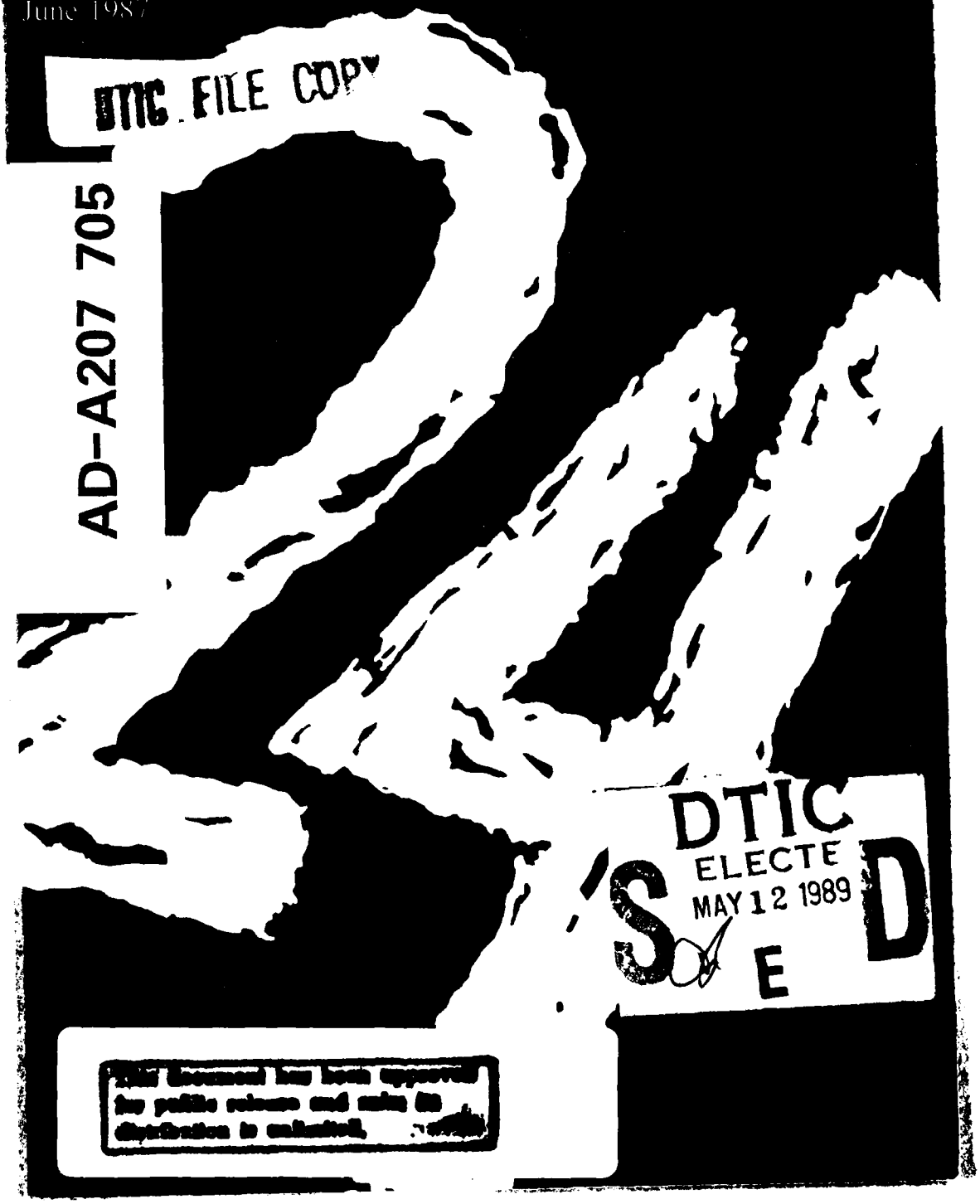
Massachusetts Institute of Technology

Laboratory for Computer Science Progress Report

July 1986-
June 1987



(4)

Massachusetts
Institute of
Technology

July 1986-
June 1987

Laboratory for Computer Science Progress Report

24

The work reported herein was carried out within the Laboratory for Computer Science, an MIT interdepartmental laboratory. During 1986-87 the principal financial support of the Laboratory has come from the Defense Advanced Research Projects Agency (DARPA). DARPA has been instrumental in supporting most of our research during the last 24 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;
- United States Army Research Office;
- MIT controlled IBM funds under an IBM/MIT joint study contract;

Other support of a generally smaller level has come from Apple, Hewlett-Packard, Siemens, and Giers.

TABLE OF CONTENTS

ADMINISTRATION

Academic Staff

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

Administrative Staff

P. Anderegg	Assistant Director, Administration
G. Brown	Facilities Officer
W. Fitzgerald	Fiscal Officer
M. Jones	Assistant Director, Operations
M. Sensale	Reading Room Supervisor
P. Vancini	Information Manager

Support Staff

L. Baxter
L. Cavallaro
S. Chung
R. Donahue
H. Dor
M. Gibson
A. Kekejian
S. Kim
J. Little
M. Rebelo
M. Weston

INTRODUCTION

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

Founded in 1963 as Project MAC (for Multiple Access Computer and Machine Aided Cognition), the Laboratory developed the Compatible Time Sharing System (CTSS), one of the first time shared systems in the world, and Multics -- an improved time shared system that introduced several new concepts. These two major developments stimulated research activities in the application of on-line computing to such diverse disciplines as engineering, architecture, mathematics, biology, medicine, and management. Since that time, the Laboratory's pursuits expanded, leading to pioneering research in knowledge-based (expert) systems, computer networks and public cryptography. Today, the Laboratory's research spans a broad front of activities, grouped into four major areas.

The first such area, entitled *Knowledge-Based Systems*, involves making programs more intelligent by capturing, representing, and using knowledge which is specific to a narrow problem domain. The Laboratory's Clinical Decision Making Group uses expert medical knowledge for computer-assisted diagnosis.

Research in the second and largest area, entitled *Machines, Languages, and Systems*, strives to discover and understand computing systems at both the hardware and software levels that open new application areas and/or effect sizable improvements in their ease of utilization and cost effectiveness. A large part of the Laboratory is involved in the architecture of large multiprocessor systems (which tackle a single task, e.g., speech understanding or weather analysis) by the Computation Structures, Real Time Systems, Information Mechanics, and Parallel Programming Research Groups. Continuing research includes the analysis and synthesis of languages and operating systems for use in large geographically distributed systems by the Distributed Computing Systems and Programming Methodology Groups. Finally, a key application involving distributed databases and community information is pursued by the Programming Systems Research Group.

The Laboratory's third principal area of research, entitled *Theory*, involves exploration and development of theoretical foundations in computer science. For example, the Theory of Computation Group strives to understand ultimate limits in space and time associated with various classes of algorithms; the semantics of programming languages, from both analytical and synthetic viewpoints; the logic of programs; the utility of randomness in computation; concurrent computation and the links between mathematics; and the privacy/authentication of computer-to-computer messages. Other examples of theoretical work involve the study of distributed systems by the Theory of Distributed Systems Research Group, and the development of effective algorithms for VLSI design.

The fourth area of research, entitled *Computers and People*, is concerned with the interrelationships between people and machines -- for example, the societal impact of computers, carried out by the Societal Implications Research Group.

Some of the year's research highlights were as follows:

The Multiprocessor Emulation Facility (Professor Arvind) and the Laboratory's Simulators were used to study the feasibility of a Tagged-Token Dataflow architecture. The results of these and related studies have led us to the design of a one gigaflop (peak performance) 256-processor machine which we are currently proposing (to DARPA) for construction. A comprehensive software system, Id World, has been documented and released. It translates functional programs from the language Id to representations that can be emulated on the above facility, simulated on an IBM 4381 computer, or executed on a dataflow machine. This programming environment will be used to study and assess additional application areas.

In the same area of multiprocessor systems, we are continuing our research on: Project L (Professor Stephen Ward) and CAM-7 (Dr. Tommaso Toffoli). L is a new model of computation characterized by: (1) a large collection of finite state machines and state representations with the property that programs written in object-oriented languages (with concurrency) can be efficiently compiled into L structures; and (2) L structures can be efficiently executed on a proposed hardware architecture associated with the L project. The CAM-7 architecture is a highly parallel cellular automata machine that further extends our previous architectures in this area. This machine will be able to update in one screen refresh interval, half-a-billion digital cells which are configured either as a cube or as a plane.

During the year, a good deal of progress was achieved in designing and defining the LCS Common System, now named Mercury, which is aimed at facilitating the composition of programs across different computational environments. For example, a program written in a Lisp machine environment should be able to call a subprogram written in C under a Unix environment.

During the same period we were successful in funding (through Siemens) and formally initiating research in learning systems. This program strives to develop theories and machines that can learn from their environments, and not from their programmers. We believe that the rapid technological progress in architectures and VLSI calls for a re-examination of learning theories and approaches. In the theoretical area, we have developed (Professor Baruch Awerbuch) an approach that can make network protocols run reliably on unreliable networks, and we have established (Professor F. Thomas Leighton) some important results about broadcast networks.

Two of the Laboratory's past accomplishments have achieved higher levels of standardization. First, the NuBus standard (Professor Ward) was adopted by Apple Computer for their new open architecture systems, starting with the Apple II GS.

Second, the X-Window Standard (Mr. Robert Scheifler) for graphics has been adopted by the MIT School of Engineering's Project Athena and some 20 companies which are urging us to form a consortium for the continued development of X.

During 1986-1987, the Laboratory has continued its successful Distinguished Lecturer Series with presentations by Carnegie-Mellon University's Hans J. Berliner; Cal Tech's Charles Seitz; Robert E. Kahn, President of the Corporation for National Research Initiatives; David J. Kuck, Director of the Center for Supercomputing Development at the University of Illinois; Leslie B. Lamport from DEC's Systems Research Center; and Nils J. Nilsson, Chairman of the Computer Science Department at Stanford University.

The Laboratory had several personnel changes over the past year including: the return of Mr. Albert Vezza as Associate Director, the arrival of Dr. Thomas Greene as Director of Computing Resource Services; and the departures of Dr. Irene Grief to Lotus Corporation, Dr. Gerard Vichniac to MIT's Plasma Fusion Center, and Professor Richard Zippel to Symbolics, Inc.

Our Laboratory consisted of 320 members -- 45 faculty and academic research staff, 30 visitors and visiting faculty, 60 professional and support staff, 110 graduate and 75 undergraduate students -- organized into 14 research groups. Laboratory research during 1986-87 was funded by 12 governmental and industrial organizations, of which the Defense Advanced Research Projects Agency of the Department of Defense provided over half of the total research funds. Also during the same period the Laboratory employed 23 undergraduates through the "Hacker Heaven" project which strives to identify promising potential researchers in computer science.

Technical results of our research in 1986-87 were disseminated through publications in the technical literature, through Technical Reports (TR 367-TR 394), and through Technical Memoranda (TM 296-TM 332).

Michael L. Dertouzos,
Director

CLINICAL DECISION MAKING

Academic Staff

P. Szolovits, Group Leader
R. Patil

Collaborating Investigators

M. Criscitiello, M.D., Tufts-New England Medical Center Hospital
M. Eckman, M.D., Tufts-New England Medical Center Hospital
C. Fleming, M.D., Tufts-New England Medical Center Hospital
R. Friedman, M.D., University Hospital, Boston University
W. Hardy, Ph.D., University Hospital, Boston University
R. Jayes, M.D., Tufts-New England Medical Center Hospital
J. Kassirer, M.D., Tufts-New England Medical Center Hospital
M. Klein, M.D., University Hospital, Boston University
B. Kuipers, Ph.D., University of Texas at Austin
A. Moskowitz, 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
L. Widman, M.D., Case Western Reserve University Medical School

Research Staff

S. Marshall
W. Long

Graduate Students

D. Fogg
H. Goldberger
R. Granville
Y. Jang
I. Kohane
P. Koton
S. Novick
T. Russ
E. Sacks
M. Wellman
T. Wu
A. Yeh

Undergraduate Students

T. Chow
I. Haimowitz
T-Y. Leong

Support Staff

R. Hegg

Visitors

N. Guarino
B. Rasmussen

1. INTRODUCTION

With our colleagues at Tufts/New England Medical Center (TNEMC), we in the Clinical Decision Making Group have continued to be involved in a variety of research activities. Our two main projects are entitled *An Artificial Intelligence Approach to Clinical Decision Making* and *A Program for the Management of Heart Failure.* The main goals of our research in the former project continue to reflect the directions outlined in last year's report. Chiefly, they include (1) the creation of a coherent knowledge-representation formalism for medical knowledge that can serve as the basis for research into advanced medical reasoning techniques, and (2) the application of artificial intelligence methods to improve the effectiveness of the clinical use of decision analysis. In the last year, we have continued to advance the work in both of these major directions. In the Heart Failure Project, Dr. William Long and his colleagues continue work on developing a methodology and tools for assisting the physician in reasoning about the diagnosis and management of heart failure.

2. AN ARTIFICIAL INTELLIGENCE APPROACH TO CLINICAL DECISION MAKING

2.1. Summary of Research Progress

Our hypothesis in formulating one major aspect of our current research effort has been that the next major steps forward in the capabilities of medical expert systems would come from the successful integration of a large number of independently-developed capabilities into a single coherent system. In particular, our aim is to combine into a single overall system methods for reasoning about:

- the causal nature of disease hypotheses,
- the physiological mechanisms underlying health and illness,
- the temporal evolution of natural processes and intervention plans, and
- the partially-known likelihoods and utilities of possible future developments.

Our initial approach has been to continue to perform research on each of these problems independently, because each requires further advances, and at the same time to work toward their integration by investigating the utility of general-purpose knowledge representation languages (in particular, NIKL) to express everything required by these investigations in a single overall language. Thus far, it appears that our individual attempts to improve the state of the art concerning each of the above problems is far outstripping our ability to find suitable representations for their concepts within a single linguistic framework.

The other major focus of our efforts, on improving the use of decision analysis in

clinical settings, has produced a more balanced record of progress on the conceptual and the implementational front. Our foundational work focuses on continuing our analyses of human performance in planning for patient management and on the development of new artificial intelligence methods of planning under uncertainty. The more pragmatic focus of this research is our analysis of errors actually made by human practitioners in the conduct of their decision analyses and the creation of a Decision-Tree Critiquer (DTC) program that will help to critique and improve decision analyses, based on our growing understanding of how to evaluate such analyses.

The following subsections describe the major components of these efforts in more detail and outline our related plans for the coming year. Subsections 2--10 focus on our efforts to develop and integrate various AI methods for medical reasoning, and Sections 11 and 12 concentrate on our work on applying AI techniques to the support of decision analysis.

2.2. Integrating Various Techniques within a Single Knowledge Representation Framework

The most difficult technical task we have undertaken has been to try to develop a common underlying knowledge representation formalism that is adequate to state all the knowledge and strategies to be used in any of the various reasoning methods that we wish to use in our medical programs. At the outset of this research, we had anticipated that recent advances in knowledge representation research would provide us a basis for building such a formalism. It was clear that the myriad problems of defining a sophisticated medical epistemology and then reducing it to the implementational requirements of a particular knowledge representation language would be formidable, but we assumed that a focused, two-part process would enable us to perform the task, at least to a reasonable degree of success. First, we planned to develop an informal, English-like representation, with simple and flexible rules, that would provide an initial target for our efforts to encode the knowledge of our medical areas of focus (in cardiology and nephrology). Second, we planned to develop effective means of stating in NIKL (our choice of knowledge representation technology) whatever was encoded in our intermediate language. This translation process, though it should ideally be automated, was planned to involve significant researcher intervention.

The next section describes out efforts toward the first of these steps, in the domain of coronary artery disease. The second step has been a source of more serious difficulties, because we have been unable to find good ways of expressing in NIKL all of the concepts needed. This point is perhaps worth a short elaboration, because of course in a descriptive language in which new terms can be introduced at will, it is always possible to find some syntax into which to map anything. The crucial issue in using a knowledge representation language is that the knowledge, once represented, should work in a natural and effective way with the facilities built into the representation scheme. In fact, the principal reason advanced for the utility of knowledge representation

languages compared with predicate calculus is that a language like NIKL provides certain often-useful statements and operations in a particularly concise and efficient form. Inheritance via taxonomies and the computation of subsumption and exclusion relations are such specially-provided mechanisms in NIKL.

Profs. Ramesh Patil and Peter Szolovits and one of our students, Ira Haimowitz, are now preparing a paper charting our experience in converting the knowledge base of the ABEL program for acid-base and electrolyte diagnosis into NIKL. Our principal difficulties have been twofold: (1) many concepts and relationships that had been concisely represented in XLMS (ABEL's original implementation language) expand by a factor of five or so in NIKL, and, more seriously, (2) the definitional character of NIKL makes it virtually impossible to model accurately the notion of "typical, but with possible exceptions" that seems much more appropriate to medical knowledge.

Profs. Szolovits, Patil and Dr. William Schwartz have also prepared two reviews of AI-based medical reasoning methods for the community of physicians. One paper [19] explores the limitations of current methods and the potential utility of intermediate-term programs that fall short of the ultimate ambitions of AI researchers but nevertheless provide useful functionality based on current techniques. The second [21] presents a technical exploration of how various diagnostic problem-solving methods of proven value can be integrated to form the basis of a new generation of diagnostic tools.

Our plans for the coming year on this important problem of integration of knowledge are to pursue some revisions of the NIKL representation language (as proposed by a student, Yeona Jang), to consider alternative representation formalisms (such as FRAPPE, see below), and to adapt our ideas on diagnostic strategies and our experiences with existing representation efforts such as those described in the next two sections to suggest new integrative ideas.

2.3. Flexible Knowledge Representation for Multiple Reasoning Tasks

Drs. William Long and Robert Jayes have been working on the problem of representing knowledge in a form flexible enough to support multiple reasoning tasks and to represent the different kinds of information actually present in real medical domains. That is, it should be possible to represent the knowledge that the knowledge base designer has without having to reformulate it to meet some criteria for a specific reasoning paradigm. Furthermore, the relations between concepts should be explicitly stated so that new reasoning methods can determine the properties they need from the existing statements. The knowledge base should contain all of the information that might reasonably be considered pertinent to the domain.

To explore the issues involved in representing medical knowledge and the problems with meeting these desiderata, we have examined the problem of representing the knowledge about exercise stress testing for coronary artery disease (CAD) patients. CAD testing includes a wide variety of kinds of knowledge and reasoning, making it a

fertile domain. The knowledge of the physician includes the anatomy of the heart and coronary vessels, the physiological requirements and abnormalities involved in myocardial ischemia, the relationship between the electrocardiogram (ECG) and the heart electrical activity being measured, the relationships between the actions and effects involved in an exercise protocol, as well as the specific rules for interpreting the test within the more general physiological understanding.

Our approach has been to write the domain knowledge as simple English statements, keeping the description as concise and simple as possible. We made a number of simplifications to the medical knowledge when it seemed that more completeness would not bring out new issues. An attempt was made to keep the entire description under two pages. This description then formed the basis for what had to be represented in a knowledge representation language. The next step was to transform the descriptions into statements in Lisp syntax representing the content of each sentence and adding additional statements to define as many of the concepts as possible in terms of more primitive concepts. In this way we were able to build an initial set of primitive concepts and identify the ways in which concepts need to be combined as statements of the representation.

We have identified a number of kinds of knowledge that need to be in the knowledge base to capture the content of the CAD description. These include subsumption relations, logical assertions, probabilistic relations, cause-effect relations from constraints to tendencies, correspondences, empirical rules, and descriptions of events, actions, and plans. The subsumption relations carry important properties at each concept level. For example, the right coronary artery (RCA) supplies part of the inferior myocardium, but it is a coronary artery and therefore can be restricted in coronary artery disease. It is also an artery, so it supplies oxygen to tissues by being a kind of conduit. A logical relation can assert the general location of the RCA, but the circumflex coronary artery supplies the anterior myocardium in some patients and the inferior myocardium in others. Such relations require a probabilistic relation. The causal relations range from "exercise increases heart rate proportionately" to "when ischemia exceeds a threshold, the patient usually experiences angina." The correspondences in the domain can be quite complicated, in particular, the correspondence between electrical leads of the ECG and the location of the electrical phenomena in the heart. In addition to the causal information about CAD causing coronary restriction, causing (with exercise) ischemia, causing ST-depression on ECG, it is necessary to include the empirically determined rules such as an ST-depression greater than 3mm at maximum heart rate implying that significant CAD is very likely (i.e., CAD sufficient to require surgery). To describe the protocol for the exercise stress test -- with its progression of exercise levels and success and failure termination conditions and associated measurements -- requires a language for plans and events.

This effort has also helped to identify some of the important distinctions among concepts of the same class. For example, there are several kinds of causation represented in this domain. CAD itself is a progressive disease. The practical

implication of *progressive causation* is that the disease state is always the same or worse than it was at any known point in the past unless there has been a corrective intervention. From this arises the distinction between corrective interventions such as coronary artery bypass surgery and non-corrective therapies such as propranolol. A second type of causation is the ischemia causing angina, which is only true when the ischemia is over a certain threshold. Thus, the angina marks the ischemia.

The Lisp form of the knowledge base only uses logical connectives and a simple pattern language. For example, the relationship between oxygen deficit and ischemia is as follows:

```
;When oxygen supply to a region is less than oxygen requirement,
; the region has ischemia in proportion to the difference.
(=> (> (requirement oxygen myocardium)
       (supply-of oxygen ?(area-of myocardium)))
    (proportional-cause
      (- (requirement oxygen myocardium)
         (supply-of oxygen ?(area-of myocardium)))
      (ischemia ?(area-of myocardium))))
```

Our intention is to develop and maintain this knowledge base in multiple forms and use it for testing new knowledge representation ideas and testing the modules using the knowledge base. It contains more kinds of knowledge than we can currently represent in the NIKL constructs that we have developed so far, but as we extend the NIKL representation we will develop and maintain a version in NIKL. We have also used this knowledge base to examine the potential for using the FRAPPE representation system developed by Rich and Feldman at the MIT AI Laboratory. However, our initial conclusion is that the representational requirements are not well matched to the facilities of their system. The knowledge base will certainly help to push the development of the assertional component of the general knowledge representation language for the other projects.

2.4. Representation for Physiologic and Anatomic Knowledge

One focus of our interest has been the representation of physiological and anatomical knowledge for use in diagnostic reasoning and therapy planning. In the past few months, Dr. Oksana Senyk has collaborated with Prof. Patil and Dr. Frank Sonnenberg, an internist at NEMC, in a project exploring the explicit representation of anatomical and physiological knowledge for a program which reasons about the pathophysiological and anatomical aspects underlying the syndrome of jaundice. We have used the NIKL language to build the knowledge base, in which several "pure" taxonomic hierarchies [21], each built around one aspect of medical knowledge, such as anatomy, physiology, and etiology, are interconnected by links representing relations among the nodes. Each node in the taxonomies can have multiple parents. The resulting lattice structure provides disease descriptions which may be accessed along several dimensions, e.g.

anatomical involvement of the disease or etiology of the disease. This organization encourages a principled development of the knowledge base and permits one to capture various subtleties of the domain [12]. It also lends itself to reasoning about clinical scenarios from a variety of approaches; for example, in a situation where the anatomical involvement of a disease is known in considerable detail, while other data are sparse, this information can be used to perform a first cut at the formulation of a diagnostic hypothesis [13].

In this project, we have been particularly interested in the role of anatomical reasoning in medicine. Thus, we have implemented an experimental program which combines anatomical reasoning with the causal knowledge inherent in the knowledge base's model of the pathophysiological processes involved in hyperbilirubinemia. For example, the program can reason about the association between pancreatic carcinoma and jaundice. It finds the mechanical ("anatomical") cause(s) of jaundice, namely, common bile duct obstruction. Next it explores the mechanisms of duct obstruction in general, among which it finds that duct obstruction can be caused by external compression, which in turn is caused by a space-occupying lesion. It notes that this particular cause may be relevant to pancreatic carcinoma, which is a space-occupying lesion (it is also a type of disease, having two parents in the taxonomy). Finally, it investigates the anatomical relations of the relevant anatomical structures and finds that the head of the pancreas is anterior to the common bile duct.

We feel we have explored some interesting territory in the intersection between anatomical and physiological reasoning. The problem of jaundice, with its various mechanical etiologies, is particularly well suited to such an investigation. We plan now to adapt some of the techniques we have developed to the domains of renal and cardiovascular disease.

2.5. Hierarchical Reasoning

Echoing one of the principal themes of our work on integrating problem-solving methods, Thomas Wu has been investigating three ideas, each of which focuses on the use of hierarchies in representation and reasoning.

In an area crucial to medical reasoning because of the inherent uncertainty of medical data and inference, Mr. Wu has developed a formalism to allow the propagation of uncertainty between hierarchies of hypotheses. Current schemes for managing uncertainty allow the propagation of uncertainty only between singleton sets of hypotheses, not hierarchies of hypotheses. Nevertheless, medical concepts exist at various levels of abstraction, necessitating a hierarchical organization.

To address this issue, he has developed a framework for making uncertain inferences between hierarchical hypothesis spaces which is based upon the semantic relationship between inferences drawn from a subset of hypotheses and those drawn from its individual component hypotheses. A consistent set of inferential links from one frame

of discernment to another can therefore be derived. This framework provides one approach to combining hierarchical and associational knowledge.

A second area of his involvement concerns the cognitive psychology of medical expertise. Together with Prof. Peter Politser of the Harvard School of Public Health, Mr. Wu has recently completed a review of the various studies on clinical reasoning. Medicine has been an interesting area for psychologists to study because of its rich and large structure of knowledge. At the same time, though, this complexity has led to seemingly conflicting results in the literature. These results have been cast in a general framework for understanding and comparing results on the medical problem-solving process. This framework suggests that medical expert systems could be improved in several ways, such as increasing the precision and organization of medical knowledge, using causality and abstraction in inference, and having both a focused approach and a global control of the reasoning process.

During the next year, Mr. Wu will begin to design a novel architecture for medical expert systems. This design will be based upon the construction of medical classification trees that change dynamically during the course of a diagnosis. Current classification diagnosis systems depend upon a single hierarchy based on one attribute, such as cause or set-inclusion. His proposal would allow a large number of different hierarchies to be built at any given time in the diagnosis. The problem space can therefore be partitioned flexibly during the course of the diagnosis. This architecture would allow for the representation of the many functional interrelationships inherent in a semantically rich domain such as medicine.

2.6. Reasoning about Continuous Dynamic Systems

One of the important advances in artificial intelligence research has been the creation of new, qualitative, techniques for studying the behavior of complex physical systems whose structure and laws of operation may only be known approximately. This sort of capability must also underlie sophisticated medical reasoning programs that are able to trace their understanding of the behavior of a complex disorder to the pathophysiologic derangements that are its etiology. Beginning with the study of physiological systems, Elisha Sacks has developed a series of more and more powerful programs that analyze the behavior of a physical system that is described by its equations, even when those equations include unknown parameters.

Earlier results worked only for linear systems, which represent a useful though very limited set of mechanisms. During the last year, Mr. Sacks has designed and begun to implement a program called PLR that analyzes continuous *nonlinear* dynamic systems. It simplifies a nonlinear system by approximating the equations that describe it with piecewise linear ones. It derives phase diagrams for the piecewise system by combining the local behaviors on the linear regions. The current program can analyze common nonlinear systems including pendulums, van der Pol oscillators and servo-mechanisms.

This work is the subject of Mr. Sacks' doctoral thesis, which should be completed in the coming year. He has also presented a paper at the *AAAI* qualitative physics workshop in Urbana/Champagne in May [18], and two additional papers will be presented at AAAI in Seattle this July [16][17]. A fourth is being presented at the Second International Conference on Applications of AI in Engineering in Boston this August.

2.7. Human Performance on Planning Tasks in Medical Decision Making

Dr. Alan Moskowitz and several colleagues have been conducting an empirically based study to gain insight into the methods employed by physicians in planning patient management and to build on these insights in developing the planning methodology and terminology required for an artificial intelligence system to perform the complex task of planning clinical management.

Clinical Case Selection

During the last year we collected the clinical accounts of several seriously ill patients, which we summarized for presentation to clinicians to observe their problem-solving behavior. Because we are interested in the intermediary stages of reasoning, the clinical cases we selected for presentation needed to be unique enough that our physician-subjects would not have pat responses to them and yet familiar enough that their solution did not rely on obscure information or virtuoso performance. In order to elicit planning behavior, we chose patients with multiple active medical problems with potentially conflicting goals. We selected two such cases for conducting the interview experiments (see Figures 2-1 and 2-2).

Subject Selection

We selected for interview eight board certified, academic internists, who are recognized as highly competent physicians and who were unfamiliar with the nature of the research being conducted. We deliberately selected internists without subspecialty training in the medical concerns of the two clinical cases because they would be less likely to have managed patients with precisely the same constellations of findings and would, therefore, be more likely to exhibit the use of first principles in their problem solving, rather than offering precompiled solutions to the problems. We plan to look at the behavior of domain subspecialists at a later time.

The subjects of the study volunteered their participation and we obtained their informed consent.

Format of the Interview Experiments

The interview was comprised of two parts: a thinking aloud experiment, in which the subjects responded to the written case material in a spontaneous fashion, uninterrupted by the interviewers; and a direct questioning segment, in which the interviewer asked

The patient is a 73 year old man with a history of adult onset diabetes mellitus, hypertension, peptic ulcer disease (inactive), gout (inactive) and mild chronic renal insufficiency (creatinine 1.9 mg/dl) who presented to his physician with a headache and was discovered to have an enlarged liver and elevated hematocrit (Hct 65%). He was admitted to the hospital for work-up and treatment of his hepatomegaly, headache and erythrocytosis.

