notes in Computer Science, Springer-Verlag, 1990. To appear.

[255] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, 1990. Winner of STOC '90 Best Student Paper Award.

[256] A. Rudich. *Finite Observation Semantics for Dataflow Networks.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990. Supervised by A.R. Meyer.

[257] T. A. Russ. Using hindsight in medical decision making. *Symposium on Computer Applications in Medical Care*, 1989. Finalist in student paper competition.

[258] Y. Sagiv. Concurrent operations on b-trees with overtaking. *Journal of Computer and System Sciences*, 33(2):275–296, October 1986.

[259] R. Sandberg, et al. Design and implementation of the Sun network filesystem. In *Proceedings of the Summer 1985 USENIX Conference*, pages 119–130, June 1985.

[260] R. E. Schapire. The strength of weak learnability. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 28–33, IEEE, 1989. Abstract also appeared in *Proceedings of the Second Annual Workshop on Computational Learning Theory*, July 1989. Also to appear in *Machine Learning.*

[261] R. E. Schapire. Pattern languages are not learnable. April 1990. Submitted for publication.

[262] E. J. Schwabe. Normal hypercube algorithms can be simulated on a butterfly with only constant slowdown. 1989. Submitted to *IPL.*

[263] E. J. Schwabe. On the computational equivalence of hypercube-derived networks. In *Proceedings of the Second Annual ACM Symposium on Parallel Algorithms and Architectures*, 1990.

[264] W. B. Schwartz, R. S. Patil, and P. Szolovits. Artificial intelligence in medicine: where do we stand? *New England Journal of Medicine*, 316:685–688, 1987.

[265] S. Seneff. A joint synchrony/mean-rate model of auditory speech processing. In *Proceedings of Journal of Phonetics*, 16:55–76, January 1988.

[266] S. Seneff. TINA: a probabilistic syntactic parser for speech understanding systems. In *Proceedings from Speech and Natural Language Workshop*, pages 168–178. Philadelphia, PA, 1989.

[267] A. Shamir. LP=pspace. Manuscript.

[268] G. Shaw and S. Zdonik. A query algebra for object-oriented databases. In *Proceedings of the Sixth International Conference on Data Engineering*, IEEE, February 1990.

[269] T. Shepard. *TCP Packet Trace Analysis.* Master's thesis, Massachusetts Institute of Technology, June 1990.

[270] D. B. Shmoys and D. P. Williamson. Analyzing the Held-Karp TSP bound: a monotonicity property with application. To appear in *Information Processing Letters.*

[271] M. Smith. *Nuclear Fusion through Dimensional Confinement.* MIT/LCS/TM-409, MIT Laboratory for Computer Science, August 1989.

[272] M. Smith. Representation of geometrical and topological quantities in cellular automata. *Physica D*, 1990. To appear.

[273] K. R. Sollins. *Plan for Internet Directory Services.* Request for Comments 1107, DDN Network Information Center, SRI International, July 1989.

[274] E. W. Stark. A simple generalization of Kahn's principle to indeterminate dataflow networks. December 1989. Unpublished manuscript.

[275] C. Stein. *Using cycles and scaling in parallel algorithms.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, August 1989. Also appears as MIT/LCS/TR-457.

[276] Sun Microsystems, Inc. *NFS: Network File System Protocol Specification.* Technical Report RFC 1094, Network Information Center, SRI International, March 1989.

[277] D. L. Tennenhouse. Layered Multiplexing Considered Harmful. In H. Rudin and R. Williamson, editors, *Protocols for High Speed Networks*, Elsevier Science Publishers, 1989.

[278] The Committee on Family and Work, P. Elias, chairman. Report of the MIT Committee on Family and Work: Analysis of Survey Findings (Preliminary). May 1990. To appear in *TechTalk.*

[279] The Committee on Family and Work, P. Elias, chairman. Report of the MIT Committee on Family and Work: Summary and Recommendations (Preliminary). May 1990. To appear in *TechTalk.*

[280] M. M. Thistle and B. J. Smith. A processor architecture for Horizon. In *Proceedings Supercomputing '88*, pages 35–41, IEEE, Orlando, FL, November 1988.