During hospitalization red blood cell labeling revealed a 50% increase in red cell mass with normal serum B12 and leukocyte alkaline phosphatase levels. An abdominal ultrasound was obtained and showed a normal sized spleen, but revealed an 8 x 10 x 11 cm singular mass in the right lobe of the liver and an 5.8 cm abdominal aortic aneurysm. The ultrasound was followed by an abdominal CT scan which showed a singular mass extending from the right lobe of the liver to the caudate lobe, enhancing slightly with injection of contrast material. CT scan directed percutaneous needle biopsy of the mass was reported to be bloody and complicated by a vasovagal reaction. He recovered from this reaction without further incident. The biopsy specimen showed clusters of cells with malignant morphologies that were consistent with hepatocellular carcinoma and his alphafetoprotein level was markedly elevated, 74,500 ng/ml (normal <2.5 ng/ml). The patient was phlebotomized 3 units of packed red blood cells and his hematocrit dropped to 42.2%.[1] [27].

Figure 2-1: Case I

A 72 year old man with a long history of hypertension and coronary artery disease presented for abdominal evaluation of abdominal cramping was discovered to have an abdominal aortic aneurysm, which measured 4.6 x 5.6 x 9.3 on ultrasound. His abdominal cramping was self limited but on a follow-up visit he related that he was having regular episodes of exertion associated angina pectoris. He underwent an exercise tolerance test during which he developed 3mm of st segment depression in the inferior leads, at a heart rate of 140 and a BP of 170/80.

His medical history is also notable for a cerebral ischemic attack from which he recovered fully. Other medical history includes peptic ulcer disease and osteoarthritis.

Figure 2-2: Case II

probing questions to clarify the subject's earlier responses or to elicit responses to topics the subject did not address before. Both the thinking aloud segment and the direct questioning segment were tape recorded and transcribed verbatim. The transcripts of

[1] We published a formal decision analysis of this case in the *Western Journal of Medicine*.

the interviews serve as the data for our analysis. The transcripts do not contain specific information about the subject, to preserve their anonymity.

Analysis of the Data

We are analyzing the data using "script analysis", an analytic formalism intended to reveal the goal structure of the problem-solving process or explanation strategy. The technique involves reading each line of the transcript and keeping track of the status of the decision or argument by noting such items as: the nature of the point or argument being made, the trigger for the line of reasoning, the knowledge presumed, the choices made, or any dangling threads of the argument. The product of this analysis is an outline or "script" of the different stages of the reasoning process, identifying operations being performed (e.g., summarizing, making choices, defining) and the content of the knowledge being used. We demonstrate this methodology in two manuscripts that were recently accepted for publication [7][10]. The former paper reports the results of our preliminary work on planning in medical management.

Our work to date appears to confirm our preliminary observations that there is considerable individual variation in the overall reasoning processes of physicians, especially at points where hypotheses are considered and risks are assessed. The management planning processes that we observed appear to follow an opportunistic model, in which the different steps of the planning process are not strictly ordered and their sequence appears to be determined by the structure of the problem, the information happens to be available, and the inferences that happen to be activated.

Goals for Next Year

Our plans for the coming year are to finish the analysis of previously collected protocols and collect additional protocols to substantiate our findings. We plan to use additional case materials in other medical domains to see whether our findings hold for problem solving in these other areas. We also plan to employ clinicians at different levels of training to see how this affects the planning processes that we observe.

2.8. Planning under Uncertainty

Work on planning under uncertainty is aimed at generalizing the traditional AI planning framework to handle partially satisfiable goals and uncertain effects of actions, two requirements of medical therapy application domains. Decision theory provides a criterion for choice in such cases, but does not address the problem of assembling complex strategies from distributed specifications of individual actions or plan fragments. Yet this is the form that knowledge is likely to take in comprehensive medical knowledge bases of the future.

Michael Wellman, as part of his doctoral research program, has designed and begun to implement a new planning system that will address this problem. The heart of our approach is to introduce a dominance prover into the basic planning engine. The

dominance prover tries to prune away large classes of potential plans by finding classes that are guaranteed to contain plans at least as good. Dominance relations are recorded in a lattice of the plan classes generated. New plan classes are classified in the plan lattice according to the subsumption relation, thereby ensuring that subclasses of classes already determined to be dominated are not searched. The plan lattice supports a constraint-posting approach where plans can be explored at multiple levels of abstraction. A further description of this planning methodology is provided in a recent paper [23].

To design a planner based on this approach, we need to choose several other representations and mechanisms. In particular, we need a representation for the effects of actions and relations between events. Our knowledge representation describes influences among actions and events in terms of a qualitative probability formalism we have developed for this purpose [25][26]. This representation has significant modularity advantages over purely numerical schemes because, while the exact joint probability distribution over a set of events varies substantially with context, the direction of influences are more reliably taken as constant. The conclusions that may be drawn are weaker, but are often sufficient to derive useful dominance results for the planner.

At this time the design for SUDO-Planner is substantially completed and the implementation is well underway. For a fuller description of the project, see [24].

Over the coming year we will complete the implementation of SUDO-Planner and test it on a range of overlapping examples. Extensions to the theoretical framework will include an investigation of representations for qualitative synergy and the development of reasonable focus mechanisms for limiting the search space.

2.9. Case-Based Reasoning in a Domain with Causal Models

Most current expert systems have the following characteristics:

1) They rely on knowledge compiled from experts.

2) Their performance does not improve with experience, unlike human problem solvers. When faced with the same difficult problem twice in succession, they work just as hard the second time.

3) They can not modify their knowledge as a result of their problem solving history.

The difficulty of extracting the expert's compiled knowledge, and the problems of depending only on compiled knowledge for problem solving have been well documented (e.g., by Waterman and Davis). The advantages of having a problem solver that could learn from experience has also been addressed elsewhere (e.g., in Michalski's writings).

Phyllis Koton is implementing a medical therapy and follow-up program for heart disease (CASEY) which will incorporate the experiential component seen in human problem solving. It draws on previous work of Michalski, Winston and Kolodner on learning, reasoning by analogy, case-based reasoning, and memory organization. CASEY is designed in response to the three characteristics described above. First, instead of having to extract compiled knowledge from experts, the system itself is able to create new pieces of knowledge as a result of generalizations from the cases it has seen. Second, the system will be able to use diagnostic and treatment solutions from previously seen patients to aid in the treatment of subsequent patients. Finally, some knowledge (e.g., strategic and treatment, but not basic physiology) contained in the system will be revised in response to feedback on its success in solving clinical problems.[2]

The basic idea of the program is the same as other programs which use analogical reasoning: compare the current problem to a set of previously seen problems stored in memory. If current problem is "sufficiently similar" to a previous problem, the program can use the previous solution as a "shortcut" to solve the current problem. However, if the program does not find a sufficiently similar problem, then the current problem must be solved from scratch (presumably a more difficult process). The system also adds the current case to its memory of problems it has seen.

In order for the program to know if the transferred solution worked, the stored precedent solution must include an expectation of the results of using it, and the program must be given follow-up information so that it can evaluate the results. If the actual results don't meet the expectation, the program will go back and analyze the current case and the precedent to see why its expectation was not met, which might result in a more restricted applicability for the precedent. If the expected results were achieved, the current case is added as one more piece of evidence in favor of using the transferred solution. After solving a number of related problems, the program can make generalizations from the experience in solving the earlier problems. Generalization consists of placing cases that have features in common in (conceptually) adjacent locations in memory, and extracting the features in common for use in making predictions about subsequent similar cases.

The proposed system differs from previous work in two ways. The first difference is that CASEY makes use of causal explanations of illness. The causal explanation from a precedent case may be transferable to a new case presenting with similar symptoms, thus allowing the new case to be understood quickly. Comparing the causal explanations of a precedent case and a new case allows the program to make causally-

[2]The philosophical assumption here is that there are two kinds of knowledge in the program's knowledge base: associational knowledge (as, for instance, between a symptom set and a disease) which can be modified by experience, and basic science or "first principles" knowledge, which cannot.

grounded predictions[3] about the new case based on the precedent.

The second difference is that while all previous case-based reasoners have worked to restrict the set of features that are used to store and recall cases from memory, the system will index every given feature of the case. The reason for doing this is that the importance of a feature can not always be predicted in advance. If a feature is determined to be unimportant *a priori*, and a request is later received to match a case using that feature as an index, the system must construct an alternate method of searching for cases having that feature. This is time-consuming and not always successful. By using all features as indexes the system increases its chances of retrieving relevant and useful cases. The frequency with which an index is used will determine the strength of the recall of that index.

The proposed system also offers a method for combining "shallow" and "deep" reasoning in a program. By comparing a new problem with previously seen problems, the program has information about the types of problem-solving methods that have been successful on similar problems in the past. If the new problem matches a case that has been solved in the past, the solution that is transferred is essentially a "shallow" solution, because the reasoner can avoid the complicated causal reasoning that the original solution required (only verifying it, as will be discussed below). As several similar cases are solved, the information about them and their solution paths is generalized and becomes new "shallow" knowledge. If no matching case is found, the problem solver must use "deep" reasoning to solve the problem, using basic knowledge.

2.10. Temporal Reasoning

During the past year, we have completed two projects investigating different aspects of the problems of temporal reasoning. Mr. Thomas Russ has made available the initial implementation of a temporal control structure, a set of program utilities for the Symbolics 3600 that helps in the implementation of programs that reason with data that change over time [15]. The chief facility provided by this package is a separation between two components of a program's capabilities for reasoning about time. One component allows the expression of decision and interpretation rules in a time-independent, "instantaneous" fashion, supporting the convention that decisions are made instantaneously based at all times on the available current information. The second component keeps track of the dependencies of current data on past reports and arranges to re-execute those instantaneous decision rules whose data prerequisites have changed, either because of the advance of time or because of the correction of previously-incorrect historically-reported data. We plan to incorporate this model into our common representation.

The other project, done as part of Dr. Isaac Kohane's M.D. studies at Boston

[3](As well as empirically-based predictions.)

University and done in collaboration with us, investigates a knowledge representation for reasoning with imprecisely-known temporal intervals. Its principal contribution is to suggest practically-useful ways to limit the complexity of the obvious book-keeping necessary to calculate the temporal constraints imposed by a number of independent partially-known time relationships. Dr. Kohane's study emphasizes the utility of those temporal relations that correspond to the causal structure of hypotheses, in preference to other temporal relations. This work has been reported in a paper presented at SCAMC [5], and the thesis is available as a technical report [6].

2.11. Improving the Clinical Application of Decision Analysis

Our colleagues at Tufts/NEMC are continuing to develop an expanded terminology for decision trees to allow the representation of events in more detail than would be engendered by simply *decision*, *chance*, and *outcome* nodes. We have been tracking the clinical decision making consultation service for the past year, cataloging the errors that appear in decision trees and the sequences of errors made as such models evolve over time. Although most of our computing development has been in the Symbolics environment, we have begun a parallel effort on PC class machines, in both PROLOG and Lisp. We have also continued to improve our microcomputer **DecisionMaker** environment, distributing the first major update in three years. That system is now being used in over 100 settings outside of Tufts. Its representation scheme has become the *de facto* standard of the field.

The utility acquisition project has been focusing on three related domains -- prenatal diagnosis, contraceptive counseling and counseling about changes in behavior, and seeking information about antibody status with respect to HIV infection. We hope to have a form of these modules, integrated with the DecisionMaker tree analysis environment, running very shortly. That module will then be available in a public display through which we can collect data about its utility. As a portion of this project, some of the conclusions of one of our analyses concerning HIV infection were recently published in the *New England Journal of Medicine* and have provoked widespread discussion about screening of low prevalence populations.

DecisionMaker

We have completed a major upgrade of the DecisionMaker decision analysis system, now being distributed as Version 6.0. It was demonstrated at MedInfo 86 and will be demonstrated at SCAMC 87. The representation of decision trees has been enhanced and a full tree editing capability has been added. We have also been exploring the implications of the distribution of risk profile results and have added the automatic calculation of variance (the second moment) to expected utility calculations. The cost-effectiveness section now automatically calculates the frontier of non-dominated cost effective strategies.

We have almost completed a script player for recording and replaying (with the option

of including variable input) sessions for either education or for developing simple variations on fixed decision trees. We are beginning to develop the ability to recall arbitrary text tagged to various objects within a decision tree problem.

We have also begun to include the concepts being developed in the Symbolics 3600 environment into our PC system. A representation for different sub-types of decision nodes has been specified in a manner compatible with existing software for summarizing decision trees. We have made our first foray into symbolic evaluation of complex decision trees, both in the 3600 and PC environments.

Utility Analysis

We have developed a scheme for acquiring patient utility information in both lottery and time trade-off modes using a PC system with a graphic display. We have applied this technique to problems in utility acquisition for prenatal diagnosis and contraceptive counseling. We are developing a module to provide counseling about HIV screening. The modules have been developed in a combination of Turbo Pascal (the native language for DecisionMaker) and Turbo Prolog. The systems uses several scenarios and tradeoffs to check answers for internal consistency and feeds inconsistencies back to the user for resolution.

We have arranged for this module to be displayed in the Boston Computer Museum, where it will be used by the general public. The information we collect from that exposure should provide information about the durability of this approach.

2.12. Automatic Critiquing of Errors in Decision Analyses

As we have stated, one of the major foci of the group's interests in medical artificial intelligence centers around the automatic critiquing of errors made by decision analysts in structuring and analyzing medical decision trees. During the past year, the group has developed a catalogue of errors made by novice medical decision analysts and generated a taxonomy of errors based on the catalogue. These products have been used to develop an existence demonstration of a program, the Decision Tree Critiquer DTC. This is a decision-tree building software environment that we developed in Lisp. The work has been done in collaboration with the Division of Clinical Decision Making at Tufts-New England Medical Center (TNEMC).

The material for the catalogue of errors was collected at TNEMC. New Fellows arrived in the Division on July 1, 1986 and participated in the patient consultation service. Many of the consults present medical problems that are amenable to decision analysis. Active consults are presented formally semi-weekly and are discussed informally as needed. These discussions give ample opportunity to observe the Fellows' learning process and to collect errors from their draft analyses. We have also developed a taxonomy of errors in medical decision analysis based on the material in the error catalogue. The work will be generalized into a taxonomy that is domain independent.

The error catalogue has been used to develop the critiquing capability of the DTC. This program was originally written by Michael Wellman, a graduate student in our Group. The program, as originally written, allowed the user to develop and modify decision trees with a minimum of effort and to analyze these trees and choose an optimal strategy based on the maximization of expected utility. Further development of the DTC during the past year has resulted in a version of the program that can recognize and critique structural errors in decision trees.

Our critiquing methodology is based on the hypothesis that many essential features of a decision problem can be understood from a decision tree without massive amounts of domain-specific knowledge, such as medical knowledge. Our tree representation elucidates these features by providing a taxonomy of node categories to specify distinctions among nodes beyond the usual types of decision, chance, and terminal nodes. Test decisions and treatment decisions, for example, are two kinds of decision nodes. Chance nodes are classified as test results, treatment efficacies, physiologic states, and complications. The nodes are the primitives for building an abstract structural description of a decision problem. The critiquer analyzes the description encoded in the decision tree to validate the structural accuracy of the proposed model.

The DTC's response to a decision tree is governed by critiquing rules expressed within our representational language. For example, the program recognizes and questions test decisions that are modeled without discernible costs (e.g., side effects or money) because the decision to order a truly cost-less test need not be modeled explicitly. The program also questions the analyst about strategies in which actions are conditioned on unobservable physiologic states rather than observed test results and strategies that include explicit test decisions but in which no differential action is taken for different test results.

Recently, work has commenced on developing the capability of DTC to revise probabilities that appear in a decision tree using an event space that represents all possible events on which probabilities in the tree can be conditioned. Probability revisions are a frequent source of errors in medical decision analysis because the calculations can be very complex, especially when dealing with non-exclusive sets of events or conditioning events that have many possible outcomes. The availability of an automatic transformation capability may prevent some errors in probability calculations.

We plan to present two abstracts describing aspects of this work at the Society for Medical Decision Making meeting this October [2][20]. During the past year, we have also developed a classification for decision problems which was applied to the literature of existing decision analyses. That study was published in the *Annals of Internal Medicine* [4].

Plans for Future Work

The development of the error catalogue will continue as long as it yields a significant

amount of new information. Since new Fellows will arrive in the Division of Clinical Decision Making in early July, we anticipate that the next few months will be the most important. Shortly, we will begin to generalize the catalogue of errors and the error taxonomy to be domain independent.

We are continuing to work on the development of the DTC. Current rules for critiquing have been analyzed for common structure in order to begin to develop a rule processor. Next, we plan to develop rules to critique more complex, non-structural errors. Finally, during the current year we hope to begin to develop an automatic explanation capability for the DTC.

2.13. Miscellaneous

In addition to the range of work reported above, various members of the CDM Group have been involved in studies and reviews that have led to publications. Prof. Patil co-directed an MIT conference on the future of artificial intelligence research, which led to publication of a book on that topic [3]. Prof. Szolovits contributed a study of tools for expert systems construction to that volume [22], and Prof. Patil also contributed a chapter outlining his thoughts on how medical diagnosis systems are evolving [14]. Dr. Robert Kunstaetter won second place in the 1986 student paper competition at SCAMC for his paper on using a knowledge-based model of physiologic reasoning in medical education [8].

3. A PROGRAM FOR THE MANAGEMENT OF HEART FAILURE

The objective of this research is to develop a computer-based methodology for assisting the physician in reasoning about the diagnosis and management of patients, particularly when causality, physiological relationships, and changes over time are important to the reasoning process. The disease context for developing the methodology is the diagnosis and management of cardiovascular diseases that have as part of their manifestation the syndrome of heart failure, thus requiring a thorough understanding of the hemodynamic and physiological relationships of the cardiovascular system for effective management. The approach is to develop a system that represents what is known about the patient in a form that can be displayed graphically and can be manipulated by the physician to explore hypotheses and investigate their implications. During the first two years of the project, we developed the basic structure of the program including the logical support for relationships between physiological states, a methodology for projecting changes given interventions, a method of graphical entry of the input data, and a simple scheme for interpretation of the data. During this past year we have continued to work on the reasoning of the therapy management module as we are expanding the physiology module to cover the diseases that lead to heart failure. The next two years of the project will emphasize developing the facilities necessary to make the program a useful tool in the hospital setting, but will also include further development of the reasoning mechanisms and physiological model of the program.

During the past year, we worked on all of the modules of the program to some extent, but most of the effort was focused on the mechanisms and relations in the physiology module needed for reasoning about patient management. First, however, we will discuss the progress in the development of the input and diagnosis modules including developing outlines of the primary pathophysiological mechanisms and manifestations of the pertinent disease entities for use in diagnostic reasoning, adapting a mechanism for probabilistic reasoning about input data as evidence for physiological states, and completing the reimplementation of the earlier PDP-10 based angina program on the Symbolics computer.

In addition to the work on the program, there are two other closely related efforts being conducted by graduate students Thomas Russ and Alexander Yeh. Mr. Russ is developing a general control structure for supporting reasoning with time dependent data and Mr. Yeh is developing methods for dealing with the uncertain values and relations in predicting therapy effects in the Heart Failure Program. These projects will be discussed later in this section.

3.1. Module Analyses

Each of the diseases that can cause heart failure has pathophysiological consequences that are useful for confirming or rejecting the disease as a hypothesis and for tracking the progress of the patient management. As a starting point for extending the input and diagnostic modules to cover these diseases, we have made outlines of the physiological mechanisms and the important manifestations of each disease. The physiological mechanisms are specified in terms of the chains of causation that lead to the abnormal states encountered in the diseases. These causal chains are specified at the same level of abstraction as the existing model. The observations included in the outlines are those that are most useful in tracking the changes in the disease during therapy and those that are useful for differential diagnoses. These outlines will be used to extend the diagnostic module and input module to cover the diseases as we extend the physiological model to cover these diseases.

One of the weaknesses of the earlier version of the program is the mechanism by which inputs are used as evidence for determining the state of the physiological parameters. The present mechanism attaches procedures to each parameter that evaluate the pertinent inputs. In a search for a more principled mechanism for evaluating evidence, one that also takes into account evidence from other parameters, we have begun exploring causal interpretations of Bayesian probabilities as developed by Judea Pearl [11] and others. This year we produced an initial implementation of Pearl's scheme for the Heart Failure Program. One problem with this scheme is its restriction of no loops and requirement that rejoins be handled specially in the chains of relationships. In our implementation this problem was avoided by only drawing conclusions when there are no loops or rejoins and withdrawing conclusions if they develop. We have not yet incorporated this mechanism into the whole program, so we

do not yet know whether this restriction will be seriously limiting. However, this mechanism should provide the appropriate functionality for local reasoning about evidence.

The angina program, developed on the PDP-10, includes the initial implementations of all of the modules of the program from input to therapy recommendation, but did not include any of the graphics that are now possible on the Lisp machine. The physiological model in the program focused on the determinants of myocardial oxygen supply -- the considerations most important for managing a patient with angina and heart failure. Over the past year we have incorporated all of the important functionality of this program into the Heart Failure Program on the Lisp machine. Thus, we have a shell that includes a simple implementation of all of the modules to which we can add the more sophisticated modules we are developing now. With the angina physiological model as the program model, this provides an effective demonstration of the functioning of the whole program.

The therapy module has been the primary focus of our efforts over the past year. Two years ago we began the development of a mechanism for reasoning about the expected changes in parameters resulting from application of therapies. The mechanism is based on signal flow analysis and computes the changes in the parameters from steady state to steady state. The advantage of this mechanism over other means of solving the equations is that we end up with a record of the pathways of influence on the parameters and their relative contribution. This record constitutes an explanation of the change. Last year we applied this mechanism to a model with qualitative relations on the links between parameters. This worked well in our early tests in which we compared the actions of drugs in the normal patient to the model predictions, but we had considerable difficulty extending the model to account for the behavior of mitral stenosis, the first valve disease we considered. Since most of the parameters have known *quantitative* relationships to other parameters and the computational mechanism supports quantitative reasoning, we decided to explore the feasibility of a quantitative model. To do so required two significant additions to the reasoning mechanism as well as a new model. The two additions are the handling of integrated parameters and compensating for non-linearities.

The shape of the physiological model is intended to conform to the usual notion of causality in the cardiovascular system (basically in the direction of blood flow), even though the equations are actually constraint equations and therefore directionless. Since one normally considers that the input pressures produce cardiac output on both the left and right sides, there is a problem in representing the equality of left and right outputs in steady state. Physiologically these outputs are the same because blood volume shifts between circulations until they are the same. To capture this idea, it is necessary to have levels or integrated variables to represent the volume in a particular circulation.

We extended the reasoning mechanism to handle integrated relationships by making the observation that in steady state, the derivative of an integrated parameter is zero.

This provides the additional constraint needed to determine the level of the parameter. With this extension the procedure for determining the change in parameters resulting from an intervention requires two steps: first determine the levels of the integrated parameters necessary for their derivatives to be zero, then use these values plus the original change to determine the final values of all of the parameters.

The second extension to the reasoning was to handle non-linearities in the relationships between parameters. A non-linearity in the relationship between two parameters implies that the gain between the parameters varies over the course of the change. The algorithm uses the initial value of the gain to determine the changes, but that is not always adequate. Our solution has been to adjust all of the gains to be the average gain over the range of the change and iterate until the final values conform to the constraint equations. Although this approach has theoretical limitations, in practice it seems to converge rapidly to the solution.

With this mechanism in place, it was possible to convert our physiological model to a quantitative form. The equations we used came from several sources. Some of the equations were obvious, such as the relationship between cardiac output, vascular resistance, and pressures. Others were determined from data in the literature, such as the relation between heart rate and systolic time. The rest were borrowed from existing models such as Coleman's Human Program [1], including the relation between blood pressure and vagal stimulation. The basic set of equations we have been using is in Figure 2-3. The parameters included in the model are those that are likely to be measured in the patient or reported in the literature plus the parameters needed to account for the actions of the usual cardiovascular agents. So far we have focused on short term effects, so mechanisms such as those causing changes to the blood volume have not yet been included in the model.

To this initial model we have added some of the common cardiovascular therapies. (New therapies are being added as we investigate experiments that include them.) Each therapy is added as a set of changes to the parameters that the therapy effects directly. The proportions are determined by comparing the model predictions to published results. In some cases the model of the drug had to be extended to include lesser direct effects to account for the published changes. The nodes and links for beta blockers, hydralazine, and nitroglycerin are included in the graphic presentation of the model in Figure 2-4.

Also, to the initial model we have started to add relationships to account for the different diseases causing heart failure. So far, we have added three to the model, aortic stenosis, aortic regurgitation, and mitral stenosis. These were added by including equations relating the pressure drops and regurgitant volumes to the degree of value dysfunction from the original equations developed by Gorlin. As we investigate other diseases we will include more such relations in the model.

```
venous_vol          =  blood_vol − pulm_vol −
                       dead_vol / (.7 + .3 × symp_stim) × venous_constr
; right output
rap                 =  5.7 × venous_vol − resist_venous_ret × co
rvedp               =  rap
rv_output           =  rv_compl × rv_emptying × 1.375 × (rvedp + 4.0)
pa_press            =  lap + pulm_vascul_resist × rv_output
; left output
lap                 =  if (< pulm_vol .7) then 36.0 × pulm_vol − 10.2
                       else 125.0 × pulm_vol − 72.5
lvedp               =  lap
lv_output           =  if (< lvedp 8.0) then .75 × lvedp × lv_compl × lv_emptying
                       else (.5 × lvedp + 2.0) × lv_compl × lv_emptying
co                  =  lv_output
; blood pressure and sympathetic response
blood_press         =  co × svr
systol_press        =  blood_press
symp_stim           =  -.03 × (blood_press − blood_press_base) + 1.0
vagal_stim          =  .033 × (blood_press − blood_press_base) + 1.0
; heart rate
heart_rate          =  base_heart_rate + 37.5 × symp_stim −
                       (if (< vagal_stim 1.0) then 23.0 × vagal_stim
                       else-if (< vagal_stim 2.0) then 17.0 × vagal_stim + 6.0
                       else 10.0 × vagal_stim + 20.0)
syst_time           =  17.3 + .075 × heart_rate
dias_time           =  42.7 − .075 × heart_rate
; left systolic function
inotrop             =  symp_stim
lv_systolic_funct   =  inotrop × lv_systolic_funct_base
lv_emptying         =  lv_systolic_funct ×
                       (if (< systol_press 100) then 1.0
                       else 1.4 + -.004 × systol_press)
; right systolic function
rv_systolic_funct   =  inotrop × rv_systolic_funct_base
rv_emptying         =  rv_systolic_funct ×
                       (if (< pa_press 20) then 1.0
                       else 1.1 − .005 × pa_press)
; vascular resistance
svr                 =  svr_base + symp_stim × svr_response
resist_venous_ret   =  svr × .025 + .9
pulm_vascul_resist  =  pvr_k1 + rv_output × pvr_k2
; pulmonary volume
pulm_vol            =  integral (rv_output − co)
```

Figure 2-3: Basic Physiological Model Equations

3.2. Model Validation

Our method of validating the model has been to compare published data to the model predictions. We are taking the data from papers in the literature in which patients with one of the diseases was given one or more of the therapies or was exercised and the values of the hemodynamic variables are reported before and after the intervention. As long as the hemodynamic data is fairly complete, the initial values of the model parameters can be computed or estimated and the model can simulate the patient. Our efforts thus far in validating the model have been very fruitful. The paper we presented at the Computers in Cardiology Conference in October [9] provides more detail of the modeling of the valve disorders and exhibits the initial results of the comparison. The model proved sufficient to account for the average behavior reported in each of the five papers studied. With only two or three minor exceptions, the predictions were within the errors of the mean reported in the papers for all of the parameters included in the model, once appropriate distributions of direct effect were determined for the therapies and the exercise the patients experienced.

Figure 2-4 shows an example of the prediction of the model for a case in which a patient having mitral stenosis is exercised (actually we used the average data for 10 patients). The data in the paper included the heart rate, cardiac output, left ventricular systolic pressure, left ventricular end diastolic pressure, pulmonary artery pressure, and pulmonary wedge pressure. The rest of the parameters were computed from these to run the model by making assumptions consistent with the state of the patient. Since right atrial pressure was not reported and these patients were not reported to have any signs of right sided failure, the right atrial pressure was assumed to be zero. The degree of mitral stenosis shown in the diagram (zero is normal) is computed from the pressure drop across the mitral value. Parameters that have no initial value at the bottom of the node are qualitative parts of the model intended to indicate the source of influences on parameters such as myocardial ischemia and blood volume. For this example, the amount of exercise applied was chosen to produce the cardiac output of 8.0 that was reported in the paper. Under these conditions the predictions for the other reported parameters were all within the errors of the mean. In the figure there is also highlighting along two pathways from exercise to cardiac output. This illustrates the capability of the program to identify the most important pathways for producing the change. This highlighting serves as a type of explanation of the predicted changes and points the user to the areas of the model that would be most sensitive to changes.

Since then we have continued the investigation, both to include other disease situations and to determine the extent of variation in response to interventions among patients. The next paper we studied investigated patients with congestive cardiomyopathy given dobutamine. The model could be made to reproduce the average behavior by adjusting the direct effects of dobutamine, but among the individual patients for whom adequate data was included in the paper, there was considerable variation. We are currently considering possible explanations for this variation

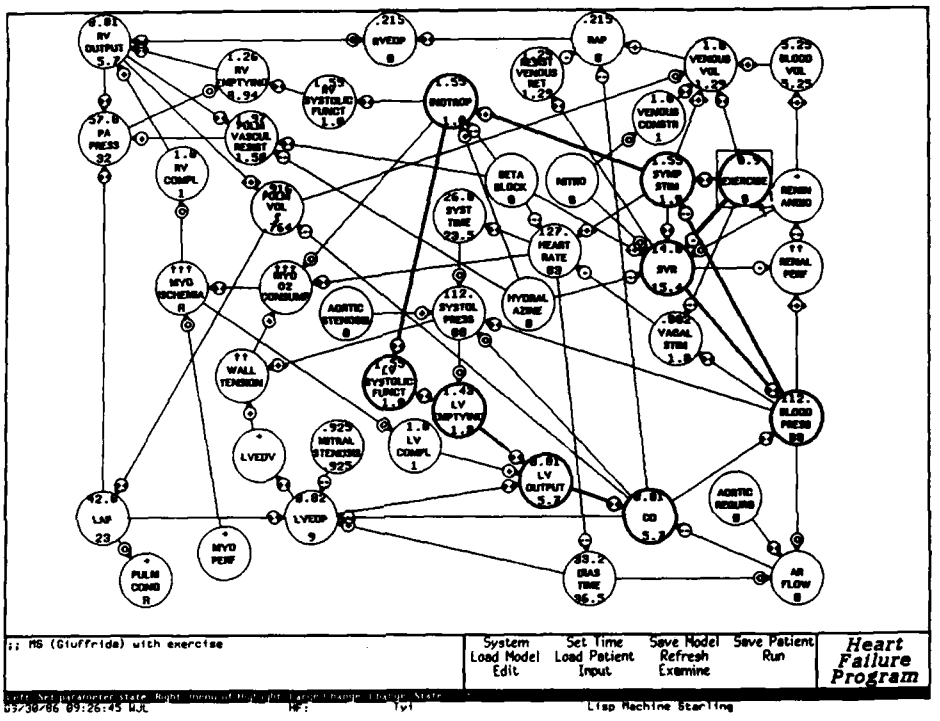


Figure 2-4: Prediction for Mitral Stenosis with Exercise

including the possibility of variable mitral regurgitation. In any case, we need to determine the extent and types of variation among patients with all of the diseases to be able to give realistic predictions of the likely outcomes of a therapy.

One of the advantages of the method we have chosen is the possibility of extending it to reason about ranges of possible behavior. As we gather data on patient responses under various conditions, we will gain a better understanding of the ways in which patient response varies. Once we have this information, we plan to add to the mechanism facilities for including ranges of possible gain along links and ranges of direct effect from interventions as needed to anticipate the usual range of patient response.

Overall, the results of comparing the model predictions to literature have been very encouraging. Not only do the appropriate qualitative changes take place, but the quantitative predictions have had enough resolution to force us to include some of the minor direct effects of some of the therapies to account for the reported data. For example, in modeling propranolol our first model of the effect was to say that it decreased inotropic effect and heart rate in the proportions that sympathetic stimulation increased them. From the predictions, it was clear that that did not account for the increased systemic vascular resistance reported in the patients. We investigated further and discovered that in decreasing the beta sympathetic tone in the vascular system, the drug leaves the alpha sympathetic effect unopposed and therefore the systemic vascular resistance actually does increase. When this and a similar effect on the pulmonary vascular resistance were taken into consideration, the predictions of the model corresponded to the actual behavior.

Thus over the past year we have made progress on all of the modules of the program, but most of the effort has gone into improvements to the therapy module. We feel that the new quantitative capabilities of the therapy prediction mechanism will be beneficial not only for predicting therapy results but also for using the results of therapy to differentiate among contending hypotheses.

3.3. A Control Structure for Building Time-Dependent Data

Thomas Russ has been continuing the development of a control structure for writing expert systems that use time dependent data. The ability to have multiple sessions and track changes in a patient's state is crucial for patient management applications. The essential simplification that the control structure provides is the decomposition of decision-making over time into a series of "snapshots," each of which is a static framework which provides the context for more traditional reasoning strategies.

In support of this static framework simplification, it is necessary to develop a state description from raw input data. The construction of state descriptions from the data is an example of abstraction and problem structuring. One concept to receive special attention in the past year was the idea of "persistence" of laboratory or observational values. After a certain period of time, previous results are considered to be too old to have a bearing on current decision-making.

As an example of the usefulness of this approach, a Master's thesis project was completed by Steven Novick using the control structure that is being developed. The thesis describes a program to reason about the underlying disease state of patients in a cardiac intensive care unit suffering from ventricular arrhythmias.

3.4. Dealing with Model Uncertainty

Alexander Yeh is building a model of the human cardiovascular system which tries to answer questions about cardiac patients relating to the possible effects (main and side-) of a certain therapy on a particular patient (with a particular set of conditions). The model is based on the work of Dr. Long and others, and includes:

- About 40 to 50 variables, including blood pressure *BP*, cardiac output *CO*, systemic vascular resistance *SVR*, etc.
- Constraints (equations) between the variables. An example is: $BP = CO*SVR$.

The constraints are arranged in an 'intuitive' way, with causality in terms of blood flow being a primary determiner of what is intuitive. As an example, the constraint above indicates that *CO* and *SVR* together 'causes' *BP*'s value.

The model's properties include the following:

1) The constraints are nonlinear in the variables that can vary in value. An example is the *BP* equation above. Besides multiplication, exponentiation to a constant power is also present. In other words, algebraic operations are used.

2) Because of stability and other properties, the modeled system can be viewed as moving from one temporary equilibrium to another.

3) Like many models in biology, this one has many feedback loops. The loops affect many variables, and different loops affect a given variable in different ways (one loop may increase a variable's value, while another is decreasing it). The loops are comparable in the size of the changes they induce (one loop is not negligible compared to another).

A major problem with using the model is that the input data can vary greatly from patient to patient, and even within a given patient from one moment to another when (s)he is at rest. Examples include the following:

- In one experiment, the ten patients had pulmonary arterial pressure measurements ranging from 14 to 53 mm of mercury (a factor of over 3).
- For one patient, the pulmonary wedge pressure went from 25mm in one control run to 34mm in another, a gain of over one third.

Current methods for dealing with this varying data all have problems: Bounding algorithms (find upper and lower bounds), qualitative math systems, and order of magnitude math systems all tend to give results that are too loose, general, or ambiguous to be useful. Monte-Carlo methods give results in a form that is hard to use,

because one never knows when an important case has been randomly left out, and because finally, complications can arise when the assumed input probability density is not quite right. Unless an equation has only linear operations and/or multiplications between uncorrelated variables, using it directly to find the means has little formal justification (probabilistic or otherwise), and often gives bad results. Assuming the model is quasi-linear about an operating point and using Gaussian densities sometimes results in skewed densities being called Gaussian. However, this last method does seem to have some utility.

To improve on the situation, Mr. Yeh proposes two possible methods. The first is a variation using upper and lower bounds. Normally, one uses known variable upper and lower bounds and any known (in)equalities to find the upper and lower bounds of the variables of interest. The bounds on the latter often tend to be too loose to be useful. Instead of this, the method lets the known bounds and (in)equalities not hold all the time, but just at least $n\%$ (say for example, 50%) of the time. Given the proper bounding algorithm, the resulting bounds on variables of interest will also hold at least $n\%$ of the time. Since the known bounds and inequality(s) that may only hold 50% of the time will be tighter than those that hold all the time, the resulting bounds should be too. One then repeats the process with other values of n (like 25, 75, etc.). The resulting sets of variable bounds will give a bound on what the joint probability density of the variables looks like. Combining this with some methods that can merge sets of bounds and (in)equalities that only hold some of the time, causes a possibly useful method to emerge.

The second method deals with estimating the probability densities (or bounds on densities) of operations on random variables. The results are exact when operations only involve one of the following two types of operations:

1) Linear combinations (multiplication by a constant, addition, and subtraction), or

2) exponentiation by a constant, multiplication, and division.

If both types of operations are involved, some approximations are done to convert between two types of densities. The random variables need to have 'Gaussian-like' densities (or perhaps a combination of such).

Once these density bounds or approximations are found, one can use them to indicate:

1) The odds of some variable being below or above some critical value (above or below which pain, damage and/or improvement may result).

2) How much a variable can vary in value, which may help determine if it is possible or desirable to try to measure the variable more accurately.

During the past year, Mr. Yeh has been writing code and running some preliminary

examples based on data from a study on 10 patients with mitral stenosis. The results from those examples look promising. However, more testing needs to be done, including:

- Rerun the examples with better models of the example data densities.
- Try the cardiovascular model on more examples. Especially useful would be examples where the data variation is not between patients, but within the measurements of one patient.

As well as running more examples this coming year, we plan to develop some needed criteria for gauging density shape (type) from data. For example, we might ask on what grounds one would say that a density is approximately Gaussian, lognormal, or something else.

Finally, a number of methods or their variations have yet to be tested, and the various methods have to be combined into a coherent system.

3.5. Directions for the Coming Year

There will be two major foci for our research in the heart failure domain in the coming year. The first is to develop the interface and support functions to make the program a useful tool for the medical users. The second is the further development of the model and reasoning.

All of our testing of the program thus far has been by using cases from the literature or carefully gathered cases from the CICU. These cases were then entered into the program by the system developers, who also assessed the results. To make the program a useful tool for the medical community, we need to extend the functionality by including a data base for saving and retrieving the cases that are in progress, extending the reasoning to make appropriate inferences during multiple sessions with the same patient, enhancing the input module to provide appropriate summarization of the case material, improving the graphic presentation to tailor the display to include only those relationships that are pertinent to the patient being considered, and making the other improvements necessary to eliminate any gaps we discover as medical users gain experience with the system. The objective is to have a system that will support the functions needed by the user to carry on an extended dialog with the machine, possibly over several sessions. Thus, the program needs to remember what went on before and incorporate the new information into the existing information. Since we need the user's opinions to find the kinds of improvements that would most help the user, we will include comment facilities in the program so that the user can add comments when they arise in using the program.

Development of the model and reasoning will proceed in two directions. First, we need to continue the addition of disease entities to the model until we have all of the

commonly occurring diseases that lead to heart failure. This includes mitral regurgitation, the various forms of cardiomyopathy, constrictive pericarditis and tamponade, pulmonary disease, and tricuspid regurgitation. Each of these presents an additional challenge for modeling the dynamics under the therapeutic interventions, but our experience with the diseases we have modeled thus far should enable us to extend the model without major difficulty. Our method for developing the models for these diseases will be to use the data available in the medical literature to identify the physiological effects of the disease that account for any changes in hemodynamic response. In the end, we will have a single physiological model that can account for all of the commonly occurring pathophysiological states of the cardiovascular system. As a result, there should be no difficulty in accounting for multiple disorders or comparing the implications of competing hypotheses.

The knowledge bases used by the input and diagnostic modules also need to be extended to cover the diseases being added to the physiological model. The first step in doing this was to develop the disease outlines, which were done this past year. Now we need to include the physiological concepts in those outlines in the causal relations used by the diagnostic module as well as include the manifestations in the input frames. The interpretation of the input will also require incorporating the new probabilistic reasoning algorithm into the evidence evaluation both from the input and between parameter states where causality is possible rather than definite. This will give us the experience with the probabilistic methods needed to determine whether they are adequate.

References

1. Coleman, T. G. and J.E. Randall. "HUMAN: A Comprehensive Physiological Model." *The Physiologist*, 26, (1983), 15-21.

2. Eckman, M.H., M.P Wellman, S.L. Marshall, C. Fleming, F.A. Sonnenberg and S.G. Pauker. "A Knowledge-Based System for Critiquing Medical Decision Trees." Abstract to be presented at the Annual Meeting of the Society for Medical Decision Making, October 1987.

3. Crimson, W.E.L and R.S. Patil (eds.). AI in the 1980's and Beyond: an MIT Survey. Cambridge, MA, MIT Press, 1987.

4. Kassirer, J.P., A.J. Moskowitz, J. Lau and S.G. Pauker. "Decision Analysis: A Progress Report." *Annals of Internal Medicine*, (1987), 106, 275-291.

5. Kohane, I.S. "Temporal Reasoning in Medical Expert Systems." *MEDINFO 86: Proceedings of the Fifth Conference on Medical Informatics*. 1986, 170-174.

6. Kohane, I.S. "Temporal Reasoning in Medical Expert Systems." MIT/LCS/TR-389, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

7. Kuipers, B., A.J. Moskowitz and J.P. Kassirer. "Decisions in Medicine: Representation and Structure." *Cognitive Science*, to appear.

8. Kunstaetter, R. "Intelligent Physiologic Modeling: An Application of Knowledge-Based Systems Technology to Medical Education." *Computer Methods and Programs in Biomedicine*, 24, (1987), 213-225.

9. Long, W.J., S. Naimi, M. G. Criscitiello and R. Jayes. "Using a Physiological Model for Prediction of Therapy Effects in Heart Disease." *Proceedings of Computers in Cardiology Conference*, Boston, MA, October 7-10, 1986.

10. Moskowitz, A.J., B. Kuipers and J.P. Kassirer. "Dealing With Uncertainty, Risk and Tradeoffs: A Cognitive Science Approach." *Annals of Internal Medicine*, to appear.

11. Pearl, J. "Fusion, Propagation, and Structuring in Bayesian Networks." Technical Report CSD-850022, University of California at Los Angeles, Los Angeles, CA, June 1985.

12. Patil, R.S. "Review of Causal Reasoning in Medical Diagnosis." *Proceedings of the Tenth Annual Symposium on Computer Applications in Medical Care*, IEEE, 1986, 11-15.

13. Patil, R.S. and O. Senyk. "Efficient Structuring of Composite Causal Hypotheses in Medical Diagnosis." *Proceedings of the Eleventh Annual Symposium in Computer Applications in Medical Care*, IEEE, 1987 - forthcoming.

14. Patil, R.S. "A Case Study on Evolution of System Building Expertise: Medical Diagnosis." In AI In The 1980's and Beyond: An MIT Survey. Grimson, W.E.L and R.S. Patil (eds.), Cambridge, MA, MIT Press, 1987.

15. Russ, T.A. "Temporal Control Structure Reference Manual." MIT/LCS/TR-331, MIT Laboratory for Computer Science, Cambridge, MA, June 1987.

16. Sacks, E.P. "Hierarchical Reasoning About Inequalities." *Proceedings of the National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, 1987, 649-654.

17. Sacks, E.P. "Piecewise Linear Reasoning." *Proceedings of the National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, 1987, 655-659.

18. Sacks, E.P. "Qualitative Sketching." in *Knowledge Based Expert Systems for Engineering: Classification, Education, and Control*, Boston, MA, Computational Mechanics Publications, 1987, 1-13.

19. Schwartz, W.B., R.S. Patil and P. Szolovits. "Artificial Intelligence in Medicine: Where Do We Stand?" Sounding Board article in *New England Journal of Medicine*, 316, (March 12, 1987), 685-688.

20. Sonnenberg, F.A., M.P. Wellman, M.H. Eckman, C. Fleming, S.L. Marshall and S.G. Pauker. "Automatic Probability Revision in Decision Trees Using Event Spaces." Abstract presented at the annual meeting of the Society for Medical Decision Making, October 1987.

21. Szolovits, P., R.S. Patil and W.B. Schwartz, "Artificial Intelligence in Medical Diagnosis." *Annals of Internal Medicine*, to appear.

22. Szolovits, P. "Expert Tools Systems Techniques and: Past, Present and Future." AI in the 1980s and Beyond: An MIT Survey. W.E.L. Grimson and R.S. Patil (eds.), Cambridge, MA, MIT Press.

23. M, P.Dominance. "and Subsumption 1987 in Constraint-Posting Planning." *Proceedings of the Tenth International Joint Conference in Artificial Intelligence,* 1987.

24. Wellman, M.P. "Formulations of Tradeoffs in Planning Under Uncertainty." MIT/LCS/TM-332, MIT Laboratory for Computer Science, Cambridge, MA, 1987.

25. Wellman, M.P. "Probabilistic Semantics for Qualitative Influences." *Proceedings of the National Conference on Artificial Intelligence,* American Association for Artificial Intelligence, 1987, 660-664.

26. Wellman, M.P. "Qualitative Probabilistic Networks for Planning Under Uncertainty." *Uncertainty in Artificial Intelligence,* J.F. Lemmer (ed.), North-Holland, 1987.

27. Wong, J.B., A.J. Moskowitz and S.G. Pauker. "Clinical Decision Analysis Using Microcomputers--A Case of Coexistent Hepatocellular Carcinoma and Abdominal Aortic Aneurysm." *Western Journal of Medicine,* 145, (1986), 805-815.

Publications

1. Grady, G. and R.Patil. An. "Expert System for Screening Employee Pension plans for the Internal Revenue Service." *Proceedings of AI and Law Conference,* Boston, MA, May 27-28, 1987.

2. Hirsch, D.E. "An Expert System for Diagnosing Gait for Cerebral Palsy Patients." MIT/LCS/TR-388, MIT Laboratory for Computer Science, Cambridge, MA, May 1987.

3. Kohane, I. "Temporal Reasoning in Medical Expert Systems." *Proceedings of MEDINFO-86: the Fifth Congress on Medical Informatics,* Washington, DC, October 26-30, 170-174.

4. Kohane, I. "Temporal Reasoning in Medical Expert Systems." MIT/LCS/TR-389, MIT Laboratory for Computer Science, Cambridge, MA, June 1987.

5. Kunstaetter, R. "Intelligent Physiological Modeling: an Application of Knowledge Based Systems Technology to Medical Education." *Proceedings of MEDINFO-86: the Fifth Congress on Medical Informatics,* Washington, DC, October 26-30, 1986.

6. Kunstaetter, R. "Intelligent Physiological Modeling: an Application of

Knowledge Based Systems Technology to Medical Education." MIT/LCS/TR-360, MIT Laboratory for Computer Science, Cambridge, MA, April 1986.

7. Leong, T-Y. "Murmur Clinic: an Auscultation Expert System." MIT/LCS/TM-319, MIT Laboratory for Computer Science, Cambridge, MA, January 1987.

8. Long, W.J., S. Naimi, M. G. Criscitiello and R. Jayes. "Using a Physiological Model for Prediction of Therapy Effects in Heart Disease." *Proceedings of Computers in Cardiology Conference*, Boston, MA, October 7-10, 1986.

9. Long, W.J., S. Naimi, M.G. Criscitiello and S. Kurzrok. "Reasoning About Therapy From a Physiological Model." *Proceedings of MEDINFO-86: the Fifth Congress on Medical Informatics*, Washington, DC, October 26-30, 1986, 756-760.

10. Patil, R.S. "Review of Causal Reasoning in Medical Diagnosis." *Proceedings of Tenth Annual Symposium on Computer Applications in Medical Care*, Washington, DC, October 25-26, 1986, 11-15.

11. Patil, R.S. and W.E.L. Grimson (eds.). AI in the 1980s and Beyond: An MIT Survey. Cambridge, MA, MIT Press, 1987.

12. Patil, R.S., "A Case Study in Evolution of System Building Expertise: Medical Diagnosis." AI in the 1980s and Beyond: An MIT Survey. R.S. Patil and W.E.L. Grimson (eds.), Cambridge, MA, MIT Press, 1987.

13. Russ, T.A. "A System for Using Time Dependent Data in Patient Management." *Proceedings of MEDINFO-86: the Fifth Conference on Medical Informatics*, Washington, DC, October 1986, 165-169.

14. Russ, T.A. "The Temporal Control Structure Reference Manual." MIT/LCS/TM-331, MIT Laboratory for Computer Science, Cambridge, MA, June 1987.

15. Sacks, E.P. "Hierarchical Inequality Reasoning." MIT/LCS/TM-312, MIT Laboratory for Computer Science, Cambridge, MA, February 1987.

16. Schwartz, W.B., R.S. Patil and P. Szolovits. "Artificial Intelligence in Medicine: Where Do Stand?" Sounding Board article in *New England Journal of Medicine*, 316, (March 12, 1987), 685-688.

17. Szolovits, P. "Expert Systems Tools and Techniques: Past, Present and Future." AI in the 1980s and Beyond: An MIT Survey. W.E.L. Grimson and R.S. Patil (eds.), Cambridge, MA, MIT Press, 1987.

18. Szolovits, P. "Computer Science and Medical Decision Support." *Proceedings of the 1985 Workshop on Computing and Medicine,* University of Texas Health Sciences Center at San Antonio, to appear.

19. Szolovits, P., J.P. Kassirer, W.J. Long, A.J. Moskowitz, S.G. Pauker, R.S. Patil and M.P. Wellman. "An Artificial Intelligence Approach to Clinical Decision Making." MIT/LCS/TM-310, MIT Laboratory for Computer Science, Cambridge, MA, September 1986.

20. Wellman, M.P. "Representing Health Outcomes for Automated Decision Formulation" *Proceedings of MEDINFO-86: the Fifth Conference on Medical Informatics,* Washington, DC, October 1986, 789-793.

21. Wellman, M.P. "Qualitative Probabilistic Networks for Planning Under Uncertainty." *Proceedings of the Workshop on Uncertainty in Artificial Intelligence,* Philadelphia, PA, August 1986, 311-318.

22. Wellman, M.P. "Formulation of Tradeoffs in Planning Under Uncertainty." MIT/LCS/TM-332, MIT Laboratory for Computer Science, Cambridge, MA, June 1987.

Theses Completed

1. Kunstaetter, R. "Intelligent Physiological Modeling: an Application of Knowledge Based Systems Technology to Medical Education." E.E. and S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, April 1987.

2. Novick, S.L. "Reasoning Over Time About the Causes of Arrhythmias." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

Theses in Progress

1. Fogg, D. "Design Under Multiple Performance Criteria." Ph.D. Dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected June 1989.

2. Koton, P.A. "Using Experience in Learning and Problem Solving." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected January 1988.

3. Russ, T.A. "Reasoning With Time Dependent Data." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected June 1988.

4. Sacks, E.P. "Piecewise Linear Reasoning." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected May 1988.

5. Wellman, M.P. "Formulation of Tradeoffs in Planning Under Uncertainty." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected May 1988.

6. Yeh, A. "Quantitative Qualitative reasoning." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, expected May 1988.

Talks

1. Coldberger, H. IIASA Conference on Informatics and Medicine, Tbilisi, USSR, October 1986.

2. Kohane, I. "Temporal Reasoning in Medical Expert Systems." MEDINFO-86: the Fifth Congress on Medical Informatics, Washington, DC, October 1986.

3. Kunstaetter, R. "Intelligent Physiological Modeling: an Application of Knowledge Based Systems Technology to Medical Education." MEDINFO-86: the Fifth Congress on Medical Informatics, Washington, DC, October 1986.

4. Long, W.J. "Using a Physiological Model for Prediction of Therapy Effects in Heart Disease." Computers in Cardiology Conference, Boston, MA, October 1986.

5. Long, W.J. "Reasoning About Therapy From a Physiological Model." MEDINFO-86: the Fifth Congress on Medical Informatics, Washington, DC, October 1986.

6. Patil, R.S. "Knowledge Representation Meets Knowledge Acquisition: What Are the Needs, and Where is the Leverage?" Panel Discussion at the Fifth National Conference on Artificial Intelligence (AAAI-86), Philadelphia, PA, August 1986.

7. Patil, R.S. "Coordinating Clinical and Pathophysical Reasoning for Medical Diagnosis." The First International Conference on Artificial Intelligence and

its Impact in Biology and Medicine (I.A. Biomed. 86), Montpellier, France, September 1986.

8. Patil, R.S. "Review of Causal Reasoning in Medical Diagnosis." Tenth Annual Symposium on Computer Applications in Medical Care, Washington, DC, October 1986.

9. Patil, R.S. "Causal Models for Reasoning and Explanation." Boston University Seminar on Expert Systems, Boston, MA, November 1986.

10. Patil, R.S. "Knowledge Representation: Lessons from Research in Medical Expert Systems." MIT/ILP Symposium on Expert Systems from the User's Perspective, Cambridge, MA, December 1986.

11. Patil, R.S. "Compiling Causal Knowledge for Efficient Diagnosis." Carnegie-Mellon University and University of Pittsburgh Clinical Decision Making Group, Pittsburgh, PA, January 1987.

12. Patil, R.S. "Exploiting Causal Constraint in Model-Based Reasoning." Applied Expert Systems, Inc., Cambridge, MA, February 1987.

13. Russ, T.A. "A System for Using Time Dependent Data in Patient Management." MEDINFO-86: the Fifth Conference on Medical Informatics, Washington, DC, October 1986

14. Sacks, E.P. "Piecewise Linear Reasoning." AAAI Workshop on Qualitative Physics, Urbana-Champaign, IL, May 1987.

15. Szolovits, P. "Medical Artificial Intelligence." Meet the Experts session, Symposium on Computer Applications in Medical Care, Washington, DC, October 1986.

16. Szolovits, P. "Medical Artificial Intelligence and Planetary Space Exploration." NLM/NASA Symposium for the Sesquicentennial of the National Library of Medicine, Bethesda, MD, November 1986.

17. Wellman, M.P. "Representing Health Outcomes for Automated Decision Formulation." MEDINFO 86: the Fifth Conference on Medical Informatics, Washington, DC, October 1986.

18. Wellman, M.P. "Qualitative Probabilistic Networks for Planning Under Uncertainty." Workshop on Uncertainty in Artificial Intelligence, Philadelphia, PA, August 1986.

19. Wellman, M.P. "Automated Decomposition of Multiattribute Utility Functions." ORSA/TIMS Joint Conference, New Orleans, LA, May 1987.

COMPUTATION STRUCTURES

Academic Staff

Arvind, Group Leader

J.B. Dennis

R.S. Nikhil

Research Staff

G.A. Boughton

Graduate Students

P.S. Barth
S.A. Brobst
A.A. Chien
T-A. Chu
D.E. Culler
G-R. Gao
G. Guha Roy
S.K. Heller
R.A. Iannucci
S. Jagannathan
V.K. Kathail
G.K. Maa
D.R. Morais
G.M. Papadopoulos
K.K. Pingali
S.A. Plotkin
R.M. Soley
I. Taylor
K.R. Traub
E.W. Waldin
S. Younis

Undergraduate Students

H. Chan
J. Cheng
S. Desai
J. Hom
J. Hicks
C. Joerg
T. Leung
S. Malinack
S. Manandhar
J. Nieh
T. Olkin
M. Penniston
S. Sanghani

Support Staff

S.M. Hardy

N. F. Tarbet

Visitors

M. D. Atkins
A. Konagaya
J. Lewis
M. Mack
H. Nohmi
S. Truve

Technical Staff

J.P. Costanza
R.F. Tiberio

1. INTRODUCTION

The Computation Structures Group (CSG) made significant progress from July 1986 through June 1987. The primary thrust continues to be towards general-purpose parallel machines, focusing on functional and declarative languages, and on the Tagged-Token Dataflow Architecture (TTDA). In addition, we have been looking at parallel graph reduction, persistence in functional languages and the TTDA, and functional databases.

Our dataflow programming language Id Nouveau (or Id, for short) was redesigned and reached some degree of stability, based on vastly improved understanding of the denotational and abstract operational semantics of I-structures, our parallel data-structuring primitive. With experience in Id, we are beginning to develop a programming methodology for languages with I-structures. We are also looking beyond I-structures to other parallel, determinate object-oriented structures.

The Id compiler was completely rewritten in Common Lisp, and is now used quite heavily. Work continues in developing a type-system for Id and incorporating a type-checking module in the compiler. Because of its extremely modular structure, the compiler is very amenable to modifications, and is thus being used for research into other parallel architectures and languages.

A major activity in our Group was the development of Id World, an integrated programming environment for experimenting with parallel programs. The components of Id World include editor customizations for Id, the Id compiler, GITA -- an extensively instrumented emulation of the Tagged-Token Dataflow Architecture, and the Id debugger. These are packaged in an integrated user interface that makes it extremely easy to prepare, run, debug and study parallel programs quickly. We believe that with its flexibility and power, Id World is a unique tool for studying parallelism in programs.

Various members of our Group have begun using Id World to study a variety of application programs. Id World was publicly released in April 1987, and is available to anyone with a Symbolics or TI Explorer Lisp Machine. Based on the interest it has generated so far, we expect that Id World will soon be used in several research projects in the U.S. and abroad.

In the past year we have conducted many experiments to study instruction counts, instruction mixes, the effects of memory latency, granularity of parallelism, networks, program mapping, etc. These experiments have significantly improved our understanding of fundamental issues in parallel architectures, and thrown much light on the TTDA and its relation to conventional von Neumann machines. This improved understanding of dataflow and parallel von Neumann machines has sparked two exciting new research efforts.

One such effort is "Monsoon," a concrete architecture for the thus-far relatively

abstract Tagged-Token Dataflow Architecture. Monsoon solves many of the open questions about the TTDA, such as the implementation of the Waiting-Matching store, and for the first time we have reached a point where we are ready to build a hardware dataflow machine. We are seeking funding to build the machine in the next three years. We have already begun building and studying prototypes of some components of the Monsoon architecture.

The second outcome of this deeper understanding has been research into hybrid von Neumann/dataflow architectures. Here we are exploring changes to the classic von Neumann architecture, borrowing ideas from dataflow. These changes address the fundamental latency and synchronization problems that we believe make it difficult, if not infeasible to use conventional von Neumann architectures in a general-purpose parallel machine.

The study of resource management issues continues to be a major effort. We have also begun to study various other topics that concern both language design and architectural support thereof: lazy evaluation, input-output and general persistence of arbitrary objects. We have also continued work DisCoRd, an emulator for a parallel graph-reduction machine on the MEF.

In April 1987 we participated in the third MIT/IBM Workshop on parallel computing held in Essex, CT (held every other year). This is a useful forum for the exchange of ideas -- we learned about the status of IBM's RP3 architecture and hardware construction, and about the EPEX programming environment for studying parallel programs, in use by IBM and several university partners. We continue to have strong and productive connections with IBM, particularly with Dr. K. Ekanadham. We now also have strong connections with Cornell University -- Dr. Pingali joined the faculty there after graduating from MIT in August 1986.

Professor Nikhil is also working on the database problem in the Computer-Aided Fabrication Project (Professor Paul Penfield). Under his guidance, they have implemented a database system interface using ideas from functional languages. It is unique because with its clean functional formalism, non-computer experts have been able to learn its use rapidly; it is accessible both from C and from Common Lisp; and it encompasses data transparently from three distinct and different real database management systems.

The Multiprocessor Emulation Facility (MEF) has stabilized to a large extent. It currently consists of 32 TI Explorer Lisp Machines on our circuit-switch network, with no current plans for enhancements or improvements. It is currently used for two kinds of emulations: the TTDA and DisCoRd. The generic MEF software to support arbitrary parallel emulators has been rewritten and cleaned up significantly as a result of our experience with the dataflow and graph-reduction emulators. We have given several public demonstrations of the MEF running parallel programs. These and various related matters are discussed in greater detail in sections below.