[281] T. Toffoli. *Analytical Mechanics from Statistics: $T = dS/dE$ Holds for Almost Any System.* Technical Report MIT/LCS/TM-407, MIT Laboratory for Computer Science, August 1989.

[282] T. Toffoli. Frontiers in computing. In *Information Processing,* Volume 1, North-Holland, 1989.

[283] T. Toffoli. How cheap can mechanics' first principles be? In *Complexity, Entropy, and the Physics of Information,* Addison-Wesley, 1990. To appear.

[284] T. Toffoli. T. Toffoli. Cellular automata. In *Encyclopedia of Physics*, VCH, 1990.

[285] T. Toffoli and N. Margolus. Invertible cellular automata: a review. *Physica D*, 1990. To appear.

[286] T. Toffoli and N. Margolus. Programmable matter. *Physica D.*, 1990. To appear.

[287] L. G. Valiant. A theory of the learnable. *Communications ACM*, 27(11):1134–1142, November 1984.

[288] E. W. Walker. *Orphan Detection in the Argus System.* Technical Report MIT/LCS/TR-326, MIT Laboratory for Computer Science, 1984.

[289] W. E. Weihl. *The Impact of Recovery on Concurrency Control.* MIT/LCS/TM-382.b, MIT Laboratory for Computer Science, August 1989. Supercedes MIT/LCS/TM-382.

[290] W. E. Weihl and P. Wang. Multi-version memory: software cache management for concurrent b-trees. Submitted for publication.

[291] J. Wein. Las Vegas RNC algorithms for unary weighted matching and T-join problems. Submitted to *Information Processing Letters.*

[292] J. Wein and S. Zenios. Massively parallel auction algorithms for the assignment problem. Submitted to the Third Symposium on the Frontiers of Massively Parallel Computation.

[293] D. Weinreb, N. Feinberg, D. Gerson, and C. Lamb. An object-oriented database system to support an integrated programming environment. 1988. Submitted for publication.

[294] S. M. Weiss, C. A. Kulikowski, S. Amarel, and A. Safir. A model-based method for computer-aided medical decision making. *Artificial Intelligence*, 11:145–172, 1978.

[295] D. P. Williamson. *Analysis of the Held-Karp Heuristic for the Traveling Salesman Problem.* Master's thesis, MIT Department of Electrical Engineering and Computer Science, May 1990.

[296] W. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13:591–606.

[297] A. Yonezawa, et al. Modelling and programming in an object-oriented concurrent language ABCL/1. In A. Yonezawa and M. Tokoro, editors, *Object-oriented Concurrent Programming*, MIT Press, 1987.

[298] V. L. Yu, B. G. Buchanan, E. H. Shortliffe, S. M. Wraith, R. Davis, A. C. Scott, and S. N. Cohen. An evaluation of the performance of a computer-based consultant. *Computer Programs in Biomedicine*, 9:95–102, 1979.

[299] S. Zdonik and P. Wegner. Language and methodology for object-oriented database environments. In *Proceedings of the 19th Annual Hawaiian Conference on Systems Science*, January 1986.

[300] L. Zhang. *A New Architecture for Packet Switching Network Protocols.* PhD thesis, Massachusetts Institute of Technology, August 1989.

[301] V. Zue, J. Glass, D. Goodine, M. Phillips, and S. Seneff. The summit speech recognition system: phonological modelling and lexical access. In *Proceedings from ICASSP*, pages 49–52, Albuquerque, NM, 1990.

[302] V. Zue, J. Glass, M. Phillips, and S. Seneff. Acoustic segmentation and phonetic classification in the summit system. In *Proceedings from ICASSP*, pages 389–392, Glasgow, Scotland, 1989.