2. PERSONNEL

Over the course of the year, Computation Structures had six visiting researchers. Hitoshi Nohmi, a hardware specialist from Nippon Electronic Corporation, spent his year-long study leave strengthening his knowledge of computer languages, and participating in aspects of MEF design work. His colleague Akihiko Konagaya arrived in January to study dataflow and reduction architectures, and to further his interest in logic languages with constraints.

Staffan Truve, a Fulbright fellow from the Chalmers University of Technology in Sweden, joined the group to continue his study of logic languages. John Lewis, taking a year off from his post at Boeing in Seattle, has been studying the parallel execution of scientific applications.

1986 saw the termination of IBM/Endicott's participation in CSG design work. In late September, senior associate engineers Michael Mack and Mark Atkins returned to Endicott. Mack had successfully designed the group's first chip, an 8 x 8 4-bit crossbar, and made a sizable contribution to the design of the MEF transmitter. Atkins had been working on the FIFO logic and 48Mhz clock subsystem of the MEF.

The departure of the Endicott team did not mean the end of IBM's interest in the group's work, however. Although not formally a visitor, Dr. Ekanadham, a researcher in parallel processing at IBM/Yorktown Heights, has worked very closely with us over the past year. His work includes studies comparing the instruction set of the TTDA with von Neumann machines, and a significant rewrite of the SIMPLE application to take advantage of Id's high-level features.

On March 1, Prof. Jack Dennis took early retirement to devote more time to his new company, Dataflow Technologies. He now holds the rank of Senior Lecturer in Computer Science, and in that capacity will see his remaining graduate students through to the completion of their degrees. He will also continue to pursue his long-standing VimVal interests.

3. ID WORLD

Last year we reported the completion, by Dinarte Morais, of a first version of Id World, an integrated programming environment for preparing, running, debugging and analyzing Id programs and their behavior on the Tagged-Token Dataflow Architecture. Based on that experience, we invested much effort this year in a major redesign to make it stable and flexible, with the objective of not only making it easier to use internally, but also to release it publicly for use elsewhere. Originally aiming for January 1987, we released Id World in April 1987. Id World is implemented mostly in Common Lisp and runs on Symbolics and TI Explorer Lisp Machines. It is accompanied with extensive documentation that minimizes prerequisite knowledge of Lisp Machines [9]. We have plans to port it to other popular workstations, such as Suns and MicroVaxes. Several

research projects in the U.S. and abroad have expressed interest in acquiring and using Id World.

The program development process in Id World is patterned after the analogous process for Lisp. The editor (currently Zmacs) is customized for Id -- it knows about Id syntax and has commands that facilitate entry and formatting of Id programs. The editor also has commands to invoke the Id compiler on segments of Id source code -- the resulting object code (dataflow graphs) is normally loaded automatically into GITA, the "Graph Interpreter for the Tagged-Token Architecture." In case of compilation errors, the editor can immediately display the relevant parts of the source for correction.

To run compiled Id programs, one switches to the GITA window. Here one can compile and load Id files, and invoke any compiled Id function on GITA, the graph interpreter. Id objects for function arguments may be entered using a special "Lisp-ified" syntax.

A significant and unique part of Id World is the Id debugger, designed and implemented by Mr. Dinarte Morais. In case of run-time errors, GITA enters the debugger, which has commands similar to the Lisp debugger except that they are with respect to a tree of contexts instead of a stack, because this is a truly parallel emulator. Using the debugger, the programmer can examine the state of the machine, mostly using source-language names and constructs. A unique feature of Id is that the set of run-time errors (a parallel machine may not have a *single* run-time error!) does not change with machine configuration or run-time scheduling, and so debugging is easy, and may be performed on a single processor.

After debugging, programmers can study the parallel behavior of Id programs: they can choose the number of processors, the network latency, and the kinds of run-time statistics that should be collected. After running the program, they can immediately display and plot the statistics on the three graphics panes in the GITA window. There are facilities to save, restore and hardcopy statistics.

Id World is exciting not only for dataflow research, but also for other approaches to parallelism. Because the parallelism in Id Nouveau and the TTDA is limited *only* by data dependencies, Id World can supply a reference point for the *maximum* parallelism in a given algorithm -- a calibration point against which one can compare the actual parallelism obtained in an encoding of the algorithm in, say, parallel Fortran running on a parallel von Neumann machine.

The development and implementation of Id World involved a significant cooperative effort by many members of the group, notably Dinarte Morais, Richard Soley, Ken Traub, Ian Taylor, and David Culler.

4. LANGUAGES AND SYSTEMS

This year saw significant effort on several language and systems issues in Id Nouveau and the TTDA.

4.1. I-Structure Semantics

The concept of I-structures as an architectural idea for parallel data structures is not new to the dataflow project, but it had long been unclear how to incorporate them into a programming language. Two significant steps this year cleared the way. First, Keshav Pingali established a connection between I-structures and Logic Variables -- variables whose values are incrementally refined by Unification -- and thus showed a fixpoint denotational semantics for a language with I-structures. Based on this, Profs. Arvind and Nikhil and Keshav Pingali, assisted by Vinod Kathail, developed an abstract Plotkin-style operational semantics for Id Nouveau. The semantics are given as rewrite rules that transform an Id program to its result, and capture exactly the parallel dataflow behavior of the program [2].

Abstractly, the machine state is modeled as a number of components *executing in parallel* -- an expression and zero or more statements:

```
E ; S ; ... ; S
```

The result of the program is the ultimate value of the expression. The rewrite rule for function applications looks like this:

```
(f Eargl ... Eargn)
E ; S ; ... ; S
-------------------------------
(Ebody')
E ; S ; ... ; S ; x1 = Eargl ; ... ; xn = Eargn
```

i.e., the upper machine state which somewhere contains the expression `(f...)` can be rewritten to the lower machine state with the expression replaced by `(Ebody')`, where `Ebody'` is the body of function `f` with the formal parameters given new names `x1` through `xn`. This captures exactly the behavior in the TTDA where a function can begin executing while its arguments are still being computed. We call this dataflow behavior the *parallel call-by-value* computation rule. Similarly, there is a rule which says that an identifier can be substituted only when there is a statement in the machine state that binds the identifier to a reduced *value* -- this corresponds exactly to the arrival of a token on an arc in the dataflow machine.

The rule for I-structure allocation is:

```
(array (vl,vu))
E ; S ; ... ; S
-------------------------------
(<Xvl,...,Xvu>)
E ; S ; ... ; S
```

Where `Xvl` through `Xvu` are new variables. This rule illustrates a difference from *functional* languages -- rewrite rules for functional languages never introduce new variables on the right-hand sides.

An exciting aspect of these rewrite rules is that, for the first time, one can now understand the parallelism of the dataflow machine purely in source-language terms, without any appeal to dataflow graphs or the Tagged-Token Dataflow Architecture. In addition to facilitating the dissemination of dataflow ideas to a wider audience, this simplification also gives us a much better perspective on the relation of dataflow to other approaches to parallel execution of functional languages such as parallel graph reduction.

4.2. Id Nouveau

The advances in understanding the semantics of I-structures gave us insight into the semantic categories to be supported in the language Id Nouveau. The difficulties were in integrating cleanly the expression-oriented constructs of the functional subset with the refinement-oriented constructs of the I-structure subset. The syntax was frozen in January 1987, and the compiler and Id World upgraded accordingly.

One consequence of the introduction of I-structures is that the language loses "referential transparency", thus making it more difficult to reason about programs. For this reason, many in the functional programming community are still skeptical about I-structures and advocate the use of "bulk" functional array operators. For example,

```
make-array n f
```

returns an array of size `n` such that the `i`'th component contains `(f i)`. Thus the returned array can be considered a "cache" for a finite part of `f`.

We have argued in [3] that any fixed set of functional primitives will result in inefficient programs. The programmer must be allowed to invent and code new array abstractions, and for this, I-structures are essential in the language. We have thus gradually evolved a programming methodology in which one part of the program contains the definitions of program-specific array abstractions using I-structures, and the remaining, major part of the program is a purely function program that uses these abstractions and does not mention I-structures at all. We are still experimenting with this programming methodology.

To encourage this programming style, we have defined a large library of standard functional array, list and set operators [8] in the hope that programmers will use this common library and train themselves to think along those lines. Paul Barth wrote the code for these libraries which are now loaded automatically as part of Id World.

4.3. Id Compiler

Version II of the Id Compiler, written by Kenneth Traub, compiled its first program one month ahead of schedule in July 1986. Besides simply accepting the latest version of the Id programming language, Version II has several features which set it apart from most other compilers, including its predecessor, Version I:

- It is founded on a common core of data structures and abstractions that is general and powerful enough to support all conceivable dataflow compilers. This common core is described in Kenneth Traub's "A Dataflow Compiler Substrate" [11].

- It has a highly modular structure, and includes a facility (known as `defcompiler`) which permits modules to be incorporated into the compiler with very little effort.

- It includes a novel attribute grammar evaluator which incrementally computes parse tree attributes on demand, and automatically adjusts to changes in parse tree structure made by source-to-source transformation modules. The evaluator is designed to work from grammatical specifications developed with **PAGEN**, a parser generator program also written by Traub.

- It supports incremental compilation through a sophisticated database mechanism for recording properties of Id procedures. Assumptions about separately compiled procedures are recorded in object code, allowing the consistency of a collection of procedures to be verified at load time.

- It includes a number of code optimization modules, including such well-known transformations as common subexpression elimination, loop invariant code motion, and procedure integration (these were implemented by Ian Taylor). There is also a peephole optimizer for dataflow machine code, believed to be the first use of peephole optimization within a compiler for dataflow architectures. The peephole optimizer is also noteworthy as it is completely specification-driven.

- The compiler and **PAGEN** are written entirely in Common Lisp, ensuring their portability.

Many of these features reflect the Id Compiler's special nature as a *research* compiler; it is specifically designed to support experiments at all phases of the compilation

process. Already, the compiler has proved itself adaptable enough to be used in two projects for which it was not originally designed: Bob Iannucci has modified the back end of the compiler to support his VNDF architecture, resulting in a compiler from Id to VNDF object code, while John Lucassen of LCS' Programming Systems Research Group has replaced the front end with one for his FX language, resulting in a compiler from FX to TTDA object code.

So far, Version II of the Id Compiler has proved to be an overwhelming success. Future plans include a type-checking module to be written by Prof. Nikhil, support of pattern-matching syntax, and code-generation for the Monsoon architecture.

4.4. Types and Type-checking in Id Nouveau

In Fall 1986 Prof. Nikhil implemented a first version of a Milner-style polymorphic type-checker for Id. One problem was to devise the type-checking rules for I-structure constructs, which are analogous to side-effects in a functional language such as ML, and which normally make the Milner-style rules unsound. A solution exists in implementations of ML, but have never been published, and so we had to re-invent it for Id Nouveau.

Preliminary use of the type-checker was very encouraging -- it promises to be a major aid in debugging and compiling. Our plans for the type-checker are:

- Upgrade it for the current Id Nouveau syntax,
- Fix a major limitation, which is the lack of user-defined union types and the associated pattern-matching syntax,
- Design a limited inheritance capability which is another kind of polymorphism that also simplifies programs,
- Use the type information to improve compiled code,
- Permit incremental type-checking, and the coexistence of typed and untyped code.

4.5. Lazy Structures

Eager interpreters are able to exploit vast parallelism, yet lazy interpreters have more desirable termination properties. Pingali proposed a source-to-source program transformation for achieving lazy behavior within an eager interpreter [10]. Pingali's approach offers the power of a lazy interpreter within the framework of dataflow, but is difficult in practice. When some values are not demanded, cleanup problems occur. For example, forks are not self-cleaning -- if one arm does not demand a value that is demanded by the other arm, the value sits at the fork forever. This cleanup problem is

quite difficult in the context of the TTDA, and we cannot ignore it. If values are always demanded by all possible consumers, lazy evaluation buys us nothing.

Steve Heller is considering another approach. An eager interpreter evaluates expressions as soon as the inputs are available, and a lazy interpreter evaluates an expression if and only if its value is needed to produce an answer. These extreme positions span a spectrum of possibilities, and he is studying some of these mixed evaluation strategies. If we delay only those expressions that sit in array slots, an interesting compromise is achieved. The TTDA already synchronizes array producers and consumers in hardware using I-structure Memory [3][4][5]. A similar synchronization mechanism is required to support demand propagation for delayed expressions that sit in array slots. By generalizing I-structures to *L-structures* ("lazy" structures) we can support both producer/consumer synchronization and demand propagation in hardware.

4.6. Accumulators

In studying various applications, we have repeatedly encountered a paradigm that cannot efficiently be handled by functional data structures or I-structures. One initializes an object, performs numerous "accumulations" on that object, and finally reads the value of the object. Because the accumulations are commutative, the order of accumulations is immaterial. An example would be to compute a histogram of 10 000 values into 10 intervals. In an imperative (and sequential) language, one would start with an array with 10 zeroes, and repeatedly increment the components. This cannot be done with functional data structures or with I-structures without much copying because one cannot update an array element in place. On the other hand, it is safe to do them in parallel because the increments may be done in any order.

Profs. Arvind and Nikhil, Keshav Pingali and Kenneth Traub have produced an initial proposal -- both linguistic and architectural -- to solve this accumulation problem in Id Nouveau on the TTDA. For example, to allocate an array for the histogram, one says:

```
xa,xr = 1D_accumulator (1,10) 10000 (+)
```

This allocates a vector with bounds 1 and 10, where each cell can accumulate values by addition, and where a total of 10000 accumulations are allowed. The expression returns two descriptors -- an accumulate-only descriptor `xa` and a read-only descriptor `xr`. One can increment bucket (cell) `j` by saying:

```
1D_accumulate (xa, j, 1)
```

One can read bucket `j` as if it were an ordinary I-structure: `xr[j]`. However, the token for `xr` is not released until `10000` accumulations are done, thus ensuring that there are no read-write races. The function `1D_accumulator` is non-strict in `n`, the number of accumulations, so that the number of allowed accumulations need not be known beforehand.

We know how to compile these constructs into dataflow graphs for the TTDA and plan to implement it and start using it immediately. We certainly do not expect this to be the final word on accumulators -- the hope is that experience in using it will allow us to understand the problem better and to produce a better solution.

4.7. Serialization and Serial Input-Output

Richard Soley has begun exploring explicit and implicit serialization methods, with the aim of controlling machine resources efficiently during the execution of combinatorially explosive expert system programs. This serialization would be carried out in a completely distributed fashion, without any centralized control over the system. This methodology has carried over to a serialization scheme to enable serious input/output facilities within the Id language. In Soley's approach, serialization of calls to I/O primitives is carried out by the Id compiler, which adds static and dynamic program arcs to order the run-time execution of I/O primitives as the programmer has implicitly specified.

4.8. DisCoRd: Parallel Graph Reduction

Ian Taylor has been working on modifying the Id Compiler to generate combinator code for DisCoRd, a parallel graph reduction architecture. In contrast to other parallel graph reduction machine projects, we assume eager evaluation wherever possible, using this assumption to minimize message traffic. Ted Leung has been working on implementing an emulator on the MEF for the parallel graph reducer. This work is a redesign of the initial version of DisCoRd that we reported last year.

4.9. Persistence in Id/TTDA

Bhaskar Guha Roy and Prof. Nikhil have been investigating the design and implementation of databases on the Tagged-Token Dataflow Architecture. Many database applications contain a high degree of inter-transaction parallelism and performance is often limited by disk latency. We believe the TTDA provides an excellent substrate for a high-performance database machine because of its ability to tolerate high latencies. Dataflow allows us also to take advantage of intra-transaction parallelism. I-structure memory allows synchronization among tasks to be expressed naturally, and the synchronization is achieved in hardware.

We are developing extensions to Id Nouveau for experimenting with functional databases. The main extension is the *bag* data structure for modeling large, homogeneous collections of objects. Bags have parallel, I-structure-like semantics. We are currently examining how operations in this language can be implemented to take advantage of features of the Monsoon architecture.

We have also been working on architectural extensions to the TTDA to support

persistent store (disks). Objects of any data type in the language can be made persistent. A persistent object is initially referred to by a *logical name*, and can be associated with a name in the program. The actual movement of data from primary to persistent store is transparent and incremental. We are currently extending the Id Nouveau compiler to support a variety of operations related to persistent objects. In the coming year, our goal is to complete the design of the persistent storage system and design and implement a complete transaction processing system.

4.10. Environments as First-Class Objects

This past year, Suresh Jagannathan (with Prof. David Gelernter of Yale University and Prof. Nikhil) has been examining the ramifications of incorporating *environments* as first-class values into a programming language. We have produced a new programming language with several novel features. Symmetric Lisp is built around an environment-building structure, the ALPHA form. The semantics of an ALPHA is derived by (conceptually) transposing the familiar Lisp PROGN or Algol compound-statement symmetrically around a time-space axis. Where the elements of a PROGN are evaluated during sequential lifetimes in a fixed temporal order, the elements of an ALPHA form are evaluated during concurrent lifetimes in a fixed spatial order. Concurrent evaluation lifetimes mean that the ALPHA form is a concurrency creating structure, and that Symmetric Lisp is a parallel language. Because elements of an ALPHA can refer to one another, they all have a shared evaluation lifetime. Shared lifetimes mean that the elements of an ALPHA-form may be taken to define a *scope*; all name-binding, program-building and scope-defining mechanisms in the language are based on this form. The semantics of Symmetric Lisp is defined by a collection of rewrite rules that preserve the structure of the source program (that is, the number and order of its elements). Unlike other languages, the "shape" of a program is invariant over the transformation process. Thus, ALPHA forms evaluate to new ALPHA forms and, consequently, Symmetric Lisp has no notion of a data structure: a data structure is simply any program that evaluates to itself.

"First-class environments" means that Symmetric Lisp allows programmers to write expressions that evaluate to environments and to create and denote variables and constants of type environment. One consequence is that the roles filled in other languages by a variety of limited, special-purpose environment forms like records, structures, closures, modules, and classes are filled instead by the ALPHA. In addition to being the fundamental structuring tool in the language, environments also allow us to treat function application as syntactic sugar for environment building: LAMBDA-forms become constants and are no longer constructs in their own right. Because the elements of an environment are evaluated in parallel, Symmetric Lisp is a parallel programming language intended for implementation on fine-grained architectures such as a dataflow or graph-reduction machine. Because environments may be constructed statically as well as dynamically, Symmetric Lisp accommodates an unusually flexible and simple parallel interpreter that is well-suited as an interface to a concurrent, language-based

Symmetric Lisp computer system. Our goal for the coming year is to refine the design of the language and build an implementation on an available multi-processor architecture such as the MEF.

4.11. Functional Databases

Prof. Nikhil has been guiding the database effort in the Computer-Aided Fabrication (CAF) project run by Prof. Penfield. The problem here is to provide a single on-line information facility that encompasses not only traditional data-processing mainstays, such as personnel and accounts, but also highly complex data such as IC masks, process-flow programs, intermediate states of process-flow programs, wafer states, etc. In addition, the facility must be able to access data from other existing software packages such as IC simulation packages.

With our suggestions and guidance, Mike Heytens, a graduate student in the CAF project, has designed and implemented GESTALT, a Functional Data Model interface, in which one views information as a collection of database types and functions that map between those types. These functions are embedded in a full functional language. The database is unique in its power and flexibility. The interface uses three separate commercial database management systems underneath for data storage; however, users see a single, integrated model of all the data. Accessible from C and Common Lisp, users have been able to learn to use it very quickly, and it is in daily use. We expect to continue this collaboration with the CAF project. Now that immediate operational needs have been met, we are exploring several enhancements to the type system, and to the data model so that it incorporates a notion of history. That is to say, data is never updated, only appended to.

5. ARCHITECTURES

5.1. TTDA Experiments

Gino Maa conducted extensive experiments using the existing emulation tools to study the effects of the various code-mapping strategies and grain sizes, the presence of significant latencies in the communications network, and interleaved memory allocation on the performance of relatively large systems (hundreds of processors) executing a large-scale scientific application kernel. The results have all shown that a dataflow machine, when running large programs with sufficient parallelism, is indeed very tolerant of extreme communications latencies: system performance degrades very gracefully even with an almost order-of-magnitude increase in such latencies.

The resource allocation experiments provided evidence that relatively simple run-time strategies such as round-robin and randomized code mapping produce surprisingly good results consistently. They also showed that by choosing the grain size of the code-mapping unit to be around the iteration level, we get the scalability characteristics of

mapping at the instruction level while still maintaining much of the locality property of mapping at the code-block level. Past studies on data structure reference patterns have indicated that contention for specific memory locations may hinder scalability in large systems, but recent experiments showed that a modest degree of interleaving in memory allocation yields much improvement over a non-interleaved memory system, although some contention for constant data structures can only be eliminated by altering the source program.

5.2. Storage Usage

GITA and the new Id compiler have been put to extensive use by David Culler in studying the resource requirements of dataflow programs under a variety of conditions. Our expectation that storage requirements grow in proportion to the amount of unfolding under idealized execution with unrestricted parallelism was confirmed; this implies cubic growth for triply nested loops, for example. Moreover, restricting the amount of parallelism exploited in executing a program does not alleviate the problem, rather, it is necessary to constrain the unfolding of the program itself. Loop-bounding techniques developed by Culler have proved effective in this; under restricted parallel execution, it is possible to reduce resource requirements dramatically without increasing the running time of the program appreciably. We are continuing to explore this direction.

5.3. Instruction Counts

An important metric in assessing the effectiveness of the dataflow approach is the total number of instructions executed. TTDA instructions are roughly comparable in power to those of a load/store architecture - memory access operations are disjoint from arithmetic operations. We expect instruction counts of dataflow programs to be somewhat higher than a good sequential implementation, as there is a certain amount of work required to initiate and synchronize concurrent computations. Nonetheless, for dataflow machines to be viable, they must be comparable to sequential machines in this regard. Comparative studies performed by David Culler in conjunction with K. Ekanadham at IBM/Yorktown indicate that Id programs with little optimization typically require 2-3 times as many instructions as highly optimized Fortran. This comparison is encouraging in light of the number of additional instructions that would be performed in a "parallelized" Fortran version. However, it is clear that without relatively sophisticated program graph generation, as in the current Id compiler, the gap would be much worse. Recent work with common subexpression elimination has narrowed the gap dramatically.

5.4. Towards Real Implementations

While it has been well understood for some time that dataflow offers a framework for thinking about parallel computation, it is only within the last year that we have been able to make statements about the *essence* of dataflow architecture which may be meaningfully applied in the von Neumann multiprocessor domain. It has become clear through analysis [1] and experiments by Gino Maa that any scalable architecture must be able to tolerate basic, machine-induced latencies and must provide, at the hardware level, a synchronization mechanism that is inexpensive to use. The former is necessitated by the physical partitioning of a machine into cooperating processing and memory elements separated by nontrivial communication delays. The latter is a direct result of decomposition of the program into communicating pieces or *tasks*.

Dataflow by its very nature allows parallelism in the program to be traded off against latency -- given sufficient parallelism in the program (on the order of the processor-memory-processor pipeline depth *times* the number of such parallel pipelines), latency cost as measured by induced processor idle time can be controlled. Dataflow also offers a uniform synchronization paradigm through the tagging and matching of data. Each enabled instruction represents a *task* which can execute independent of all other such tasks. Tags serve to identify these tasks. Dataflow machines provide the means for bringing together identically tagged values; this is the necessary and sufficient condition for the task's execution.

The essential features of the dataflow mechanism are a large namespace for identifying "meeting places" (synchronization events), provision at the hardware level for multiple, concurrent tasks, and the ability to switch between these tasks as necessary with speeds approaching single instruction times.

Von Neumann Dataflow Machine

One proposal which has grown out of this work is the construction of a hybrid dataflow vonNeumann machine by extending von Neumann architecture with some embodiment of the essential features of dataflow. The proposal is made and discussed by Iannucci [6]. The goal of this work is to refine further the notion of *essential features*, and to explore compiler-directed, pipelineable sequential code sections as a tool for implementing resource management primitives and for exploiting vector-type instructions.

One possible approach is to recognize the relationship between arcs in a compiled dataflow graph and slots in a traditional invocation stack frame. Both are used for holding temporaries local to the invocation of the associated procedure. As such, they embody the intra-process communication mechanism. Augmented with a basic synchronization mechanism, stack frame slots allocated out of a relatively large address space would provide two of the three above-mentioned features considered essential for a scalable multiprocessor. Local memory with I-structure-like synchronization bits on each slot (indicating *slot-empty*, *slot-full*, or *deferred-read*) [5] provides such a

synchronization mechanism; deferred reads cause suspension of the current process and storage of the current program counter (PC) into the empty slot. Subsequent writing to the deferred slot reawakens the suspended process by extracting the deferred PC and making it a candidate for execution once again.

The third feature, fast task switching, implies sufficient high-speed storage to hold the computation state for a large number of such PCs (similar to the requirement for a large, fast waiting-matching memory and token buffer in a dataflow machine), and the ability to interleave instructions from different tasks on a per-instruction (or nearly so) basis. Note that it is neither essential nor always desirable to switch tasks at each instruction dispatch -- it is often the case that small groups of instructions may be statically scheduled for execution as a unit given the satisfaction of only a few input data dependencies. Thus, the quanta of execution may be bigger than single instructions. It is, however, essential that the task switching be done to the *resolution* of individual instructions.

Progress to date includes definition of a simple machine model, complete definition of the syntax and semantics of a suitable machine language, design, coding, and testing of a von Neumann/dataflow back end for the Id compiler (previously described), and preliminary work on an emulator for the architecture.

Monsoon

We have been sufficiently encouraged by our research results to contemplate and evaluate critically a hardware implementation of a multiprocessor based on the Tagged-token Dataflow Architecture. Central to this goal is the ability to translate the dataflow execution mechanism, specifically waiting-matching and I-structure operations, into practical and efficient hardware. Greg Papadopoulos has described a novel instruction execution mechanism that implements both the I-structure storage and the waiting-matching section in the same *explicitly addressed* storage. He has also given the outline for generating code for such a machine from TTDA-style dataflow graphs.

The processing element is somewhat more general than a TTDA graph interpreter. The design draws heavily on traditional pipelined von Neumann techniques as popularized by the "RISC" methodology. A processing element is really a *multi-threaded* non-blocking RISC pipeline, where a join of two threads, an operation, and a fork of two threads can all occur within a single pass through the pipe. Threads are interleaved each cycle, without switching overhead, from a hardware managed task queue. We address the two fundamental multiprocessing issues by (1) providing non-blocking split-transaction global memory references, and (2) providing very efficient hardware synchronization on an instruction-by-instruction basis. We believe that this architecture brings the dataflow machine a step closer to von Neumann machines, exploiting the efficiencies of pipelined designs while reducing the overhead of fine-grained data-driven evaluation.

We intend to construct a 256 PE multiprocessor prototype called *Monsoon*. Because

we believe the processor pipeline to be well balanced and technologically scalable, we are initially employing fairly conservative TTL and CMOS gate array technologies. Our initial implementation calls for each PE to have a 100ns. cycle time, 64-bit floating point, and two Megawords of local storage. The network will be a packet switched 256-way two-stage exchange with 800 Mbits/sec/port. This will yield a machine with a peak performance of over two GigaFLOPS. We believe it will *sustain* between 100-300 MegaFLOPS on a wide variety of scientific codes, making it competitive with the fastest general purpose von Neumann machines presently available. A discrete logic laboratory prototype of a processor (125ns. cycle time) is now under construction.

Compiling for Sequential Architectures

Many of the programming languages devised for fine-grained parallel architectures are *non-sequential*. Non-sequential languages, which include Id as well as most lazy functional languages, cannot be directly compiled into ordinary sequential code (as for a von Neumann machine). Instead, they must be compiled into fine-grain parallel code (as for a dataflow machine) or into many sequential threads, executed concurrently. In his Ph.D. research, Kenneth Traub is examining the problem of compiling non-sequential languages into multi-thread code. Multi-thread code can be run on von Neumann machines by simulating parallel scheduling, and so this work will address the problem of efficient execution of non-sequential languages on von Neumann machines. More importantly, this work will have direct application to machines whose architecture is based on a multi-thread model, such as Iannucci's VNDF.

As part of his dissertation work, Iannucci [6] has constructed a new code generator for the Id compiler. While the target machine architecture is markedly different *c.f.* TTDA, the relative ease with which new modules were integrated to the existing compiler was significant. This new compiler retains parse tree and program graph generation and substitutes a new machine graph generator, an altered peephole optimizer, and a new assembler for a von Neumann style (i.e., program counter based) architecture.

6. APPLICATIONS

K. Ekanadham of IBM/Yorktown, working closely with Prof. Arvind, has been rewriting the SIMPLE code in Id Nouveau. With extensive use of higher-order functions and array abstractions, Ekanadham's masterful SIMPLE code in Id has provided the best example to date of the high level that scientific programming can reach. His SIMPLE code has become the standard against which we compare the quality of codes written here and elsewhere.

During the fall, three scientists from Los Alamos National Laboratory visited the group to learn about our programming environment and to start work on several large dataflow applications. The discussions prompted a valuable review of Id. These visitors became a beta-site for the Id World release. One of their applications, a Particle-In-Cell

(PIC) electrodynamics code, had been implemented on a variety of parallel machines. Culler wrote a version of it in Id, and we are now comparing various implementations. The PIC code involved more sophisticated data structures than most scientific applications and stressed the expressiveness of the I-structure paradigm. One of the Monte Carlo codes involving neutron transport proved very difficult to express efficiently with I-structures and added to the on-going discussion of accumulators.

Paul Barth wrote a signal-processing application in Id Nouveau based on his experience at Schlumberger with software for oil-exploration. Jamie Hicks also wrote a general electronic signal-processing application in Id Nouveau. Both these experiments shed light on the need for streams and stream-processing operators in the language.

Paul Barth wrote several algorithms for the single-source, shortest path problem. These algorithms highlight several methods of applying dataflow to graph traversal problems. One algorithm uses I-structures for synchronizing the traversal of several parallel paths; another encodes the graph as a dataflow program that can be executed directly. Two algorithms were written that used nondeterministic constructs for marking the graph. These nondeterministic constructs are not currently part of Id Nouveau; their addition would support these algorithms, as well as many others, such as dynamic programming and search problems in AI.

Serge Plotkin studied the problem of writing a symbolic polynomial arithmetic package in Id Nouveau. This exercise again reinforced the need for "accumulators" in the language. For example, when multiplying two polynomials with coefficients $a_0,a_1,...$ and $b_0,b_1,...$ for x^0, x^1, ..., respectively, each coefficient c_j in the result is sum of products of the form $a_i \; x \; b_{j-i}$. The summing can be done in any order, and so can ideally be expressed as an "accumulation" of product terms.

Richard Soley has been performing experiments with Id World on pattern-matching systems, both as an approach to debugging Id World and a study of the potential sources of parallelism within "expert" production systems. He has identified various combinatorially explosive aspects of these computations, leading to his current work in efficiently serializing such highly parallel, and highly "speculative" programs.

Arun Iyengar joined our group late in the Spring Term, 1987. He will be studying the implementation of graph algorithms in Id Nouveau/TTDA based on his experience in writing applications in molecular biology.

In the fall, Jamie Hicks wrote an interface that enhances the use of the Quicksim circuit simulator, part of our Mentor logic CAD system. A designer can use an ordinary text editor to enter specifications of circuit inputs and expected circuit outputs, and then run a driver that performs the circuit simulation, automatically applying the inputs at the right (simulated) times and comparing the simulated outputs with the expected outputs. We expect this to be of use in our future hardware design efforts.

7. MULTIPROCESSOR EMULATION FACILITY

Andy Boughton, Jack Costanza, and Ralph Tiberio have been responsible for ensuring the reliability of the MEF hardware. During the past year the circuit switch and the other MEF hardware have stabilized. The high infant mortality rate among certain active components on the circuit switch accounted for the greatest number of failures. For example, many optoisolator failures were recorded in the first few thousand hours of service. As the number of hours on the circuit switch boards has gone up, the failure rate has gone down. Currently, the average time between circuit switch hardware failures is a few months. The reliability of the circuit switch has been more than adequate to support large experiments.

During the same period we have also seen a decrease in the failure rate of the MEF processors. We have kept detailed records of all MEF hardware failures in an IBM SQL database designed by Jack Costanza. The failure rate that we have observed on the MEF processors is consistent with industry averages. The failure rate of the circuit switch boards has been substantially less.

7.1. Network Development

We have redirected our development effort from the MEF to Monsoon. The departure of the IBM team of Atkins and Mack, coupled with the emergence of a design for an extremely practical hardware implementation of the Tagged-Token Dataflow Architecture has caused this change in emphasis. The MEF packet switch design of the IBM team promised greatly improved robustness, flexibility, and reliability over the existing MEF circuit switch. The team's departure, however, prompted the shift to the circuit switch. While the circuit switch has proven to be sufficient for the current 32 processor MEF configuration, the MEF packet switch would have provided the robustness and reliability necessary to support larger and more flexible MEF configurations.

The key characteristics required of a network for Monsoon are bandwidth and reliability. Monsoon requires a network capable of supporting 800 megabits per second of bandwidth on each network input. We believe that such a network can be constructed using concepts similar to those developed for the MEF packet switch.

Andy Boughton, Chris Joerg, and Greg Papadopoulos have developed an overall structure appropriate for the Monsoon network. The proposed structure is a staged packet switched network. The proposed network is composed of two stages of 16-input 16-output switch boards. The maximum size of the network is 256 inputs and 256 outputs. Each switch board is composed of eight four-input four-output Packet Switched Routing Chips (PaRC's). Each PaRC will be a complete switching node with a crossbar, control circuitry, and packet buffering. The data paths of the network are assumed to be 16 bits wide and capable of running on a 50 Mhz clock. The network also supports circuit switched connections between network inputs and network outputs.

This facility is required by the Monsoon architecture in order to allow a processor to maintain, if necessary, a strict order among the arrival times of its messages at other processors.

A preliminary logic design for one possible implementation of PaRC has been completed by Chris Joerg. The proposed implementation is based on a number of the concepts developed for the MEF packet switch [7]. The implementation uses LSI Logic compacted gate array technology. The proposed PaRC is composed of four major subcomponent types; fifo input controller, scheduler, transmitter, and crossbar. The overall structure is similar to that of the MEF packet switch board but there are some differences. PaRC uses a distributed scheduling scheme with a scheduler associated with each output. While buffering a packet received on a given input and destined for a blocked output, PaRC is capable of transferring a subsequent packet from the same input to a different output if that output is not blocked. PaRC is also capable of supporting circuit switched connections. The proposed PaRC was designed and simulated using LSI Logic 7000 series logic. At the time our CAD system only supported the 7000 series and we decided to proceed with a preliminary design for PaRC rather than wait for the upgrade of our CAD system. Since the 10000 series array is required to fabricate a chip of the size of PaRC, we must now transfer the design to the 10000 series and simulate it in more detail.

Jack Costanza and Ralph Tiberio have explored potential link technologies for the Monsoon network. The size of Monsoon may require some of its network links to be 30 to 40 feet long. We have tested a link technology that uses a 16 bit wide data path. This link is based on earlier work done by Mark Atkins, using coaxial cables and CMOS drivers and receivers. While our initial results have been encouraging, much more work is required to develop a link technology with the reliability that is needed for the Monsoon network.

7.2. Evolution of MEF Software into A General Emulation Model

Two years ago, we reported on Tanglewood, a powerful, general substrate for MEF experiments built on top of the EtherNet. Unfortunately, Tanglewood proved to be rather slow. Last year we put Tanglewood aside in favor of CSWITCH, an extremely lean, specialized interface to the MEF circuit switch network. This network interface was tightly integrated with the dataflow emulator MEF-GITA. In a sense, this year we have come full circle: the circuit switch interface has been abstracted from MEF-GITA and generalized to provide a simple, efficient substrate for a broad variety of MEF experiments. The new CSWITCH abstraction has facilitated many extensions to GITA and has been used as a substrate for other MEF experiments, including DisCoRd, a graph-reduction architecture, and the game of MultiLife.

7.3. Demonstrations on the MEF

MEF made a number of public appearances this year. The demonstration to participants in the Lisp and Functional Languages conferences in August drew a large crowd. All thirty-two TI Explorers in the MEF were used in executing a variety of dataflow programs. In December we showed various aspects of MEF-GITA and GITA to the executive director of DARPA. This included system utilities of the MEF, a large hydrodynamics code written in Id and running on thirty-two machines, and a non-dataflow MEF application, MultiLife, employing the new CSWITCH interface. In addition, we demonstrated many facilities in GITA for studying the behavior of parallel programs.

8. WORK UNDER PROFESSOR DENNIS' SUPERVISION

Tam-Anh Chu has completed his doctoral dissertation entitled "Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications" under the supervision of Prof. Dennis. In this thesis, he presents an approach for direct and efficient synthesis of self-timed (asynchronous) control circuits from formal specifications called Signal Transition Graphs (STGs). Control circuits synthesized from this graph model are speed-independent and capable of performing concurrent operation. The property of speed-independence means that the circuit operates correctly regardless of variations in delays of logic gates, thus implying that the circuit is hazard-free under any combination of gate delays. The capability of STGs for explicitly specifying concurrent operations internal to a control circuit is unique to this model, unlike other approaches based on Finite State Machines.

STGs are a form of interpreted Petri nets, in which transitions in a net are interpreted as transitions of signals in a control circuit. While other synthesis approaches based on Petri nets have not been very successful, we have developed a number of analytical results which establish the equivalence between the static structure of nets (their syntax) and their underlying firing sequence semantics--an analytical approach called structure theory of Petri nets. This equivalence permits the characterization of the low-level properties of control circuits in terms of STG syntax: the deadlock-free and hazard-free properties of circuits are characterized as syntactic properties of liveness and persistency of STGs. A preliminary STG specification of a control circuit can be modified into one which is live and persistent, from which a deadlock-free and hazard-free logic implementation can be derived mechanically.

STGs allow efficient synthesis of control circuits by using a method of decomposition based on a graph-theoretic technique called contraction. Instead of implementing a logic circuit from a STG directly, it can first be decomposed into a number of contracted nets, one for each signal generated by the control circuit. A logic element can then be determined from each contracted net, and the composition of logic elements produces the final circuit implementation.

References

1. Arvind and R.A. Iannucci. "Two Fundamental Issues in Multiprocessing." Computation Structures Group Memo 226-5, MIT Laboratory for Computer Science, Cambridge, MA, July 1986.

2. Arvind, R.S. Nikhil and K.K. Pingali. "Id Nouveau, Reference Manual Part II: Operational Semantics." Computation Structures Group, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

3. Arvind, R.S. Nikhil and K.K. Pingali. "I-structures: Data Structures for Parallel Computing." Computation Structures Group Memo 269, MIT Laboratory for Computer Science, Cambridge, MA, February 1987. Also in *Proceedings of the Graph Reduction Workshop*, Santa Fe, NM, October 1986.

4. Arvind and R.E. Thomas. "I-Structures: An Efficient Data Type for Functional Languages." Computation Structures Group Memo 178, MIT Laboratory for Computer Science, Cambridge, MA, October 1981.

5. S.K. Heller. "An I-Structure Memory Controller (ISMC)." M.S. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1983.

6. Iannucci, R. A. "A Dataflow/von Neumann Hybrid Architecture." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected August 1987.

7. Joerg, C.G. "Design of a Circuit Switched Routing Chip for a Dataflow Supercomputer." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

8. Nikhil, R.S. "Id Nouveau, Reference Manual Part I: Syntax." Computation Structures Group, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

9. Nikhil, R.S. "Id World Reference Manual." Computation Structures Group, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

10. Pingali, K.K. "Demand-driven Evaluation on Dataflow Machines." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, July 1986.

11. Traub, K.R. "A Dataflow Compiler Substrate." Computation Structures

Group Memo 261, MIT Laboratory for Computer Science, Cambridge, MA, March 1986.

Publications

1. Arvind and K. Ekanadham. "Future Scientific Programming on Parallel Machines." *Proceedings of the International Conference on Supercomputing (ICS)*, Athens, Greece, June 1987.

2. Arvind and R.A. Iannucci. "Two Fundamental Issues in Multiprocessing." *Proceedings of DFVLR - Conference 1987 on Parallel Processing in Science and Engineering*, Bonn-Bad Godesberg, Germany, June 1987. Also MIT/LCS/TM-330 and Computation Structures Group Memo 226-6, MIT Laboratory for Computer Science, Cambridge, MA, May 1987.

3. Arvind and R.A. Iannucci "Two Fundamental Issues in Multiprocessing." Computation Structures Group Memo 226-5, MIT Laboratory for Computer Science, Cambridge, MA, July 1986.

4. Arvind and R.S. Nikhil. "Executing a Program on the MIT Tagged-token Dataflow Architecture." *Proceedings of the PARLE Conference*, Eindhoven, The Netherlands, June 1987. (Also Computation Structures Group Memo 271, March 1987.)

5. Arvind, R.S. Nikhil and K.K. Pingali. "Id Nouveau, Reference Manual Part II: Operational Semantics." Computation Structures Group, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

6. Arvind, R.S. Nikhil and K.K. Pingali. "I-structures: Data Structures for Parallel Computing." Computation Structures Group Memo 269, MIT Laboratory for Computer Science, Cambridge, MA, February 1987. Also *Proceedings of the Graph Reduction Workshop*, Santa Fe, NM, October 1986.

7. Chien, A.A. "Hot Spots in Routing Networks: A Collection of Studies." Computation Structures Group Memo 267, MIT Laboratory for Computer Science, Cambridge, MA, October 1986.

8. Chien, A.A. "Structure Referencing in the Tagged-token Dataflow Architecture." Computation Structures Group Memo 268, MIT Laboratory for Computer Science, Cambridge, MA, October 1986.

9. Chu, T-A. "Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications." MIT Laboratory for Computer Science, Cambridge, MA, June 1987.

10. Chu, T-A. "Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications." *Proceedings of the International Conference on Computer Design*, IEEE, New York, October 1987.

11. Chu, T-A. "A Method of Abstraction for Petri Nets." *Proceedings of the International Workshop on Petri Nets and Performance Models*, Madison, WI.

12. Chu, T.-A. and L.A. Glasser. "Synthesis of Self-timed Control Circuits from Graphs: An Example." *Proceedings of the International Conference on Computer Design*, IEEE, New York, October 1986.

13. Chu, T.-A. and C.K.C. Leung, "Design of High Performance FIFO Queues for Packet Communication Architectures." *Proceedings of the International Conference on Parallel Processing*, IEEE, Chicago, IL, August 1986.

14. Gelernter, D., S. Jagannathan and T. London. "Environments as First-class Objects." 14th Conference on Principles of Programming Languages, Munich, Germany, January 1987.

15. Gelernter, D., S. Jagannathan and T. London. "Parallelism, Persistence and Meta-cleanliness in the Symmetric Lisp Interpreter." 1987 SIGPLAN Conference on Interpreters and Interpretive Techniques, St. Paul, MN, June 1987.

16. Nikhil, R.S., K.K. Pingali and Arvind. "Id Nouveau." Computation Structures Group Memo 265, MIT Laboratory for Computer Science, Cambridge, MA, July 1986.

17. Nikhil, R.S. "Id Nouveau Quick Reference Guide." Computation Structures Group (internal document), MIT Laboratory for Computer Science, Cambridge, MA, February 1987.

18. Nikhil, R.S. "Id Nouveau, Reference Manual Part I: Syntax." Computation Structures Group, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

19. Nikhil, R.S. "Id World Reference Manual." Computation Structures Group, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

20. Traub, K.R. "A Compiler for the Tagged-token Dataflow Architecture." MIT/LCS/TR-370, MIT Laboratory for Computer Science, Cambridge, MA, August 1986.

Theses Completed

1. Brown, D.A. "Concurrent Synchronous Simulation." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

2. Chien, A.A. "Congestion Control in Routing Networks." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, October 1986.

3. Chu, T.-A. "Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge MA, May 1987.

4. Joerg, C.G. "Design of a Circuit Switched Routing Chip for a Dataflow Supercomputer." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

5. Kaushik, S. "Design of a Cyclic Redundancy Code Generator Circuit for a Packet Switch." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

6. Traub, K.R. "A Compiler for the MIT Tagged-token Dataflow Architecture." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, August 1986.

Theses in Progress

1. Culler, D.E. "Effective Dataflow Execution of Scientific Applications." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected December 1987.

2. Guha Roy, B. "Transaction Processing on Dataflow Computers." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected December 1988.

3. Heller, S.K. "Efficient Streams on a Dataflow Machine." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected December 1987.

4. Iannucci, R. A. "A Dataflow/von Neumann Hybrid Architecture." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected August 1987.

5. Jagannathan, S. "The Design and Implementation of a Symmetric

Programming Language." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected August 1988.

6. Kathail, V.K. "Optimal Evaluators for Functional Languages." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected December 1987.

7. Maa, G. "Scalability of the Tagged-token Dataflow Machine." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1987.

8. Papadopoulos, G.M. "Implementation of a General-purpose Dataflow Multiprocessor." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected August 1987.

9. Traub, K.T. "Sequential Implementation of Non-sequential Programming Languages." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected June 1988.

Talks

1. Arvind. "Dataflow and the Multiprocessor Emulation Facility." DARPA/JASON Workshop on Advanced Computer Architecture Research, La Jolla, CA, July 10, 1986.

2. Arvind. "A von Neumann-Dataflow Machine." IBM Research, Hawthorne, NY, August 26, 1986.

3. Arvind. "Data Structures for Parallel Computing." Graph Reduction Workshop, Santa Fe, NM, October 1, 1986.

4. Arvind. "Quantifying Parallelism in Programs." Los Alamos National Laboratory, Los Alamos, NM, October 2, 1986.

5. Arvind. "Dataflow Architectures." Keynote Talk, ICCD '86, Rye Town Hilton, Port Chester, NY, October 6, 1986.

6. Arvind. "Dataflow Architectures." Michigan State University, E. Lansing, MI, October 16, 1986.

7. Arvind. "Parallel Computing: New Directions in Dataflow," MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, November 17, 1986.

8. Arvind. "Quantifying Parallelism in Programs." IBM-ACES 11th University

Study Conference, Bonaventura Hotel, Ft. Lauderdale, FL, November 19, 1986.

9. Arvind. "Dataflow Architectures." NCR Workshop on Distributed Systems Architecture, San Diego, CA, December 9, 1986.

10. Arvind. "Dataflow and Scientific Programming." Lawrence Livermore Laboratory, Livermore, CA, December 10, 1986.

11. Arvind. "Future Scientific Programming: Ek's SIMPLE Code." Parallel Processing: Matching Execution Models with Program Classes, A High-Speed Computing Conference, sponsored by Los Alamos and Lawrence Livermore National Laboratories, Gleneden Beach, OR, March 17, 1987.

12. Arvind. "Parallel Computing: The need to move away from the von Neumann Model." Keynote Talk, SEAS Meeting, Montpellier, France, April 7, 1987.

13. Arvind. "Monsoon: A realization of the MIT Tagged-Token Dataflow Architecture." The 3rd MIT-IBM Workshop on Parallel Processing, Essex, CT, April 20, 1987.

14. Arvind. "Future Scientific Programming: Ek's SIMPLE Code." The 3rd MIT-IBM Workshop on Parallel Processing, Essex, CT, April 21, 1987.

15. Arvind. "Future Scientific Programming." A Keynote Talk, International Conference on Supercomputing, Athens, Greece, June 11, 1987.

16. Arvind. "Executing a Program on the MIT Tagged-Token Dataflow Architecture." A Keynote Talk, PARLE Conference, Eindhoven, The Netherlands, June 15, 1987.

17. Arvind. "Two Fundamental Issues in Multiprocessing." DFVLR - Conference 1987 on Parallel Processing in Science and Engineering, Bonn-Bad Godesberg, W. Germany, June 25, 1987.

18. Iannucci, R. A. "Multiprocessor Emulation Facility: Retrospective." DARPA/IPTO meeting held at LCS, December 18, 1986.

19. Iannucci, R.A. "Dataflow Computer Architecture: an Introduction." Bolt, Beranek and Newman Laboratories, Cambridge, MA, March 2, 1987.

20. Iannucci, R.A. "Dataflow Computer Architecture." DSD Kingston Laboratory, IBM Corporation, Kingston, NY, March 2, 1987.

21. Jagannathan, S. "Environments as First-class Objects." 14th Conference on Principles of Programming Languages, Munich, Germany, January 1987.

22. Jagannathan, S. "Parallelism, Persistence and Meta-cleanliness in the Symmetric Lisp Interpreter." 1987 SIGPLAN Conference on Interpreters and Interpretive Techniques, St. Paul, MN, June 24-26, 1987.

23. Nikhil, R.S. "Functional Database System." IEEE Hyderabad Chapter, Hyderabad, India, July 1986.

24. Nikhil, R.S. "I-structures: Data Structures for Parallel Computing." Department of Computer and Information Sciences, University of Pennsylvania, Philadelphia, PA, November 1986.

25. Nikhil, R.S. "Functional Databases." Boston SIGMOD Seminar, Cambridge, MA, November 1986.

26. Nikhil, R.S. "Dataflow and Databases." DARPA/IPTO meeting held at LCS, December 18, 1986.

27. Soley, R.M. "Lisp Carries Its Own Weight." 3rd Artificial Intelligence Applications Conference, IEEE, Orlando, FL, February 25, 1987.

DISTRIBUTED COMPUTING

Academic Staff

D. D. Clark, Group Leader
J. Saltzer

Research Staff

M. Lambert

Graduate Students

D. Feldmeier
T. Ng
L. Zhang
L. Kolodney
T. Shepard

Undergraduate Students

J. Capo
A. Heybey
J. Devine
J. Lunny

Support Staff

G-L. Staton

1. INTRODUCTION

The Distributed Computing Systems Group is concerned with the issues surrounding the design of networked computing nodes. During the year a number of projects were pursued, as described below.

2. BULK DATA TRANSFER PROTOCOL

Work is continuing on the NETBLT project. Current research topics within NETBLT are (1) inventing algorithms to raise or lower protocol transmission rates dynamically in the face of network congestion or channel noise, (2) inventing methods for congestion detection that do not involve querying sources of information external to the protocol module, (3) development of a protocol (or addition to an existing protocol) that will allow NETBLT to query gateways and obtain network load information, and (4) modifications to a gateway that will allow it to monitor network load and give out information to network hosts describing current (and possibly predicted) network load.

During the year, we performed a number of performance tests on NETBLT, with positive results. Over the wideband network, we demonstrated a sustained throughput of 930 kbps, which is more than an order of magnitude improvement over previous demonstrations of transport protocols, and a channel utilization of over 90%.

A NIC RFC describing NETBLT test results over a variety of networks is in draft form and almost ready for distribution. Another RFC is being written describing possible algorithms for dynamic rate control. Finally, a paper describing the design philosophy behind bulk data transfer protocols such as NETBLT is in draft form.

3. RESOURCE ALLOCATION IN PACKET SWITCHING NETWORKS

Lixia Zhang has been working on the design of a new architecture for packet switching networks. Although packet switching technology has achieved a great success in its first 20 years, we are yet to gain further understanding of the performance control issues. A noticeable example is the network congestion problem which, despite years of research effort, has never been satisfactorily solved.

Zhang considers that the performance problems seen in today's packet switching networks reflect an architectural defect, that is, the effective traffic control components were missing from the design. Both datagram and virtual circuit architectures emerged in the early stage of packet switching era, when a full understanding of the new technology was yet to be gained through experimentation. The performance issues, as in usual cases with any new technology, were significantly exposed only after early successes made packet networks grow to enormous sizes. The future success of packet switching will depend on how well the network can control the performance.

This study has concluded that the following distinguished features of the packet switching network make the performance control challenging and interesting:

1) There are wide diversities in applications, end machines, and even network components themselves.

2) Theoretically, packet switching allows any single user an unlimited share of network resources. This dynamic sharing feature, on the one hand, provides the needed flexibility to meet various service requirements, but on the other hand, it can easily be abused to damage the network service, as we all have witnessed.

3) Computer generated data traffic has, in most cases, the flexibility of tolerating delays and channel bandwidth adjustment; some applications can even accept postponement of transmission services for a short time period with no negative effect on performance; network mail and background file transfers are such examples. However, merely blocking traffic at the network entrance is not an acceptable control, because computers work well only when being explicitly informed of what to do; they do poorly in coping with unexpected abnormal situations.

4) The capacity of a packet switching network should be measured in rates, and the sharing among users should also be controlled by rate.

Zhang proposes the following solutions:

1) Good performance can be achieved only through effective control. The network should build performance supporting mechanisms into the architecture, which include (a) a user interface with service specifications so that applications can express their requirements in a quantitative manner; (b) network data forwarding resource management that allocate and monitor the network usage; and (c) a control mechanism that dynamically adjusts network traffic with end users.

2) The network should apply an average-rate flow control on the so described bursty and random data traffic, based on the statistical data flow information provided in user service specifications.

3) An overall good performance can be achieved only through collaborations between the network and the application designs. We should design applications with controllable data generations.

4. GATEWAYS

Gateways interconnect computer networks, but often gateway performance is an order of magnitude less than that of the connected networks. Network transparency is increased if the gateways have a throughput as high as that of the attached networks, so that hosts can communicate at full network speed regardless of network location. A

way of improving gateway performance is to increase the speed of routing-table lookups. This routing table is a list of destination/next-hop pairs used for packet forwarding. Performance of the routing table lookup can be increased by placing a cache in front of the routing table. A cache will increase performance if there is a locality of reference of internet packet addresses; earlier network measurements by Feldmeier, Jain and Routhier suggested that this was likely. Models of various types of caches were constructed and the performance of these caches was evaluated by driving the cache models with data derived from network measurements. Thus, the performance of gateways with caches could be assessed in an existing network without actually implementing a cache on a gateway. The results demonstrate that internet packet addresses have reference locality and that a cache could indeed improve gateway performance.

5. MAM: A SEMI-AUTOMATIC DEBUGGING TOOL

During the last year Larry Kolodney completed a Master's Thesis entitled, **MAM: A Semi-Automatic Debugging Tool.** This work was done as part of the 6A program, jointly at GenRad and MIT. The problem addressed in the thesis was to provide distributed debugging tools in an environment where logging of all network activity occurred. Therefore the debugging tools could be postmortem in nature. He developed a language and mechanism for defining event abstractions of network activity, where the base events were messages of various types being sent to processes. The work included a mechanism for recognizing "near-misses" of events, to highlight potential sources of bugs. The work was innovative both in the area of abstraction of events for debugging, allowing for the definition of events that are not quite correct, as well as the display tools provided to the programmer for viewing and analyzing problems. The thesis was completed in January.

6. PCMAIL DISTRIBUTED MAIL SYSTEM

Work has largely finished on the Pcmail Distributed Mail System project. Final enhancements included bulletin-board capabilities and an interface to the Network News Transfer Protocol, allowing USENET network news to be captured and stored in Pcmail bulletin-board mailboxes. The mail system is in use by about 10 people here at the Laboratory for Computer Science, and by other groups around the country. Currently available software includes a mail server for Unix machines, as well as client software for both Unix and MS-DOS machines. A paper describing the philosophy behind Pcmail is forthcoming.

Publications

1. Feldmeier, D.C. "Statistical Monitors for Local Area Networks." *Proceedings of the 11th Conference on Local Computer Networks*, Minneapolis, MN, October 1986.

2. Feldmeier, D.C. "Traffic Measurements on a Token Ring Network." *Proceedings of the Computer Network Symposium*, Washington, DC, November 1986.

3. Sollins, K.R. and D.D. Clark. "Distributed Management." *Proceedings of the IFIP WOG6.5 International Working Conference on Message Handling Systems*, Munich, Germany, April 1987.

4. Zhang, L. "Why TCP Timers Don't Work Well." *Proceedings of Symposium on Communication Architectures and Protocols*, August 5-7, 1986.

5. Zhang, L. "Some Thoughts on the Packet Network Architecture." *Computer Communication Review*, 17, 1&2, (January/April 1987).

6. Zhang, L. "Congestion Control in Packet-Switched Computer Networks." *Proceedings of the Second International Conference on Computers and Applications*, Beijing, China, June 1987.

7. Zhang, L. "How to Build a Gateway." *Proceedings of the Second International Conference on Computers and Applications*, Beijing, China, June 1987.

Theses Completed

1. Kolodney, L.K. "MAM: A Semi-Automatic Debugging Tool for Distributed Programs." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, January 1987.

2. Lunny, J.F. "A Reliable Server Protocol and a Disk Manager Protocol." S.B. thesis, MIT Department of Engineering and Computer Science, Cambridge, MA, May 1987.

3. Ng, P. "Long Atomic Computations." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, August 1986.

Talks

1. Clark, D.D. "The 1, 5, 20 Year Goals of Internet Research." TCP-IP Vendors Workshop, Monterey, CA, March 1987.

2. Clark, D.D. "What's Wrong With Distributed Systems?" 2nd European SIGOPS Workshop, Amsterdam, Netherlands, September 1986.

3. Clark, D.D. "Future Directions in Networking." Xerox Corporation, November 1986.

4. Feldmeier, D.C. "Statistical Monitors for Local Area Networks." 11th Conference on Local Computer Networks, Minneapolis, MN, October 1986.

5. Feldmeier, D.C. "Traffic Measurements on a Token Ring Network." Computer Network Symposium, Washington, DC, November 1986.

6. Zhang, L. "EGP Miscellaneous." Second Internet Engineering Task Force Meeting, Ann Arbor, MI, July 1986.

7. Zhang, L. "The Internet Traffic Measurement." Second Internet Engineering Task Force Meeting, Ann Arbor, MI, July 1986.

8. Zhang, L. Invited panelist at the First TCP/IP Vendor's Workshop, Monterey, CA, August 1986.

INFORMATION MECHANICS

Research Staff

T. Toffoli, Group Leader

Graduate Students

N. Margolis

P. Tamayo

Undergraduate Student

D. Petty

Support Staff

T. Cloney

D. Zaig

1. INTRODUCTION

In the past year we have (1) published a major book, (2) organized a conference, (3) completed software and documentation for CAM-6 (including a 250-page User's guide), (4) revised and essentially finalized the design of CAM-7, and started working on an early prototype of it, (5) completed a Ph.D. thesis, and made substantial progress in three new areas of research, namely (6) locally-additive information flow in near-equilibrium system, (7) Lorentz transformations of parallel computing processes, and (8) a new technique for pattern recognition by cellular automata, called *texture-locked loop*.

We are almost through with a huge backlog of administrative and service duties, most of them concerned with scientific and technical support of cellular automata machines. While we have found time to do a good amount of original research, publication of results has suffered -- and will receive priority in the months to come.

2. THE BOOK

The publication of the book *Cellular Automata Machines -- A New Environment for Modeling* (MIT Press) [16], represents a milestone in our effort to transfer to the scientific community the methodology of modeling by cellular automata and the technology of cellular automata machines.

3. CA,'86

The conference CELLULAR AUTOMATA '86, held at MIT on June 16--19, 1986, and organized by Charles Bennett, Tom Toffoli, and Stephen Wolfram, has brought in about 150 participants, of which about 80 have given contributions in the form of lectures, posters presentations, demos, etc. A collection of extended abstracts has been published as an LCS Technical Memo [3]; selected full-length papers (including some by our group [5][8]) have been published in two issues of the journal *Complex Systems*.

4. THE CAM-7 MULTIPROCESSOR

We have made a major revision [12] in the design of CAM-7 -- a large cellular-automaton multiprocessor which will be especially useful for fine-grained models of physical systems; this revision allows one to achieve arbitrary neighborhoods by software methods rather than cumbersome hardware multiplexing. The target configuration will process 100 Giga-events/sec (where an *event* is the updating of a 1-bit cell as a function of up to 16 neighboring bits), and handle an array of 512 by 512 by 512 by 16 sites. The architecture is modularly expandable.

We intend to go ahead with construction as soon as we are free of other engagements and the funding prospects are a little clearer. We have started working on an early prototype of the basic module.

5. FLUID DYNAMICS

The idea of modeling fluids by means of cellular automata ("lattice gas" models), introduced by Pomeau in 1973 and independently rediscovered by us a few years later, has recently caught the attention of the scientific community. One of the stimuli for the more recent developments has been the availability of cellular automata machines -- ideally suited for simulating lattice gases. Fair and informative mentions of this topic have recently appeared in *The New York Times* [6] and *Physics Today* [9].

While hydrodynamics experts have been working on a number of issues concerning the immediate applicability of this method to the modeling of real fluids, we have concentrated on more fundamental questions: why the method should work at all, what are its connections with differential equations, with analytical mechanics, and with information theory. We have proved a number of theorems that establish a bridge between mass waves and information waves, and point at at interesting generalization of earlier results.

In particular, we have arrived at an explicit and complete characterization of equilibrium for a class of systems having computational-universal capabilities. Previously, such characterizations were known only for trivial systems (such as the Bernouilli flow). Moreover, we have proven that in this class of systems the information carried by small perturbations about equilibrium flows as a *locally-additive* conserved quantity -- and thus takes on a surprisingly material aspect. A paper on this subject [15] was presented at the *Symposium on Basic Concepts in Quantum and Stochastic Transport* (IBM Research, June 19, 1987).

6. PHYSICS AND COMPUTATION

A graduate student, Joe Herzgovic, has been analyzing a system that constitutes a good paradigm for the reduction of mechanical quantities (force, energy, etc.) to quantities of an information-theoretical nature. This system consists of a bit-string embedded in a reversible cellular automaton; macroscopically, it yields a harmonic oscillator in which the restoring forces remain linear even for finite-amplitude displacements.

Another graduate student, Mark Smith, has been working with Tom Toffoli and Tom Cloney on the behavior of lattice gases under conditions of relativistic drift velocities. It is known that such behavior departs from Galilean invariance (and thus violates the Navier-Stokes equation). We hope to show that this departure is in the direction of Lorentz invariance, and thus closer to physics than the Navier-Stokes equation itself.

Norman Margolus has successfully defended his Ph.D. dissertation on "Physics and Computation" [11]. His work on quantum computation has been presented at the conference "New Ideas and Techniques in Quantum Measurement Theory" [10].

7. TEXTURE-LOCKED LOOPS

Tom Toffoli has obtained rewarding preliminary results with a new method for pattern recognition using cellular automata [14]. This method is is capable of reliably recognizing patterns characterized by long-range correlations, even when obscured by extremely large amounts of noise -- provided that the pattern themselves are of the kind that can be obtained by iteration of short-range processes. This method is next-to-useless for recognizing objects such as faces or tanks -- which are produced by deep transformational grammars; on the other hand, it promises to be very efficient in recognizing "natural" features such as clouds, mountain ranges, rivers, coastlines, urban agglomerations, etc.; the identification of such features constitutes, of course, a large fraction of the routine preprocessing of satellite pictures.

References

1. Bennett, C., N. Margolus and T. Toffoli. "Energy Encoding vs. Spin Encoding in Ising Spin Models." submitted for publication, 1987.

2. Bennett, C. "Reversibility and Computation." *Symposium on Basic Concepts in Quantum and Stochastic Transport*, IBM Research, June 19, 1987.

3. Bennett, C., T. Toffoli and S. Wolfram (ed.). "Cellular Automata '86 Conference." MIT/LCS/TM-317, MIT Laboratory for Computer Science, Cambridge, MA, 1986.

4. Califano, A., N. Margolus and T. Toffoli. *CAM-6 User's Guide*, MIT, Cambridge, MA, 1987.

5. Cloney, T, E. Goles and G. Vichniac. "The *3x+1* Problem: A Quasi-Cellular Automaton." *Complex Systems*, 1, (1987), 349-360.

6. Gleick, J. "Fluid Math Made Simple -- Sort Of." *The New York Times*, (April 19, 1987).

7. E. Goles and G. Vichniac. "Lyapunov Functions for Neural Networks with Parallel Iterations." *Neural Networks for Computing 1986*, J.S. Denker (ed.), American Institute of Physics, (1986).

8. Hartman, H., P. Tamayo and W. Klein. "Statistical Mechanics of Inhomogeneous Cellular Automata." *Complex Systems*, 1, (1987), 245-256.

9. Kadanoff, L. "On Two Levels." *Physics Today*, (September 1986), 7-9.

10. Margolus, N. "Quantum Computation." *Annals of the NY Academy of Science*, 480, (1986), 487-497.

11. Margolus, N. "Physics and Computation." Ph.D. dissertation, MIT Physics Department, Cambridge, MA, 1987.

12. Margolus, N. and T. Toffoli. "Cellular Automata Machines." *Modern Approaches to Large Computer Systems*, Santa Fe, NM, (1986).

13. Porter, K. *CAM-6 Hardware Manual*. Systems Concepts, San Francisco, CA, 1987.

14. Toffoli, T. "Texture-Locked Loops -- Long-range Optimization by Local Mechanisms." *Neural Networks and Neurocomputers*, Raleigh, NC, 1987.

15. Toffoli, T. "Locally-additive Information Transport." *Symposium on Basic Concepts in Quantum and Stochastic Transport*, IBM Research, June 19, 1987.

16. Toffoli, T. and N. Margolus. Cellular Automata Machines: A New Environment for Modeling. Cambridge, MA, MIT Press, 1987.

MERCURY

Academic Staff

D. Gifford
B. Liskov
S. Ward
W. Weihl
R. Zippel

Research Staff

T. Bloom
D. Clark
I.Greif
R. Scheifler
K. Sollins
L. Shrira

Graduate Students

B. Ben-Zvi
A. Xu
Thu Nguyen
S. Zanarotti

Support Staff

J. Doherty
A. Rubin
G-L. Staton

Visitors

S. Blau
B. DeDecker
J. Emer

1. INTRODUCTION

Mercury, previously known as the Common System, will be a system to support computing in a heterogeneous environment. The high level goals of the project are to permit program invocation across language boundaries, as well as protect and sustain the heterogeneity inherent in the choices of various languages, operating systems, and hardware. Furthermore, it is our intention to build the system in order to validate the work. The report last year discussed these high level goals and issues. This report will only address work that has been done since that report.

This has been the first full year of work on Mercury. The work has mostly been in the form of detailed study and analysis of alternatives leading to decisions about the semantic models that are required for various mechanisms. Recently, we have reached points in several parts of the project, where implementations could begin. Some of these are viewed as a simple first phase, undertaken to get "something up and running", in order to gain experience with the mechanisms. With that experience, these parts will be replaced with better ones. The semantic model can be viewed in two parts, communications semantics, and abstract transmissable data. The implementation efforts can be broken into transport layers, language veneers (of which there are three, Argus, Lisp, and C), and clients. The work is documented in over 20 internal working notes, with several more in progress. In addition, several papers have been prepared for submission for publication. The working notes provide much more detail than can be included here. This is only a summary.

2. MODEL

A number of basic decisions in the project concern our underlying model of communication, the semantics of communication and the nature transmissable objects. Mercury takes the perspective that computation is distributed over *modules*. A module is an instance of a service interface. It resides at a single location (host) and consists of a set of single operation *ports*, procedures that can be invoked remotely. The granularity of registration of publicly accessible entities is the module. Within a module, we also saw the need for multiple threads of control. Each such thread is called an *agent*. Therefore a module contains a set of agents, which may make remote calls. A module can be invoked by remote agents through a set of ports.

The model we propose for communication is the *stream*. Each port contains a stream identifier. A stream is the set of requests from an agent to a set of ports having the same stream identifier. A stream guarantees that all requests will be delivered in the order sent and that any responses will be delivered in that same order. An agent can set several flags as part of invoking a port. For instance, the agent can specify whether a reply is required or not, or whether the request should be handled immediately or not. These features provide flexibility in the semantics of invocation. If the invoker does not expect and will not collect a response, the semantics of the invocation is that of message passing, a simple send. In this case immediacy is irrelevant, since no response is

awaited. In contrast, if a reply is expected and the request should be handled immediately, the semantics of traditional rpc is available. But, in addition, there is another interesting option. The invoking agent might make a series of requests, indicating that the responses should be returned, but that there is no urgency. The requesting agent will not wait, but will collect the responses later.

Streams provide an interesting set of advantages over previously provided semantics. First, from the language level, they can be used as a simple form of futures, as proposed in Halstead's MultiLisp. They allow for inclusion of invocations the results of which will only be needed at a later time, permitting a more relaxed form of evaluation. Second, from the perspective of the lower layers of communication protocols, they permit more efficient use of communications mechanisms, when either responses are not needed, or requests and replies can be delayed. The possibility of delaying the processing of remote invocations allows for batching of message handling. Finally, in terms of ports, there can be a single type of port, for which replies are always generated, but on transmitted when needed. From the perspective of the service programmer, a simpler model can be used, since there is only one form of invocation.

The project has a commitment to supporting transactions and atomicity. This work is still in its formative stages, although it is clear that the semantics of message passing will have a strong influence on atomicity.

Given the model of streams, it has also been important to reach agreement on the nature of transmissable objects. In keeping with the goal of the project to support heterogeneity and therefore not to take the route of the intersection of facilities in the languages, but rather something closer to the union, the project has a commitment to permitting transmission of data abstractions. The entities being transmitted are values; they do not have operations, and what is of primary importance is the representation of values. Therefore, their types are termed *valuespaces* or *vpsaces*. There are an agreed upon set of base vspaces including such vspaces as BOOL, BYTE, CHAR, INT16, INT32, etc. In addition there are vspace generators such as STRUCT (tagged union), ONEOF, and PTR (a generator for supporting sharing). In addition, there are mechanisms still under discussion for defining abstractions. The issues of polymorphism and type inheritance are not supported at present, although they may be investigated further at a later date. There is much remaining work in determining how and where both interfaces for modules and vspaces will be defined. First, in order to define them, there must be a language, and then there must be translation mechanisms between that language and the local languages.

The underlying model proposed here is that modules form the basis of activity. Each module, residing at a single site, will contain one or more threads of control, called an agent, and can be invoked remotely through a set of single operation ports. Ports for a single module are grouped by stream identifier. The set of invocations from an agent to a set of ports having the same stream identifier form a stream. All invocations on a stream are sequenced. Finally, the objects that are transmitted on a stream are extensibly typed by vspaces.

3. IMPLEMENTATIONS

The implementation efforts at present are focused on providing the mechanisms needed to support initial remote invocations in order to build the first applications. Therefore the first areas of effort are the transport layer, to allow for actual transmission of requests and replies, and the language veneers, so that the communication mechanisms are accessible from the languages in question. This has led to four separate activities, in the form of four subgroups within the project, one each to design the transport protocols, and each of the three language veneers.

The transport group has defined the protocols needed for lower level connections between pairs of modules and on top of that shorter-lived streams from individual agents at one or the other module. In addition to the basic creation and deletion of connections and streams, the work has had to focus on exceptions, failures, and quiescence, and when information no longer need be retained. One of the issues here is retaining information for very long-lived but seldom used connections or streams. Another issue that is also getting significant attention is authentication and security, not only how to provide it, but also how to guarantee that it does not cause inefficiencies when it is not needed. The model of a client invoking a service that in turn must invoke a series of other services in order to complete an operation on the client's behalf has highlighted the problem of cascading authorization. This is only one example of cascading. Problems also arise when it is important for sequencing of actions to know when a service module is invoked by a client, which streams are being used in further remote invocations. Similar problems arise in cascaded subactions in a transaction. All are undergoing further work.

The present situation in the transport layer work is that the protocols and data representations are in a form that the first phase of implementation is proceeding in both the Unix and Lisp machine environment. In order to decrease the amount of time and work needed to complete this first phase, the lower level connections are being built on TCP, which itself does most of the work needed to set up and maintain connections. The problems with TCP are that it does not have the idea of modules, so our connections must be a slight extension of TCP connections, and it may be very inefficient for our needs. In the long run, TCP will be replaced with our own transport layer.

In each of the three languages, the mappings from the Mercury ideas such as module, agent, and ports have been defined. In addition, in order to support remote invocations, there must be mechanisms for translating local objects into vspaces. Initially, other than for the base vspaces and simple local types, these translations will be done by the programmer, although the intention is that in the future, as much of this as possible will be automated, simplifying the task of the applications programmer. Implementations are underway in C and Lisp, and planned for Argus.

In addition to these implementation efforts, one service for Mercury is underway, an

archive service, and others are planned. The next two efforts will be a front-end for the Community Information Service and a catalog, in order to be able to register and then later find the ports to be invoked to reach a particular module. The archive service is the Master's thesis of Stan Zanarotti. The work here is first to define and prototype a publicly accessible archive service, and then within that to study further a set of proposals for naming mechanisms that meet the naming requirements for such an archive service. The Community Information Service and catalog work are two of the projects planned for this summer.

Our short-term future plans are to build two demonstrations within the next year. The first of these will use remote invocations, with hand-crafted interfaces. This will require existence of the first phase of the transport layers, the language veneers, a simple catalog to find ports, simple authentication as part of the transport layers, and transactions. The later application will demonstrate more automatic interface generation, and will therefore require the design and implementation of the language in which interfaces and vspaces are defined, as well as the library in which they will be stored, on top of the functions needed for the earlier application. Simultaneously, work will continue in many of the areas in which only preliminary work has been done, and there will be work on both a file service and object repository as well as debugging tools.

Publications

1. Sollins, K.R. "Naming Problems in the Common System." *Proceedings of the Distributed Systems Architecture Board Task Force on Naming*, February 1987.

Talks

1. Liskov, B. "Support for Heterogeneous Computing." NEC Corporation, Tokyo, Japan, October 1986.

2. Liskov B. "Communication in a Heterogeneous System."
   Texas Instruments, Dallas, Tx, February 1987;
   MIT Laboratory for Computer Science, Cambridge, MA, March 1987;
   Stanford University, Stanford, CA, March 1987;
   Georgia Institute of Technology, Atlanta, GA, April 1987;
   Dept. of Information and Computer Science, University of California, Irvine, May 1987.

3. Liskov, B. "Support for Programming in Heterogeneous Systems."
   NYNEX, White Plains, NY, June 1987;
   Annual Meeting, MIT Laboratory for Computer Science, June 1987.

4. Sollins, K.R. "Naming Problems in the Common System." Distributed Systems Architecture Board Task Force on Naming, February 1987.

5. Sollins, K.R. "Common System." University of California at Davis, February 1987.

6. Sollins, K.R. "Mercury: Heterogeneous System."
   Advanced Networks System Architecture Project (ANSA), Cambridge, England, April 1987;
   Cambridge University, Cambridge, England, May 1987.

7. Sollins, K.R. "Type and Service Registry in the Common System." IEEE Computer Society Workshop on Design Principles for Experimental Distributed Systems, November 1986.

8. Weihl, W. "Communication in the MIT Common System."

Brown University, Providence, RI, March 1987;
DEC Systems Research Center, March 1987;
IBM Almaden Research Center, March 1987;
University of Washington, March 1987.

PARALLEL PROCESSING

Academic Staff

R. Halstead, Group Leader

Graduate Students

L. Bagnall
E. Bradley
Y. Jang
J. Loaiza
M. Ma
J. Miller
D. Nussbaum
P. Nuth
R. Osborne

Undergraduate Students

D. Becker
B. Callaghan
D. Chamberland
J. Ferro
J. Gilson
S. Herron
K. Johnson
K. Lu

Support Staff

S.M. Hardy

Visitors

T. Fujita
G. Thaker
I. Vuong

1. INTRODUCTION

The 1986-87 academic year marks the first full year of operation of the Parallel Processing Group, whose ultimate goal is learning how to build parallel processors that can be programmed for general-purpose applications. Meeting this challenge requires progress on three fronts: languages, algorithms, and architectures. A major focus of our effort is in the direction of general-purpose symbolic computing. Accordingly, our current programming language work centers on MultiLisp [14][11], a dialect of Lisp including the `future` [5][19] construct for parallel computing. As a vehicle for experimentation with parallel programming, we have built *Concert* [13][2], a 32-processor shared-memory multiprocessor. Concert has been several years in the making, but the past year saw the final emergence of a full-size and reliably working Concert. MultiLisp has been implemented on Concert and has been used to experiment with implementations of `future` and to support the generation and performance measurement of a library of representative parallel programs.

The ways in which MultiLisp helps or hinders the expression of these algorithms furnish guidance on how to improve MultiLisp. At the same time, as more programs are written, a library of "benchmark" programs written in MultiLisp is gradually accumulating. These benchmarks are the calibration or "reality check" for proposed architectures: whenever a question arises as to whether the benefits of some proposed architectural feature justify its cost, simulations using the benchmarks can supply quantitative data to help provide the answer. It is important for many of the benchmarks to be sizable programs, because the ultimate use of high-performance parallel architectures will be for large problems, which often have qualitatively different properties from small ones (for example, small programs are apt to show more locality of reference).

Our work on benchmarks is discussed later, after brief descriptions of Concert and MultiLisp. MultiLisp is built on Concert using three levels of implementation: (1) the Concert multiprocessor hardware, (2) the utility (or "operating system") level, and (3) the MultiLisp implementation itself.

2. THE CONCERT MULTIPROCESSOR

The Concert multiprocessor consists of eight clusters of processors, as shown in Figure 7-1. connected by a segmented bus in the shape of a ring, known as the *RingBus*, which is controlled by a central arbiter [13][2]. Each cluster contains a block of globally accessible memory and a MultiBus tying together a small number of MC68000 processors and their private memories. The processors and memories are commercially available dual-ported boards that each have one MultiBus port and one High Speed Bus port. The High Speed Bus is a single-master bus designed to operate somewhat faster than the MultiBus.

The structure of a representative Concert cluster is shown in Figure 7-2. Arrowheads

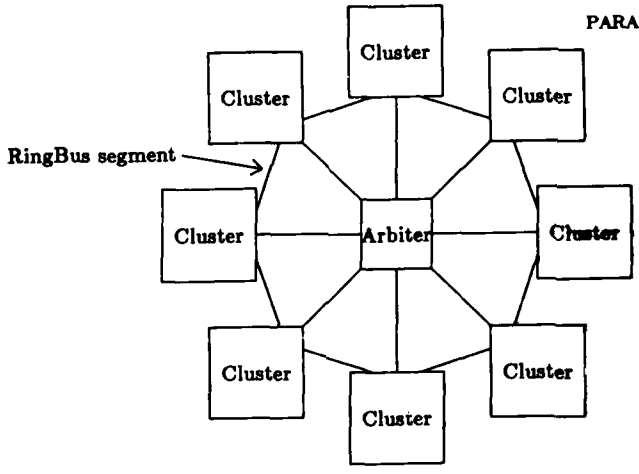


Figure 7-1: RingBus Multiprocessor System

on the vertical lines in this figure show the nature of the associated bus connections: a module is potentially a master of the bus if the arrow points toward the bus, and potentially a slave if the arrow points away from the bus. MultiBus arbitration circuitry ensures fair access to the MultiBus by each potential MultiBus master.

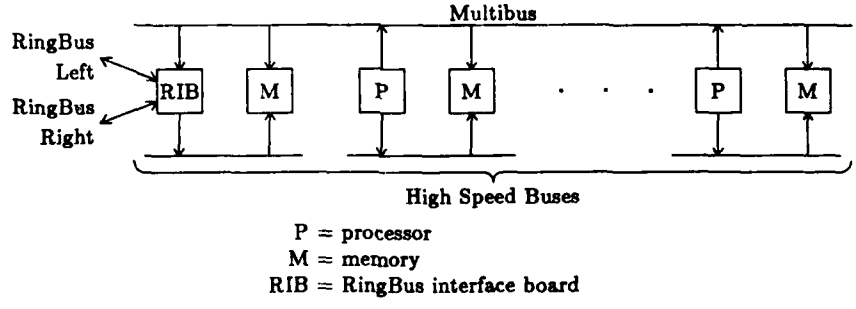


Figure 7-2: Structure of a Typical Concert Cluster

If the majority of memory accesses from each processor are directed toward its own tightly coupled memory using its High Speed Bus, enough traffic is unloaded from the MultiBus so it can be shared by several processors without significant performance degradation. When running MultiLisp with four processors per cluster, each cluster's MultiBus is busy from one-quarter to one-half of the time, and only slight performance degradation occurs.

An individual cluster's MultiBus is connected to the RingBus by a RingBus interface board, or RIB. The RIB responds to MultiBus operations like a memory module: whenever a memory read or write operation on the MultiBus has an address marking it as a global memory access, the RIB responds to the request and translates it into the appropriate RingBus operation. Each RIB also manages a portion of the global memory pool, represented in Figure 7-2 by the memory module connected to the RIB via a High-Speed Bus. Thus each RIB also listens to the RingBus for accesses to the portion of global memory connected to that RIB. A 16-bit access on the High Speed Bus, MultiBus , or RingBus takes approximately 500 nanoseconds, one microsecond, or two microseconds, respectively, in the absence of contention.

The RingBus supports read and write transactions by which processors may access global memory. All accesses to global memory are under the control of the RingBus arbiter, which periodically examines and grants requests from the clusters. The arbiter can cause multiple segments to be connected together to perform a single memory access, if a processor attempts to access a memory location several segments distant along the RingBus. Several processors may be carrying out independent accesses to different blocks of global memory, provided they do not require overlapping sets of RingBus segments. Studies show the performance of a RingBus to be intermediate between that of a crossbar and a simple shared bus [13][26].

Led by the heroic efforts of D. Nussbaum and P. Nuth, we finally completed the construction of Concert and commissioned a reliable, full-sized, eight-cluster system during the past year. J. Loaiza assisted by greatly improving the *Diag* diagnostic exerciser program, which is now run routinely whenever Concert is unloaded to watch for developing hardware problems.

Another version of Concert, which replaces the RingBus with a different global communication mechanism called the GCM, has been built at the Advanced Technology Division (ATD) of the Harris Corporation's Government Systems Sector. The past year has seen the final completion of debugging of ATD's GCM-based Concert, which is now running reliably with eight clusters containing eight processors each. A continuing collaboration with ATD has yielded, among other dividends, several hardware modules that may be plugged into Concert for performance monitoring. During the past year, personnel from ATD assisted D. Nussbaum and P. Nuth in installing the *Beckerboard*, which provides a visual display of processor status and MultiBus loading in each cluster, and the *SysM*, which provides a facility for measuring the performance of different parts of a program under test in a way that only slightly affects the performance of that program.

3. THE CONCERT SYSTEM PROGRAMMING ENVIRONMENT

Low-level Concert "system programs" are written in C. A utility subroutine package provides access to file and network I/O facilities supported by server processes that (normally) run on one processor dedicated to this purpose. The utility subroutine

interface is a subset of the UNIX "standard I/O library" interface and includes the ability to read, write, and create files on the Concert file system or remotely across the network. Although the utilities are designed to provide the needed facilities, they are also designed to be unobtrusive, so they will not generate overhead to disrupt the performance of multiprocessor programs running on Concert.

The past year saw further development of Concert's Chaos network interface to a point where it offers a reliable remote file access service, and the development of *Maestro*, a program to load and oversee other programs running on Concert, by J. Loaiza and G. Thaker. Another improvement to the Concert user environment was an input editor for commands typed to Concert, implemented by I. Vuong.

4. MULTILISP

MultiLisp is an extended version of the Lisp-like programming language Scheme [1] that allows a programmer to specify concurrent execution. The default in MultiLisp is sequential execution; concurrency can be introduced into a MultiLisp program by means of the `future` construct. The form (`future` *X* `0`) immediately returns a *future* for the value of *X* and creates a task to concurrently evaluate *X*, allowing concurrency between the *computation* of a value and the *use* of that value. When the evaluation of *X* yields a value, that value replaces the future; we say that the future *resolves* to the value. Any task that needs to know a future's value will be suspended until the future resolves.

We say that a task *T examines*, or *touches*, a future when it performs an operation that will cause *T* to be suspended if the future is not yet resolved. Most operations, *e.g.*, arithmetic, comparison, type checking, *etc.*, touch their operands (any operation that is *strict* in an operand touches that operand). However, simple transmission of a value from one place to another, e.g., by assignment, passing as a parameter to a procedure, returning as a result from a procedure, building the value into a data structure, *etc.*, do *not* touch the value. Thus, many things can be done with a future without waiting for its value. `Future` induces some patterns reminiscent of those found in graph-reduction architectures based on "lazy evaluation" [16][18] [29], yet in other ways `future` creates a style of computation much like that found in data flow architectures [3][10] [8].

The team of J. Loaiza, G. Thaker, D. Nussbaum, and P. Nuth removed the last lingering synchronization bugs in the MultiLisp garbage collector during the summer of 1986. Consequently, Concert and MultiLisp are now both operational and heavily used for developing new programs as well as for measuring properties of existing ones. Experience has generally been encouraging, and has also helped focus attention on a few areas where the MultiLisp design needs further thought. One of these areas is debugging and exception handling. A preliminary approach to these problems using *exception values* is currently implemented [12]. Another area where further thought is needed is *speculative parallelism* [14] -- the creation of tasks whose results may

subsequently be rendered irrelevant by the results of other computations. Some ways of dealing with this problem have been implemented in the MultiScheme system described below, but we have not yet been able to evaluate how completely the MultiScheme solutions address the speculative parallelism problems that may arise in realistic application programs. R. Osborne has begun studying the issues involved in supporting speculative parallelism in MultiLisp.

5. MULTISCHEME

J. Miller and L. Bagnall collaborated with Bolt, Beranek, and Newman, Inc. (BBN) in developing an alternative implementation of parallel Lisp with futures, known variously as *MultiScheme* and *Butterfly Lisp* [24][7], which runs on the BBN Butterfly Machine. This work has explored a number of new areas within MultiLisp's value-oriented parallel processing model. MultiScheme includes a parallel garbage collection algorithm which not only recycles the (unfortunately) finite memory resource but removes speculative computations that are no longer needed. The garbage collector also acts as a centralized provider of a number of user-visible services formerly provided in a far more *ad hoc* manner.

The MultiScheme work has also explored ways in which the underlying support for the `future` operation can be exploited to extend the expressive power of Scheme. In the process we have discovered a host of uses for "placeholders" (which implement futures) even in the serial language. In addition, by separating these placeholders from the underlying tasks which perform the computation, a number of alternative computation models can be embedded in MultiScheme. A number of applications have been written utilizing novel combinations of this same underlying support.

Finally, the MultiScheme system is based around a core "scheduler" written in Scheme. By recasting this portion of the implementation in a user-accessible form, a number of extensions have been easily built and tested. Among the more successful additions is a new operation, called "disjoin," which is similar in spirit to McCarthy's "amb" operator. Another useful extension has been the creation of "mutable futures." These allow a value to be tentatively assigned to a placeholder and then subsequently changed. This was the basis for a simulation of the ParaTran system [17], an experimental architecture designed to support parallelism in the presence of side-effects, which was reported on in last year's progress report.

The resulting system continues to evolve under the direction of Prof. Sussman's Scheme group in the AI Lab and the BBN Butterfly Lisp Project. The next steps are expected to be the integration of an efficient Scheme compiler with the existing Butterfly implementation of MultiScheme, followed by porting the system to the newly available Butterfly Plus processor.

MultiScheme has been distributed rather widely as the installed base of Butterfly machines grows, and already a number of projects in various research organizations are using it.

6. BENCHMARKS

Several application programs have been and are being written and studied, including sorting [11], simulation of logic circuits [6], event-based simulation [23], part of a speech recognition system [22], semantic net retrieval [4], a subset of Prolog not including "cut" [27], a kernel of EMYCIN [21] (developed at MCC), the Boyer benchmark from the Gabriel benchmark set [9], polynomial manipulation, traveling salesman, parallel parsing of Lisp expressions, and the MultiLisp compiler itself. Two of these programs underwent significant development during the past year: event-based simulation (by M. Ma) and the Boyer benchmark (by R. Halstead and R. Osborne).

7. PERFORMANCE STUDIES

A program to be executed by Concert MultiLisp is first compiled into a machine-level language called MCODE. MCODE is interpreted by a native-code MC68000 program. One copy of this program is located in the local memory of, and executed by, each processor in the Concert machine. Each processor manipulates various per-processor data structures, such as top-of-stack caches, also located in the processor's local memory. Shared data structures (including MCODE programs) reside in a garbage-collected heap that is distributed among the RingBus-accessible memory modules of Concert. Thus each processor makes mainly local accesses, punctuated by an occasional access to the shared heap for the next MCODE instruction or some other MultiLisp object. Notable features of Concert MultiLisp include a parallel, incremental garbage collector and an *unfair* scheduler [11].

The shared memory of Concert allows fairly efficient implementation of the sharing of objects between MultiLisp tasks; furthermore, the bandwidth to Concert's global memory is fairly well matched to MultiLisp's demands -- the observed performance degradation due to contention for global memory ranges from 0 to 15%. However, the decision to use an off-the-shelf microprocessor -- the MC68000 -- on an off-the-shelf processor board has imposed several costs that could have been avoided if we had designed and built more hardware ourselves. The fastest MCODE instruction (pushing `nil` onto the stack) takes 60-70 microseconds. Other instructions take longer -- sometimes much longer. For example, pushing the value of a local variable onto the stack takes only about twice as long as pushing `nil`, but the *FUTURE* instruction takes over two milliseconds. The chief preventable costs in the "inner loop" operations of instruction fetch and execution involve

- lack of support for manipulation of tagged data;
- lack of support for incremental garbage collection;
- limited facilities for atomic operations to memory (the MC68000 has only a test-and-set instruction that affects the most significant bit of a byte), leading to expensive protocols for atomic 32-bit reads and writes to shared memory;

- bounds checking on the "stack cache" maintained by each processor to cache locally the top few items on the stack of the currently executing task; and

- interpretive overhead for fetching and decoding MCODE instructions (however, compilation of MultiLisp into MC68000 code is unattractive, due to the length of the code that would have to be generated for the operations listed above).

Generally, the costs seem to be distributed rather widely across the different categories, so no single improvement would lead to an order-of-magnitude increase in execution speed. Therefore, incremental modifications to Concert to improve MultiLisp performance are not planned. Instead, the current focus of the Concert MultiLisp project is to use Concert to learn as much as possible about the MultiLisp language design and about MultiLisp application programs, and then use these insights to design a completely new architecture whose processors, memories, and communication elements would all be tailored for efficient MultiLisp execution.

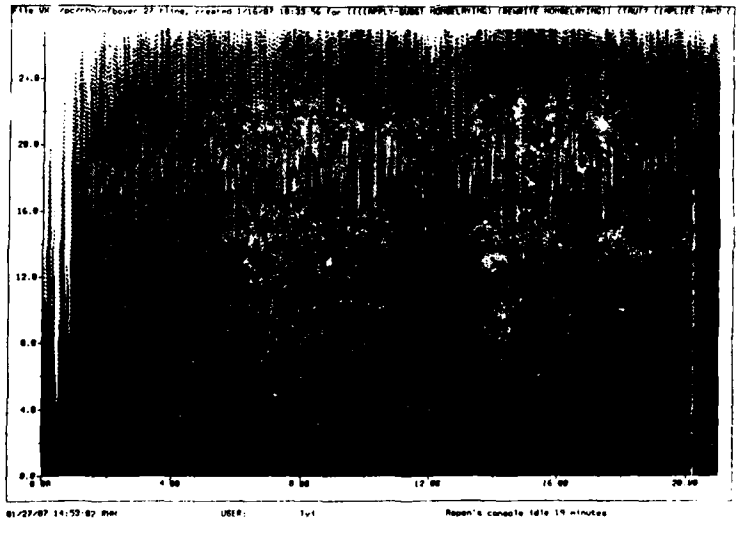


Figure 7-3: Parallelism Profile for Boyer Benchmark

These projects to refine the MultiLisp language and develop example application programs are complemented by several other activities in the Concert group. One of these other activities involves developing tools for measuring and visualizing properties

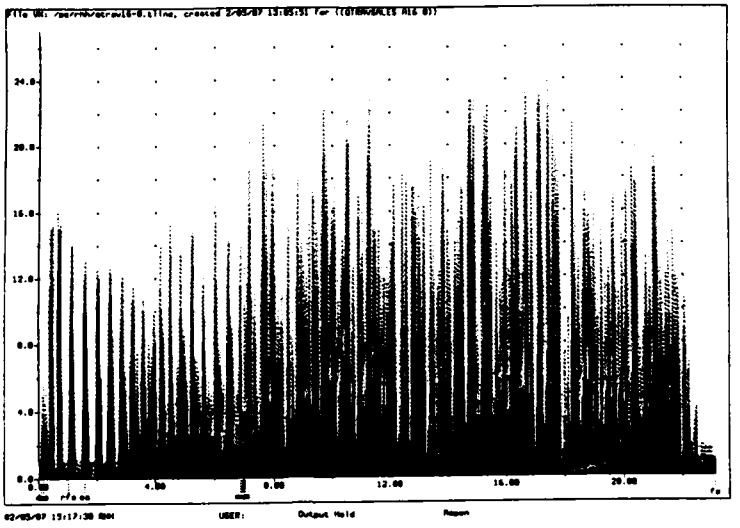


Figure 7-4: Parallelism Profile For 16-City Traveling Salesman Problem

of MultiLisp programs, such as the amount of parallelism available or the degree of locality of reference. R. Halstead and G. Thaker built a tool to produce graphical parallelism profiles for programs. Figures 7-3 and 7-4 show parallelism profiles for executing the Boyer benchmark to decide whether the expression

```
(implies  (and  (and  (implies (f x) (g x))
                      (implies (g x)  (h x)))
                (implies (h x)  (1 x)))
          (implies (f x ) (1 x))))
```

is a tautology, and for exactly solving a 16-city traveling salesman program. The horizontal axis on each profile marks the passage of time in seconds, and the vertical axis indicates the number of processors busy at a given moment. Black ink indicates "real computing" also performed in a sequential execution of the algorithm, and the shades of gray correspond to two different kinds of overhead that result from using futures. Such profiles have been useful in assessing both the amount of parallelism made available and the amount of overhead introduced by different uses of MultiLisp's parallelism constructs.

8. ARCHITECTURAL STUDIES

In pursuit of the information about program execution behavior needed for designing parallel architectures tuned for efficient execution of MultiLisp programs, two tools for measuring program behavior have been constructed. One of these is *Nusim*, an extensively instrumented re-implementation of MultiLisp which can yield detailed information about locality of reference and dynamic instruction execution frequencies [25], implemented by P. Nuth. Another tool, implemented by I. Vuong, modifies MultiLisp programs to record various statistics about themselves as they run.

Nusim simulates one processor of a multiprocessor system as a set of functional units. By running a copy of Nusim on each processor of the Concert machine, a user can emulate different parallel organizations. Nusim allows a user to vary several parameters in the implementation of MultiLisp. It can use several different strategies for scheduling tasks. Nusim can also simulate different topologies of processing nodes. Finally, a user can vary some of the internal characteristics of a MultiLisp processor in Nusim.

Nusim was used for a series of experiments on the communication load generated by MultiLisp programs. The goal was to quantify the locality of data reference of the programs, and to see how that could be improved. A specific model of a shared memory multiprocessor for MultiLisp was proposed. Five different application programs were run under Nusim, ranging from simple benchmarks to some of the larger applications yet written in MultiLisp. The number and types of data accesses made by each of these programs was measured, as was the distribution of those accesses in memory. The task scheduling algorithms used by the processors was varied, as was the topology simulated by Nusim, to see what effect that would have on communication patterns.

This research showed that MultiLisp programs exhibit some locality of data reference, as approximately one third of the data referenced by a processor is typically allocated in the local memory of that processor. Furthermore, the locality of reference can be increased by changing the manner in which processors search for executable tasks. The locality of data reference improved by 5% to 60% when processors exploited knowledge of the system topology. Finally, it is possible to improve the locality of reference by limiting the parallelism of a particular program.

R. Osborne investigated the effect of garbage collection activity on the amount of non-local communication during the execution of MultiLisp applications. This initial investigation, performed using Nusim, revealed that garbage collection activity primarily affects the variability in the amount of non-local communication. The mean amount of non-local communication was largely unaffected. Further investigation is required to confirm this finding.

I. Vuong's tool permits studying the parallelism obtained in a hypothetical MultiLisp machine. The purpose of this work is to help evaluate the relative importance of the mechanisms involved in a MultiLisp computation and hence the effort that should be

put into different parts of the architecture. This study can give some answers to questions such as how fast the intercommunication network should be, compared to the processor.

This tool was built by turning the MultiLisp (MCODE) interpreter running on Concert into a simulator. One can specify the relative duration of computing operations such as future creation, function call, *etc.*, and observe the effect on the parallelism and overall computing behavior when these parameters are varied. For example, by simulating a case where **future** is very cheap, one can experiment with increasing the parallelism of the programs by inserting more futures, thus evolving toward finer-grain parallelism. This simulator produces operation profiling and parallelism profiling in the form of time-based graphs, mean values and standard deviations. This simulator is currently being run on the various existing small to middle size programs written in MultiLisp, with the goal of extracting both application-independent and -dependent results from these experiments.

A "first cut" at the design of a processor architecture for a MultiLisp-oriented multiprocessor has been begun by T. Fujita and R. Halstead. This architecture is heavily influenced by the reduced instruction set philosophy of the SPUR multiprocessor [28] and by the multithreaded approach of the HEP-1 [20]. It features hardware tag manipulation for speed in basic Lisp primitive operations, along with multiple register sets for speed in procedure calling, process creation, trap handling, and context switching [15].

9. RELATED PROJECTS

Our collaborators at Harris Corporation continue to use their own version of Concert to pursue a research program currently centered around the SPoC (Simultaneous Pascal on Concert) programming environment. Simultaneous Pascal is a version of Pascal augmented with explicit concurrency constructs of the fork/join and do-all variety. Several recent experiments at Harris have shown the performance of Simultaneous Pascal programs on Concert to equal or surpass that of hand-coded C programs using much more primitive and lower-level synchronization constructs. Additionally, Harris continues to develop hardware for performance measurement on Concert and continues to share that hardware with us. In addition to the Beckerboard and SysM modules discussed above, a new module, the *DLA* (Degradation due to Latency and Arbitration), is currently in final debugging stages. The DLA counts memory accesses of different types to different regions of address space and also allows computation of the average access time, including arbitration and contention delays, to each region of address space. This board will allow much more precise measurement of the extent to which MultiBus or RingBus contention is affecting the performance of a program running on Concert. Eight copies of this board are expected to be available for installation at MIT during the 1987-88 academic year.

Copies of the Concert MultiLisp implementation have also been sent to certain outside organizations, notably the University of Texas and Encore Computer Corp.

10. CONCLUSION

The challenge of parallel computing requires coordinated evolution of languages, algorithms, and architectures. Lacking an established software base of parallel programs, a parallel computing research project must push back all three of these frontiers. Concert thus performs a vital support role by permitting experimentation with the MultiLisp language and with algorithms expressed in MultiLisp. This experience will lead to the specification of a new architecture more congenial for MultiLisp than Concert is. This new machine will allow experimentation with larger and more realistic applications, which, in turn, should lead to further revisions of MultiLisp and further insights regarding parallel programming. It is through such iteration between software and hardware design that we can ultimately hope to achieve genuinely cost-effective general-purpose parallel computing.

References

1. Abelson, H. and G. Sussman. Structure and Interpretation of Computer Programs. MIT Press, Cambridge, MA, 1984.

2. Anderson, T. "The Design of a Multiprocessor Development System." MIT/LCS/TR-297, MIT Laboratory for Computer Science, Cambridge, MA, September 1982.

3. Arvind, K. Gostelow and W. Plouffe. "An Asynchronous Programming Language and Computing Machine." University of California, Report TR114a, Irvine, CA, 1978.

4. Baek, H.J. "Parallel Retrieval Algorithms for Semantic Nets." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1986.

5. Baker, H. and C. Hewitt. "The Incremental Garbage Collection of Processes." Artificial Intelligence Laboratory Memo 454, Cambridge, MA, December 1977.

6. Bradley, E. "Logic Simulation on a Multiprocessor." S.M. thesis, MIT *Department of Electrical Engineering* and Computer Science, Cambridge, MA, June 1986.

7. Courtemanche, A. "MultiTrash, a Parallel Garbage Collector for MultiScheme." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, January 1986.

8. Dennis, J.B. "Data Flow Supercomputers." *IEEE Computer*, 13, 11, (November 1980), 48-56.

9. Gabriel, R. Performance and Evaluation of Lisp Systems. MIT Press, Cambridge, MA, 1985.

10. Gurd, J., C. Kirkham and I. Watson. "The Manchester Prototype Dataflow Computer." *Communications of the ACM*, 28, 1, (January 1985), 34-52.

11. Halstead, R. "MultiLisp: A Language for Concurrent Symbolic Computation." *ACM Transactions on Programming Languages and Systems*, October 1985, 501OA-538.

12. Halstead, R. and J. Loaiza. "Exception Handling in MultiLisp." *1985 International Conference on Parallel Processing*, St. Charles, IL, August 1985, 822-830.

13. Halstead, R., T. Anderson, R. Osborne and T. Sterling. "Concert: Design of a Multiprocessor Development System." 13th Annual Symposium on Computer Architecture, Tokyo, June 1986, 40-48.

14. Halstead, R. "Parallel Symbolic Computing." *IEEE Computer*, 19, 8, (August 1986), 35-43.

15. Halstead, R. "Concurrent Lisp Machines." Parallel Processing Group, MIT Laboratory for Computer Science, Cambridge, MA, March 1987.

16. Henderson, P. and J.H. Morris. "A Lazy Evaluator." *Proceedings 3rd ACM Symposium on Principles of Programming Languages*, 1976, 95-103.

17. Katz, M. "ParaTran: A Transparent, Transaction Based Runtime Mechanism for Parallel Execution of Scheme." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1986.

18. Keller, R. and F. Lin. "Simulated Performance of a Reduction-Based Multiprocessor." *IEEE Computer*, 17, 7, (July 1984), 70-82.

19. Knueven, P., P. Hibbard and B. Leverett. "A Language System for a Multiprocessor Environment." Fourth International Conference on the Design and Implementation of Algorithmic Languages, Courant Institute of Mathematical Studies, NY, June 1976, 264-274.

20. Kowalik, J.S. "Parallel MIMD Computation: HEP Supercomputer and Its Applications." MIT Press, Cambridge, MA, 1985.

21. Krall, E. and P. McGehearty. "A Case Study of Parallel Execution of a Rule-Based Expert System." *International Journal of Parallel Programming*, 15, 1, (February 1986), 5-32.

22. Lau, W. "Lexical Analysis of Noisy Phonetic Transcriptions." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, February 1986.

23. Ma, M. "Efficient Message-Based System for Concurrent Simulation." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, to appear.

24. Miller, J. "MultiScheme: A Parallel Processing System Based on MIT Scheme." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, to appear.

25. Nuth, P. "Communication Patterns in a Symbolic Multiprocessor." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

26. Osborne, R. "Modeling the Performance of the Concert Multiprocessor." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1986.

27. Solomon, S. "A Query Language on a Parallel Machine Operating System." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1985.

28. Taylor, G., et al. "Evaluation of the SPUR Lisp Architecture." *13th Annual Symposium on Computer Architecture*, Tokyo, June 1986, 444-452.

29. Turner, D. "A New Implementation Technique for Applicative Languages." *Software -- Practice and Experience*, 9, 1, (January 1979), 31-49.

Publications

1. Halstead, R. "Parallel Symbolic Computing." *IEEE Computer*, 19, 8, (August 1986), 35-43.

Theses Completed

1. Becker, D. "A Multiprocessor Emulator of a Static Data Flow Computer." S.B. thesis, Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

2. Herron, S. "A General-Purpose Simulation Environment." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

3. Nuth, P. "Communication Patterns in a Symbolic Multiprocessor." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

Theses in Progress

1. Gilson, J. "An Object Oriented Programming System in Common Lisp." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected July 1987.

2. Ma, M. "Efficient Message-Based System for Concurrent Simulation." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected January 1988.

3. Miller, J. "Multischeme: A Parallel Processing System Based on MIT Scheme." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected August 1987.

4. Nussbaum, D. "A Framework for Real-Time Graphics on a Multiprocessor." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected August 1987.

Talks

1. Halstead, R. "MultiLisp: A Language for Parallel Symbolic Computing."
   JASON Parallel Architecture Workshop, MITRE Corp., La Jolla, CA, July 1986;
   Nippon Electric Company C&C Laboratories, Tokyo, Japan, June 1986;
   Apollo Corp., Chelmsford, MA, October 1986.

2. Halstead, R. "Language and Architecture for Parallel Symbolic Computing."
   Carnegie-Mellon University, Pittsburgh, PA, March 1987;
   Alliant Corp., Littleton, MA, March 1987;
   Tsinghua University, Beijing, China, April 1987;
   Shanghai Jiao Tong University, Shanghai, China, April 1987;
   MIT Industrial Liaison Program seminar, Tokyo, Japan, April 1987.

3. Halstead, R. "Parallel Symbolic Computing Using MultiLisp."
   IBM/MIT workshop on parallel processing, Essex, CT, April 1987.
   California Institute of Technology, Pasadena, CA, April 1987.
   Stanford University Computer Science Colloquium, Palo Alto, April 1987.
   DEC Systems Research Center, Palo Alto, CA, April 1987.
   Schlumberger Palo Alto Research Laboratory, Palo Alto, CA, April 1987.
   University of California at Berkeley, Berkeley, CA, May 1987.
   University of Illinois Center for Supercomputing Research and Development, Champaign, IL, May 1987.

PROGRAMMING METHODOLOGY

Academic Staff

B. H. Liskov, Group Leader
W. E. Weihl

Research Staff

C. Chambers
D. W. Curtis
P. R. Johnson

Graduate Students

B. P. Ben-Zvi
R. E. Brown
G. Chan
M. S. Day
K. J. Goldman
D. J. Hwang
E. K. Kolodner
R. Ladin
G. T. Leavens
S. C. Markowitz
T. D. Nguyen
B. M. Oki
S. E. Perl
M. Vermeulen
A. S. Xu

Undergraduate Students

L. Bela
A. K. Farrell
B. J. Sagarin
P. A. Viola
K. Vogel
W. Wong

Support Staff

E. Pothier
A. Rubin

Visitors

B. DeDecker
L. Shrira
S. Sugihara

1. INTRODUCTION

The research in the Programming Methodology Group has continued to focus on the area of distributed computing. We have continued our work on the Argus programming language and system. We have been designing an extension to Argus to enable it to make use of Mercury mechanisms, especially the call-stream. Also, we have continued our work on a broad range of problems that arise in distributed systems. Our research in these areas is described in the following sections.

In addition, the CLU implementation on M68000 based machines has been ported to the HP machines that run HP-UX. CLU also runs now under Berkeley Unix 4.3 and under Ultrix.

2. ARGUS

The following progress has been made on Argus.

1. A revised Argus reference manual has come out [13]. The revisions were done primarily by Mark Day and Gary Leavens.

2. Work on the implementation has continued. The implementation has been ported to Berkeley Unix 4.3. Much of the effort has centered on the debugging system, which is being improved along the lines suggested by Mike Vermuelen's thesis. This thesis proposed ways for making debugging of distributed programs more like those of sequential programs with easy checkpoints on remote procedure call and return.

3. In addition we have been doing performance analysis of certain critical areas such as communication. Eventually we intend to reimplement certain areas including making some changes to the Unix Kernel.

4. A paper describing the implementation and its performance has been written and accepted for SOSP 1987 [14].

5. Elliot Kolodner has completed his thesis on a new way of supporting crash recovery [7]. His approach speeds up recovery from soft crashes in which disk storage is unaffected. Recovery uses information in both virtual memory and the stable log. Checkpoints are taken frequently in which dirty pages are written to disk; checkpoints reduce the amount of the log that must be scanned after a crash. The thesis proposes modifications to a copying garbage collection algorithm such that recovery using virtual memory is possible even if the crash occurred during garbage collection.

6. A new overview paper about Argus and its support for applications has been written for a forthcoming special issue of the Communications of the ACM [15].

7. A paper about experience using Argus to implement a distributed editor has been written [2]. Our experience in this application indicates that Argus performance is

satisfactory, but that there is a problem with expressive power in some areas. We have been studying alternative techniques that may alleviate some of these problems, in particular, in a thesis being done by Sharon Perl.

8. Mark Day has completed a thesis on a mail repository implemented in Argus [1]. The repository was designed to replace the backend of the Athena mail system in which a single server node stores the mail. The Athena approach is a performance bottleneck and has an availability problem, since a crash of the server node makes mail unavailable. The approach in the thesis replicates the repository. It allows sending and receiving mail to work in as large a number of situations as possible without exposing the user to confusing inconsistencies. It also investigates techniques for finding mailboxes and for allowing on-the-fly system reconfiguration in which individual mailboxes may be moved from one server (or group of servers) to another.

9. Gary Leavens has implemented an application in Argus to compute Hailstone numbers [4], [9]. Our experience here is that Argus made this application straightforward to implement with essentially no performance penalty for the use of atomic actions.

3. MERCURY

In conjunction with the work on Mercury, we have been studying how to extend Argus to incorporate the new Mercury communication mechanisms. Issues under study include the following:

1. We have been studying how to integrate a more general transaction model into Argus. At present Argus does not allow a parent action to run concurrently with its descendants. In Mercury such concurrency is supported because of remote stream calls in which the caller continues to run in parallel with the processing of the call at the remote site. The additional concurrency has caused us to reformulate our locking rules. In addition, we have investigated an approach that combines timestamps with locking and that would allow us to guarantee that if two streams calls are made in order s1; s2 to two different remote sites, then they are serialized in that order. We have rejected this approach, however, because it has several unfortunate properties. For example, a change in system performance can cause a program that previously ran properly to deadlock.

2. Another issue opened up by stream calls is how to integrate them into Argus in a way that is both general and type-safe. We are investigating an approach based on futures [3]. The idea is to treat a stream call as a special kind of strongly typed "fork"; we call this entity a "promise". For example, consider

p: **handler** (array[int]) **returns** (real) **signals** (cannot).

(A handler is a remote procedure in Argus.) Then the following is a legal program:

```
pt = promise (real) signals (cannot)
a: array[int] := ...
x:  pt := stream p(a)
...
y: real := pt$claim(x) + pi
      except when cannot: ... end
```

After the stream call, the calling program continues to run. As is the case with futures, a promise like x can be assigned to another variable (also of type promise) or passed as an argument or result without being "really" used; real use only happens with the value of the promise is needed. When the program uses the value, as in the expression in which the value of x is added to pi, it extracts the value by using the "claim" operation, which causes it to wait for the call to complete if necessary. The work on promises has been carried out primarily by Liuba Shrira.

3. Another problem is how to match up Argus types with the language-independent types that define how values are communicated in Mercury. For example, the types shown above are Argus types. The call might actually be directed to

PORT (SEQ[INT32]) RETURNS (FLOAT) SIGNALS (CANNOT).

Here capitals are used to indicated Mercury types. In Mercury a remote procedure is called a port. For the call from Argus to work, we need a way of relating array[int] to SEQ[INT32] and real to FLOAT, and also a way of copying the array into the call message and the real from the return message. We are investigating a method that allows types and Mercury types to be associated with one another very flexibly but that is nevertheless type-safe.

4. ISSUES IN DISTRIBUTED SYSTEMS

In addition we have continued our work on issues in distributed systems. Some achievements in this area are discussed below.

1. We have written a paper describing our orphan detection algorithm [11], [12]. An orphan is a computation that continues to run when its result is no longer of interest. Such a situation might arise, for example, because the calling node crashed while that call was being processed; the call processing is the orphan in this case. Our algorithm works by sending information on all messages in the system. It finds orphans quickly but is inefficient because of the amount of information propagated and the time it takes to process the information. Two optimizations have been proposed: using a central service to store the information, and eliminating information when deadlines have expired. We are implementing both optimizations at present.

2. Bill Weihl, working jointly with Nancy Lynch and others, has succeeded in proving the correctness of part of our orphan detection method [5], [6].

3. A paper describing our central service method has been written and submitted for publication [8]. An early version of this paper appeared in PODC 1986 [10]. The method provides a highly available service that is tolerate of common failures such as node crashes and partitions and lost messages. The service is implemented by several replicas. Clients need communicate with only one replica; replicas propagate new information to one another in background mode by exchanging "gossip" messages. Clients can request answers from the service based on old information, but can bound how old the requested information must be. The method outperforms other techniques such as voting, but is limited to applications that can use old information effectively.

4. Rivka Ladin, as part of her thesis, has developed a technique for doing garbage collection in a distributed system. The technique is tolerant of failures such as node crashes and partitions and loss of messages. It uses the central service to detect when objects are no longer referred to by clients. Clients run local garbage collection using an algorithm of their choice (a different algorithm can be used at different clients) and at a convenient time; no synchronization between clients is required. They communicate information about nonlocal references to the service, and later inquire about the status of local objects that may be referred to by other clients. The method requires relatively few messages to be exchanged compared with other methods. A preliminary report on the method appeared in PODC 1986 [10].

References

1. Day, M. "Replication and Reconfiguration in a Distributed Mail Repository." MIT/LCS/TR-376, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

2. Greif, I., R. Seliger and W. Weihl. "A Case Study of CES: A Distributed Collaborative Editing System Implemented in Argus." Programming Methodology Group Memo 55, MIT Laboratory for Computer Science, Cambridge, MA, April 1987. Also to be published in *IEEE Transactions on Software Engineering.*

3. Halstead, R. "Multilisp: A Language for Concurrent Symbolic Computation." *ACM Transactions on Programming Languages and Systems*, 7, 4, (October 1985).

4. Hayes, B. "Computer Recreations: On the Ups and Downs of Hailstone Numbers." *Scientific American*, 250, 1, (January 1984), 10-16.

5. Herlihy, M. P., N. Lynch, M. Merritt and W. Weihl. "On the Correctness of Orphan Elimination Algorithms (Preliminary Report)." To be published in *Proceedings of the 17th International Symposium on Fault-Tolerant Computing.*

6. Herlihy, M. P., N. Lynch, M. Merritt and W. Weihl. "On the Correctness of Orphan Elimination Algorithms." MIT/LCS/TM-329, MIT Laboratory for Computer Science, Cambridge, MA, May 1987.

7. Kolodner, E. "Recovery Using Virtual Memory." MIT/LCS/TR-404, MIT Laboratory for Computer Science, Cambridge, MA, July 1987.

8. Ladin, R., B. Liskov and L. Shrira. "A Technique for Constructing Highly-Available Services." To be published in *Algorithmica.*

9. Lagarias, J. C. "The 3x+1 Problem and Its Generalizations." *The American Mathematical Monthly*, 92, 1, (January 1985), 3-23.

10. Liskov, B. and R. Ladin. "Highly-Available Distributed Services and Fault-Tolerant Distributed Garbage Collection." *Proceedings of the 5th ACM Symposium on Principles of Distributed Computing*, Calgary, Alberta, Canada, August 1986. Also Programming Methodology Group Memo 48, MIT Laboratory for Computer Science, Cambridge, MA, 1986.

11. Liskov, B., R. Scheifler, E. Walker and W. Weihl. "Orphan Detection."

Programming Methodology Group Memo 53, MIT Laboratory for Computer Science, Cambridge, MA, February 1987.

12. Liskov, B., R. Scheifler, E. Walker and W. Weihl. "Orphan Detection (Extended Abstract)." *Proceedings of the 17th International Symposium on Fault-Tolerant Computing*, Pittsburgh, PA, July 1987, 2-7.

13. Liskov, B., et al. "Argus Reference Manual." MIT/LCS/TR-400, MIT Laboratory for Computer Science, Cambridge, MA, November 1987.

14. Liskov, B., D. Curtis, P. Johnson and R. Scheifler. "Implementation of Argus." *Proceedings of the ACM 11th Symposium on Operating Systems Principles*, Austin, TX, November 1987.

15. Liskov, B. "Distributed Programming in Argus." *Communications of the ACM*, 31, 3, (March 1988), 300-312.

Publications

1. Coan, B., B. Oki and E. Kolodner. "Limitations on Database Availability when Networks Partition." *Proceedings of the 5th ACM Symposium on the Principles of Distributed Computing*, Calgary, Alberta, Canada, August 1986.

2. Day, M. "Replication and Reconfiguration in a Distributed Mail Repository." MIT/LCS/TR-376, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

3. Dolev, D., N. Lynch, S. Pinter, E. Stark and W. Weihl. "Reaching Approximate Agreement in the Presence of Faults." *Journal of the ACM*, 33, 3, (July 1986).

4. Fekete, A., N. Lynch, M. Merritt and W. Weihl. "Nested Transactions and Read-Write Locking." *Proceedings of the Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, March 1987.

5. Greif, I., R. Seliger and W. Weihl. "A Case Study of CES: A Distributed Collaborative Editing System Implemented in Argus." Programming Methodology Group Memo 55, MIT Laboratory for Computer Science, Cambridge, MA, April 1987. Also to appear in *IEEE Transactions on Software Engineering*.

6. Heddaya, A., M. Hsu and W. Weihl. "Background Checkpointing and Discarding Old Events in Event-based Distributed Systems." January 1987, submitted for publication.

7. Herlihy, M., N. Lynch, M. Merritt and W. Weihl. "On the Correctness of Orphan Elimination Algorithms." MIT/LCS/TM-329, MIT Laboratory for Computer Science, Cambridge, MA, May 1987.

8. Herlihy, M., N. Lynch, M. Merritt and W. Weihl. "On the Correctness of Orphan Elimination Algorithms (Preliminary Report)." To be published in *Proceedings of the 17th International Symposium on Fault-Tolerant Computing.*

9. Ladin, R., B. Liskov and L. Shrira. "A Technique for Constructing Highly-Available Services." To appear in a special issue of *Algorithmica.*

10. Liskov, B. "Programming Methodology Group Progress Report July 1, 1984 - June 30, 1985." Programming Methodology Group Memo 51, MIT Laboratory for Computer Science, Cambridge, MA, February 1987.

11. Liskov, B. "Highly-Available Distributed Services." Programming Methodology Group Memo 52, MIT Laboratory for Computer Science, Cambridge, MA, February 1987.

12. Liskov, B. "Distributed Programming in Argus." To be published in a special issue of *Communications of the ACM.*

13. Liskov, B., et al. "Argus Reference Manual." Programming Methodology Group Memo 54, MIT Laboratory for Computer Science, Cambridge MA, March 1987.

14. Liskov, B. and M. Herlihy. *Issues in Process and Communication Structure for Distributed Programs.* Computer Systems for Process Control. R. Guth (ed.), Plenum Press, 1986.

15. Liskov, B., P. Johnson and R. Scheifler. "Implementation of Argus." To be published in the *Proceedings of the Eleventh ACM Symposium on Operating Systems Principles.*

16. Liskov, B. and R. Ladin. "Highly-Available Distributed Services and Fault-Tolerant Distributed Garbage Collection." *Proceedings of the 5th ACM Symposium on the Principles of Distributed Computing.* Calgary, Alberta, Canada, August 1986.

17. Liskov, B., R. Scheifler, E. Walker and W. Weihl. "Orphan Detection." Programming Methodology Group Memo 53, MIT Laboratory for Computer Science, Cambridge, MA, February 1987.

18. Liskov, B., R. Scheifler, E. Walker and W. Weihl. "Orphan Detection (Extended Abstract)." To be published in the *Proceedings of the 17th International Symposium on Fault-Tolerant Computing.*

19. WeihlOC, W. "Distributed Version Management for Read-Only Actions." *IEEE Transactions on Software Engineering*, (January 1987).

20. Weihl, W. "Local Atomicity Properties: Modular Concurrency Control for Abstract Data Types." Accepted for publication in *ACM Transactions on Programming Languages and Systems.*

Theses Completed

1. Bela, L. "Winsrv Support for Common ASCII Terminals." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

2. Chan, G. "Generalized Transaction Management." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge MA, May 1987.

3. Day, M. "Replication and Reconfiguration in a Distributed Mail Repository." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, February 1987.

4. Kolodner, E. "Recovery Using Virtual Memory." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

5. Vermeulen, M. "Debugging with Remote Procedure Calls." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, August 1986.

6. Vogel, K. "Highly Available Stable Storage for Argus." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

Theses in Progress

1. Farrell, A. "A Deadlock Detection Program for a Distributed Computing System." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected February 1988.

2. Hwang, D. "Constructing Highly-Available Services in a Distributed Environment." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected February 1988.

3. Leavens, G. "A Semantics for Subtypes." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected June 1988.

4. Nguyen, T. "Measurement of Orphan Detection Protocols." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected December 1987.

5. Oki, B. "Building Highly Available Distributed Programs." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected January 1988.

6. Perl, S. "Commit Protocols for Nested Atomic Actions." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1987.

7. Sagarin, B. J. "Theory and Implementation of Concurrent I/O in Argus with the X-Window System." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected February 1988.

8. Wong, W. "Investigation of Methods to Maintain Availability in Partitioned Replicated Systems." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected February 1988.

Talks

1. Day, M. "Building an Application in Argus." Second European SIGOPS Workshop on Distributed Systems, Amsterdam, The Netherlands, September 1986.

2. Ladin, R. "Highly-Available Distributed Services and Fault-Tolerant Distributed Garbage Collection." 5th ACM Symposium on the Principles of Distributed Computing, Calgary, Alberta, Canada, August 1986.

3. Liskov, B. "Argus and Support for Distributed Systems." NEC Corporation, Tokyo, Japan, October 1986.

4. Liskov, B. "Support for Heterogeneous Computing." NEC Corporation, Tokyo, Japan, October 1986

5. Liskov, B. "LCS Common System." Presentation to DARPA, MIT Laboratory for Computer Science, Cambridge, MA, December 1986.

6. Liskov, B. "Communication in a Heterogeneous System."

Texas Instruments, Dallas, TX, February 1987;
MIT Laboratory for Computer Science, March 1987;
Stanford University, Stanford, CA, March 1987;
Georgia Institute of Technology, Atlanta, GA, April 1987;
Dept. of Information and Computer Science, University of California, Irvine, May 1987.

7. Liskov, B. "Support for Programming in Heterogeneous Systems."
NYNEX, White Plains, N. J., June 1987;
Annual Meeting, Laboratory for Computer Science, June 1987.

Liskov, B. "Argus: The Programming Language and System." Bell Communications Research, Morristown, NJ, April 1987.

8. Oki, B. "Limitations on Database Availability when Networks Partition." 5th ACM Symposium on the Principles of Distributed Computing, Calgary, Alberta, Canada, August 1986.

9. Weihl, W. "Communication in the MIT Common System."
Brown University, Providence, RI, March 1987;
DEC Systems Research Center, Palo Alto, CA, March 1987;
IBM Almaden Research Center, San Jose, CA, March 1987;
University of Washington, Seattle, WA, March 1987.

PROGRAMMING SYSTEMS RESEARCH

Academic Staff

D. Gifford, Group Leader

Research Staff

S. Berlin

Graduate Students

R. Baldwin
M. Blair
A. Braunstein
R. Brown
M. Day
N. Glasser
T. Hamell
D. Heitmann
J. Lucassen
J. O'Toole
R. Rao
M. Sheldon
S. Slivan
J. Stamos

M. Strauss

Undergraduate Students

H. P. Brondmo
R. Cote
R. Gruber
J. Henderson
J. LaRocca
P. Ritto

D. Segal

Support Staff

R. Bisbee

Visitors

D. Burmaster
P. Jouvelot

1. INTRODUCTION

The Programming Systems Research Group has been involved in two major efforts over the past year. The first effort was the addition of new functions and algorithms to a distributed database system that we have designed and now operate at over 150 sites. The second effort was the development of a new programming model for parallel computation that introduces the notion of an *effect system*.

As reported last year, we have developed a new type of large scale information system that is in use at over 150 sites in the Boston area. The goal of this information system research is to explore how a new type of system architecture can be used to implement information systems for that can support very large user populations - perhaps up to one million users. Our prototype system, the Boston Community Information System, was improved during the last year to add database content labeling and query routing. Query routing allows the personal database system software to automatically connect to database servers at MIT when necessary.

As part of our work on large scale information systems, we have developed and implemented the Remote Channel System. The Remote Channel system is a remote procedure call system with procedural parameters and pipes. Pipes are procedure-like objects that, when called remotely, do not cause the caller to block waiting for completion, allowing the local and remote programs to execute concurrently. During the past year, the Remote Channel System was specified and implemented on Unix. The implementation is used by the Walter distributed database program, a component of the Boston Community Information System.

In order to evaluate our information system work we have conducted an in-depth survey of present database system users via monthly questionnaires. We are presently analyzing this information, and in the near future we hope to be able to draw conclusions about the promise of the information system technology we have developed.

Along a different line of research, we have been engaged in a project to develop a programming model for parallel computation that combines the good features of both imperative and functional programming languages. As a result of this work we have developed the notion of an effect system. An effect system is analogous to a type system (as found in many programming languages), but whereas types describe *what* results from a computation, effects classify *how* the computation proceeds.

We have completed the design and implementation of the prototype language called FX that includes a type and an effect system. The design features a type and effect system with type, effect and region polymorphism, parameterized and recursive types, and bounded quantification. FX was implemented in Zetalisp on the Symbolics 3600. The implementation consists of three parts:

- a front-end that performs syntax analysis and type checking;

- a sequential interpreter that permits rapid turnaround; and

- a parallel dataflow compiler that generates dataflow graphs suitable for execution on the tagged-token dataflow simulator constructed by the Computation Structures Research Group.

The Computation Structures Research Group dataflow simulator was also extended to accommodate a small number of new low-level memory and synchronization operations that were needed to support the execution of the prototype language.

Experience with the prototype language has lead to the redefinition of **FX**. A draft reference manual for the revised version of FX is now complete. A type-checker and type-erasing interpreter has been built on Scheme; this implementation allows the execution of FX programs but does not take advantage of type and effect information to schedule parallel evaluation of expressions.

In theoretical work, we have demonstrated that FX can be used to implement the f_e0 function on ordinals and thus has the full power of second-order typed lambda-calculus. Other well-known programming languages (such as ML, Russell, and Pebble) do not have this power. We have also found that type-checking FX in the presence of recursively defined type functions is equivalent to an open problem concerning the equivalence of pushdown automata. However, in the restricted cases currently planned for use in FX, the type-checking problem with recursive type functions is no harder than without them.

2. PLANS FOR THE NEXT YEAR

Our plans for the next year are as follows:

- Develop a Project Mercury interface to our database system in order to test the usefulness of the Mercury communication model, and to provide a useful service to users both within MIT and at other DARPA sites.

- Develop an X Window-based user interface to our database system to permit us to experiment with a new user interface, and to allow us to introduce new features, such as direct client modification of server databases. This X window interface will also provide a useful service.

- Complete the implementation of FX and to develop application programs to test the usefulness of the programming model that we have developed.

- Investigate a new type of computer architecture for distributed multiprocessing, and to design a new version of FX that will take advantage of this architecture.

3. REMOTE PIPES AND PROCEDURES FOR EFFICIENT DISTRIBUTED COMMUNICATION

As part of our work on large scale information systems we have developed a new communication model for distributed systems called the *Remote Pipe and Procedure Model* that combines the advantages of bulk data transport and remote procedure call in a single simple framework.

Remote procedure call is now a widely-accepted standard method for communication in distributed computer systems [18][6] [14][12][2]. This popularity can be attributed to three advantages provided by remote procedure call. First, remote procedure call uses a widely-accepted, used, and understood abstraction, the procedure call, as the sole mechanism for access to remote services. Second, remote procedure call allows remote interfaces to be specified as a set of named operations with certain type signatures. Such specifications enable remote interfaces to be precisely documented, and distributed programs to be statically checked for type errors. Third, since interfaces can be precisely specified, the communication code for an application can be automatically generated, by either a compiler or a specialized stub generator.

The wider use of remote procedure call systems has led to an understanding of their disadvantages as well as their advantages. Based on our recent application experience [7], we have have discovered three major problem areas in standard remote procedure call systems: protocol flexibility, incremental results, and bulk data transport.

1) *Protocol Flexibility*: Certain communication protocols are impossible to implement if a remote procedure call system does not allow remote procedure values to be exchanged freely between nodes. For example, imagine that a client node wishes to provide a server node with a procedure for use in certain circumstances, and the server node then wishes to pass this procedure on to another server. Unless remote procedures are first-class objects that can be passed from node to node this protocol can not be expressed in a remote procedure call framework.

2) *Incremental Results*: Consider a server that is computing a result on behalf of a client and wishes to communicate incremental results to the client as they are computed. In present remote procedure call systems this would be accomplished by having the client ask the server to compute the first incremental result, then the second, and so forth until all of the results have been computed. The problem with this approach is that it forces a single computation to be decomposed into a series of distinct remote procedure calls. This decomposition reduces the performance of the server since it is inactive between client procedure calls unless it creates a sophisticated process structure upon the client's first incremental result request. Sophisticated process structures are undesirable because they substantially complicate a program.

3) *Bulk Data Transport*: Remote procedure call mechanisms are optimized for both low call-return latency and the transmission of limited amounts of data (usually less than 10^3 bytes). These optimizations for the normal case seriously affect the ability of remote procedure call mechanisms to transport large amounts of data efficiently. Since in contemporary systems only one remote procedure call can be in transit at a given time between a single process client and a server, the communication bandwidth between them is limited. For example, if we assume that a program transmits 10^3 bytes per remote procedure call and the network has a 50 millisecond round trip latency, the maximum bandwidth that can be achieved is 20 KBytes/second. Furthermore, to achieve even this performance, the client must combine data values as they are produced into 10^3 byte blocks before a remote procedure call is made. If a remote procedure call was made whenever data was available to be sent, e.g. for each character to be displayed on a remote screen, communication performance could drop to 20 bytes/second.

As a direct result of our experience with these limitations we have developed a new communication model called the *Remote Pipe and Procedure Model* that extends remote procedure call in three directions and address the three disadvantages discussed above. First, we permit remote procedures to be first-class values which can be freely passed between nodes. Second, we introduce a new abstraction called a *pipe* that allows bulk data and incremental results to be efficiently transported. Third, we introduce *channel groups* which control the relative sequencing of calls on pipes and procedures.

Elements of the Remote Pipe and Procedure Model have been present in previous work although these elements have not been combined into a single consistent framework. The idea of transmitting remote procedure values is discussed by Nelson [14], and is also present in Argus [12] as handlers. However neither of these proposals allow remote procedures to be created in nested scopes which limits the generality of remote procedures. The notion of a pipe is similar in some respects to Nelson's immediate return procedures [14], and the unidirectional messages of Matchmaker [11]. Nelson however rejected immediate return procedures for his communication model because they were inconsistent with procedure call semantics. Our solution to the consistency problem is the creation of a new type of abstract object with well defined properties. In Matchmaker remote procedures are not first-class values and unidirectional message sends are not optimized for bulk data transmission. None of the above systems include an idea similar to a channel group.

The remainder of this paper is organized into three sections: Semantics (Section 2), Refinements (Section 3), Pragmatics (Section 4), and Practical Experience and Conclusions (Section 5).

4. SEMANTICS

We will discuss in this section:

- typed remote interfaces,
- the use and creation of remote pipes and procedures,
- channel call ordering,
- how channel groups provide inter-channel synchronization,
- and failure semantics.

For the purposes of our discussion we will define a *node* to be a virtual computer with a private address space. A physical computer can implement one or more nodes; the precise size and scope of a node will depend on application requirements. We will assume that all of the nodes in a system are interconnected by a network.

Both remote procedures and pipes provide a communication path to a remote node, and thus we call them *channels*. A channel represents the ability to perform a specific remote operation at a remote node. Channels are first-class values. In particular, channels can be freely passed to remote procedures or pipes as parameters, or returned as the result of a remote procedure. Connections are implicitly established as necessary when channels are used as described in Section 4.

A node that exports a set of channels will be called a *service* node, while a node that imports a set of channels will be called a *client* node. A given node can be a service and a client at the same time with respect to different sets of channels.

It is not always the case that a client calls a service; when a service and client are working together on a task it is sometimes most natural for a service to call a client. For example, a client might provide a service with a pipe in order to allow the service to send a large file to the client. Thus we will introduce terminology to describe the dynamic relationship between two nodes. We will call the node that processes calls made on a channel the channel's *sink* node, and we will call the node that makes calls on a channel the *source* node. Note that a given node can be both a source and a sink.

Remote procedures and pipes are defined as follows:

- *Procedures*: A *remote procedure value* provides the capability to call a procedure on a remote node. A *remote procedure call* blocks the caller until the remote procedure has finished execution and a response has been received from the sink node. Because a remote procedure call blocks a client, remote procedure calls are optimized for *minimum latency*. In the event that a remote procedure call fails, a distinguished error value is

returned as the value of the call. The precise types of failures that can result are described below.

- *Pipes*: A *pipe value* provides the capability to send bulk data to the pipe's sink node. A pipe is used as is a remote procedure. However, unlike a remote procedure call, a *pipe call* does not block the caller and does not return a result. Because a pipe is designed for bulk data transport, it is optimized for *maximum throughput*. Since a pipe call does not block its caller a pipe call implicitly initiates concurrent activity at the pipe's sink node. The caller continues execution as soon as the call to the sink is queued for transmission. The data values sent down a pipe by a given process are received by the pipe's sink node in the order they are made. Received means that the sink receives the data in order and performs some computation on the data. This computation could process the data to completion or simply schedule the data for later processing. The failure of a pipe call to complete can be detected as described below.

The Remote Pipe and Procedure Model is designed to be readily adaptable to a wide range of programming languages. For pedagogical purposes we will write our examples in Modula-II. Section 5 contains an discusses a C implementation of remote pipes and procedures.

4.1. Remote Interfaces are Typed

A *remote interface* is a set of type definitions that is used to describe a collection of channels that is exported as a unit by a service. We will use a collection of Modula-II type definitions to describe a remote interface, with the language extended to include types for remote procedures and pipes.

We introduce the types PIPE and PROC in our interface language to describe pipes and remote procedures, respectively. A channel's type further describes the channel's input values and the channel's result values. Note that only procedures have result values. For example a pipe value might have type

```
PIPE PutChar(CHAR);
```

indicating that the pipe is a character pipe, while a remote procedure might have type

```
PROCEDURE GetChar(): CHAR;
```

indicating that the remote procedure does not take an argument and it returns a character.

The following is an example of a remote interface that defines three channels. Note that two of the channels are pipes, while one is a remote procedure.

```
REMOTE INTERFACE Terminal;
  TYPE Color = (red, blue, green);
  PIPE PutChar(c: CHAR);
    (* Displays a character on the terminal *)
  PIPE SetColor(c: Color);
    (* Sets the display color for subsequent characters *)
  PROCEDURE GetChar(): CHAR;
    (* Returns the next character. Blocks if a character is
       not available *)
END Terminal.
```

Interfaces are used to document a service, as well as to allow an implementation of the Remote Pipe and Procedure Model to encode and decode messages of appropriate types. For example, a remote interface, by virtue of its type declarations, contains enough information to permit a stub generator [14] to automatically generate code to implement the details of a communications protocol. Once an interface is specified an application programmer can deal with pipes and procedures, and not be concerned with how information is encoded and transmitted over a wire.

Figure 9-1 shows how the interface above might be used to define a communications protocol. The interface is first transformed by an automatic stub generator into two stubs, a *server* stub and a *client* stub. As shown in Figure 9-1, the server stub includes the operation `Export`. `Export` is called by the server to register an instance of the channels described by `Terminal` for use by client programs. In order to use these operations a client program calls the `Import` operation in the client stub to obtain a specific instance of `Terminal`.

The general problem of providing clients with instances of service interfaces is known as *binding*, and we will assume that an implementation of the Remote Pipe and Procedure Model will employ a conventional binding technique. For example, the `Import` and `Export` scheme shown in Figure 9-1 is employed by the Cedar Remote Procedure Call System [2]. Another technique is to use a distributed database system to record offered services. For example, the Courier remote procedure call system [20] in the Xerox Star System uses the Clearinghouse distributed database system [19] for client-server binding.

Binding allows a node to bootstrap itself into a distributed environment. Note that binding is used to establish initial contact with a set of servers, and that this set of initial servers can easily provide a client with a rich set of channels that can be used to contact other nodes that were not provided via the binding mechanism.

4.2. Channels are Represented by Local Procedures

Remote procedures and pipes are used in the same manner as are local procedure values. In order to allow procedures and pipes to be introduced without changing Modula-II, we will represent channels with local procedures. Thus, a local procedure must be converted to a channel value for export, and a channel value must be converted to a local procedure on import. If a given channel is first imported as a local procedure, and then that local procedure is exported as a channel, the channel value exported must be the same as the original channel value imported to ensure that extra overhead is not introduced.

The channel type declarations that are used in programs are automatically produced by the stub generation program from the user's original remote interface definition. For example, the `Terminal` remote interface would result in the following client stub definitions module:

```
DEFINITION MODULE TerminalClient;
 TYPE Color = (red, blue, green);
 Terminal = RECORD
          PutChar: PROCEDURE (CHAR),
          SetColor: PROCEDURE (Color),
          GetChar: PROCEDURE (): CHAR
        END;
 PROCEDURE Import(server: ARRAY OF CHAR): Terminal;
 END TerminalClient.
```

Thus, assuming that `term` is an instance of terminal, in order to call the procedure `GetChar`, the statement

```
new-char := term.GetChar();
```

could be used.

A pipe value is used in precisely the same way as a procedure value is used except that pipes do not return result values. The following expressions send the values "o" and "k" down `PutChar`:

```
term.PutChar("o");
term.PutChar("k");
```

The values "o" and "k" are guaranteed to be received by the sink of `PutChar` in order (because the two pipe calls shown above are performed by the same process). No reception order is defined for pipe calls that are made by separate processes.

Since pipe calls do not return values and are processed asynchronously, a

Synchronize operation is provided. When a **Synchronize** operation is applied by a source process to a pipe, the pipe's sink is forced to process all outstanding data sent down the channel from the source process, after which the synchronize operation returns. If the pipe has broken for some reason (e.g. the sink node has crashed) then synchronize will return a distinguished error value and reset the pipe so that it can be used again. Errors are described in detail in Section 2.5.

```
term.PutChar("n")
term.PutChar("o")
code := Synchronize(term.PutChar);
```

A pipe value is created through the provision of a local procedure called a *pipe sink procedure*, that will process data received over the pipe. As data arrives through a pipe its corresponding sink procedure is applied to each datum in the order it is received. A pipe's sink procedure must return before it will be applied to the next datum sent down the pipe from the same source process. When a node desires to export a pipe it provides a pipe sink procedure, and this sink procedure is converted into a pipe value by the sink's stub.

We now have enough mechanism to introduce a simple file transfer protocol based on channels. Recall that channels can take channels as arguments and return channels as values. Thus, simplified file transfer interface could be represented as the two procedures **SendFile** and **GetFile**:

```
REMOTE INTERFACE Ftp;
  PROCEDURE SendFile(name: ARRAY OF CHAR): PIPE (CHAR);
    (* Returns a pipe to receive the contents of name *)
  PROCEDURE GetFile(name: ARRAY OF CHAR, put-here: PIPE(CHAR));
    (* The contents of name are sent down the pipe put-here *)
END Ftp.
```

SendFile returns a pipe to receive the contents of the file called **name**. The file is then transmitted down the returned pipe, and with the end of a file marked by a special character. The symmetric operation **GetFile** is passed a file to retrieve and a pipe called **put-here** for the retrieved file to be sent down.

4.3. Calls on a Channel By a Single Process Are Ordered

Our communication model guarantees that if a process makes two separate calls on the same channel then the calls will be processed at the sink in the order in which they were made by the process. Processed means that the second call is not executed at the sink until the procedure invoked by the first call has returned.

The ordering of channel calls not covered by the above invariant is undefined. Thus,

a single channel can be invoked simultaneously by different source processes. We assume that monitors [16] or a similar mechanism is used to ensure the proper operation of remote procedures and pipes in the presence of concurrent invocations.

4.4. Channel Groups Provide Inter-Channel Timing Invariants

In our present model the ordering of calls on separate channels is undefined. However at times it is desirable to provide a timing invariant across channels. Consider our example **Terminal** that was defined above, and the characters that would be displayed by the following statements:

```
term.PutChar("a");
term.SetColor(blue);
term.PutChar("b");
```

Because **SetColor** and **PutChar** are separate pipes, there is no sequencing invariant defined between them. Thus the character **b** may or may not be displayed in blue, depending on the order in which the pipe calls **SetColor** and **PutChar** are processed by their sink node. In this example we would like to be able to specify that calls on **SetColor** and **PutCharacter** must be performed in the order in which they were made.

When timing invariants must be preserved between a set of channels the channels can be collected into a *channel group*. A channel group is a collection of pipes and procedures that all have the same sink node and that observe a sequential ordering constraint with respect to a source process. This ordering constraint guarantees that calls made by a single process on the members of a channel group are processed in the order they were made.

In order to construct channel group values we introduce **Group**. **Group** takes a record of channel values as input and returns a new record of channels that are in the same group. Group relationships that existed in the input channels will be present in the output channels. If all of the channels that are passed to **Group** do not have the same sink node then **Group** will return a distinguished error value. Note that the channels returned by **Group** are copies, and thus the original input record to **Group** will still refer to a set of ungrouped channels after **Group** returns. The channel copies returned by **Group** will each include a new unique *sequence stamp* that has been added by **Group**. The sequence stamp added to each channel is used to identify each channel's membership in the newly created group. In addition to the sequence stamps obtained by group membership, upon creation each channel is assigned a unique sequence stamp.

We can now solve our earlier problem of synchronizing **SetColor** and **PutChar** calls. For example, the statements

```
seqTerm := Group(term);
seqTerm.SetColor(red);
seqTerm.PutChar("a");
seqTerm.PutChar("b");
```

can be used to create a new sequenced terminal value **seqTerm** and to display the characters "ab" in red.

A channel can be a member of more than one group at once, which allows a wide variety of sequencing semantics to be directly expressed using channel groups. For example, consider the following remote interface:

```
REMOTE INTERFACE ColorDisplay;
  TYPE Color = (red, blue, green);
  PIPE SetFont(font: ARRAY OF CHAR);
    (* Sets the font *)
  PIPE SetColor(c: Color);
    (* Sets the color *)
  PIPE PutChar(c: CHAR);
    (* Displays a character on the screen *)
END ColorDisplay.
```

In order to guarantee that characters appear in the proper color and font a server that exports **ColorDisplay** can create two channel groups: {**SetFont**, **PutChar**} and {**SetColor**, **PutChar**}. Note that **SetFont** and **SetColor** do not need to be sequenced with respect to one another. Utilizing two channel groups instead of a single channel group to sequence **ColorDisplay** allows **SetFont** and **SetColor** to execute in parallel.

The channel timing invariant provided by the communication model can now be succinctly stated:

Channel Timing Invariant If a process makes two separate calls on channels that (1) are at the same sink node, and (2) have a sequence stamp in common, then the calls will be processed at the sink in the order in which they were made by the process. Processed means that the second call is not executed until the procedure invoked by the first call has returned.

The channel timing invariant implies the invariant for calls on a single channel (because a channel will always be at the same sink node as itself and will have a sequence stamp in common with itself). The channel timing invariant embodies all of the ordering semantics provided by the Remote Pipe and Procedure Model. The ordering of channel calls not covered explicitly by the channel timing invariant is undefined.

4.5. Failures Complicate Channel Semantics

Our communication model guarantees that a channel call will be performed precisely once in the absence of failures. In the presence of failures the semantics of remote operations are more complicated. Many kinds of distributed system failures (e.g. node crashes, network partitions) can cause a source node to wait for a reply which will never arrive. In such cases it is impossible to tell if the corresponding remote operation was ever attempted or completed.

In the presence of failures *at-most-once* semantics can be provided for remote calls. At-most-once semantics guarantees that a remote operation either will be performed exactly once, or will have been performed at most once if a failure has occurred. A failed procedure call returns a distinguished *crash* value as its result. A failed pipe call causes a distinguished *crash* value to be returned as the result of the next procedure call on the same group. When a failure occurs it is impossible to determine whether a remote operation was completed, never started, or only partially completed. Thus *at-most-once* semantics present a serious challenge to the application programmer who wishes to cope gracefully with failure.

One technique used in several practical systems accepts the limitations of at-most-once semantics and insists that procedure calls that mutate stable storage be idempotent. With this restriction a remote procedure call that returns *crash* can be repeated safely until the call completes without failing.

Exactly-once semantics is an alternative to at-most-once semantics. Exactly-once semantics guarantees that a remote operation will be performed exactly once or not at all. Exactly-once semantics is implemented by protecting the actions of a remote operation with a transaction. If a remote operation returns *crash* the operation's corresponding transaction is aborted. The transaction abort will undo the effects of the failed remote operation and the failed operation will appear to never have happened. The failed operation can then be retried (if desired) with a new transaction.

Exactly-once semantics can be achieved through the combination of communication with transactions in one of two ways. One approach as suggested by Argus's innovative design [12] is to integrate transactions into the communication model such that each remote operation has an implicit associated transaction. A second approach is to keep the communication model and transactions separate by explicitly specifying transactions where they are required [4].

In addition to success and crash, a third result can be optionally returned from a remote call. If desired, a ping message can proceed a call to ensure that the remote node is available. If the node does not reply to the ping message within a certain amount of time then *unavailable* can be returned as the result of the call. In this case the remote call was not attempted, and thus no compensation needs to be performed.

In summary, the channel errors that can occur are as follows follows:

1) *Crash*: Communication with the remote node was lost during a call, and it is unknown if the operation was performed once, performed partially, or not performed at all.

2) *Unavailable*: The sink does not respond to ping messages and thus the requested operation was not attempted.

3) *Destroyed*: The channel that was called has been destroyed by its sink node. The operation was not attempted.

Channel errors are always reported to the source node. If a procedure call returns in an error, the error is returned as a distinguished value by the call. If a pipe call results in an error, the error is returned upon the next procedure call to a channel in the same group, or upon the next synchronize operation on a pipe in the group, whichever comes first. When a pipe call results in an error, any subsequent calls on pipes that are in the same group are discarded until the error is reported. Any calls that were made before the call that resulted in an error will have been performed exactly once.

5. REFINEMENTS

We discuss in this section two refinements to the basic model:

- explicitly serviced pipes,
- and stable channels that survive crashes,

5.1. Explicitly Serviced Pipes

An alternate model for the sink end of a pipe is to allow a program to create a pipe that is *explicitly serviced*. This is accomplished by using the procedure `Create` to create a pseudo-pipe sink procedure. However, unlike a pipe sink procedure, this procedure is not called when data arrives down the pipe. Instead, data must be explicitly taken from the pipe with the following procedures:

```
DEFINITIONS MODULE ExplicitPipe;
 TYPE Pipe = PROCEDURE ();
 PROCEDURE Create(): Pipe;
   (* Returns a new local procedure that can be used as
      a pipe sink *)
 PROCEDURE Value(p: Pipe): Any;
   (* Returns the next value from p.  Blocks if no value
      is present until a value arrives.  Successive applications of
```

```
    Value will return the same value unless Accept has
    been called *)
PROCEDURE Accept(p: Pipe);
  (* Accepts the last datum read with Value, and permits
    Value to get the next value from the pipe. Blocks its caller
    if no data has arrived for the pipe. Once Accept discards the
    present value, it does not block its caller waiting for the next
    pipe value *)
PROCEDURE Ready(p: Pipe): BOOLEAN;
  (* Returns TRUE if there is data waiting in p, FALSE otherwise *)
END ExplicitPipe.
```

A simple example of how a pipe can be explicitly serviced follows:

```
my-pipe := Create();
ftp.GetFile("fred.txt", my-pipe);
  (* pass the new pipe to remote source *)
c := Value(my-pipe);
Accept(my-pipe);
  (* remote source will terminate with 0 *)
WHILE c#CHR(0) DO
  term.PutChar(c);
  c := Value(my-pipe);
  Accept(my-pipe);
  END;
```

Accept is used to define the sequencing semantics of explicitly serviced pipes. Until **Accept** is called to acknowledge receipt of a call, further calls on the same channel will not be processed.

We call pipes which are connected to a procedure *procedure serviced*, and pipes which are polled *explicitly serviced*. We expect that both procedure serviced and explicitly serviced pipes will find application.

5.2. Stable Channels Survive Crashes

The above examples have shown how both remote procedures and pipes can be dynamically created, but their lifetimes have not been discussed. The desired lifetime of a channel depends upon its application. Thus in our model a *dynamic channel* will exist until it is explicitly destroyed by a program or until the channel's sink node crashes. An attempt to call a remote procedure which has been destroyed will result in a distinguished error value, and an attempt to call a pipe which has been destroyed will result in a *destroyed* error.

Channels which can survive node failures are useful for stable services that are

registered with a clearinghouse. We call a channel that can survive a node failure a *stable channel.* The state of a stable channel and its associated procedure must be recorded in stable storage to permit recovery of the channel upon node restart. The details of how stable channels are created will depend on the host language environment.

6. PRAGMATICS

We discuss in this section five pragmatic aspects of the Remote Pipe and Procedure Model:

- how connections are used to detect node crashes,
- failure recovery,
- how sequence numbers can be used to implement the channel timing invariant,
- normal call processing,
- and performance elaborations.

For the purposes of this section, we will assume that the message system may lose, reorder, and duplicate messages. However, we will assume that messages that are delivered are delivered without error. This ideal can be approximately with any desired level of reliability in practice by using larger and larger error detecting codes, and discarding messages with detected errors.

6.1. Connections are Used to Detect Crashes

Our crash detection and recovery algorithm is based upon *connections.* Connections provide a simple mechanism for detecting node crashes. A connection is a unique identifier that is shared between a source process and a sink that identifies the incarnation of the source and the sink. In order to implement crash detection, a node discards its connection state when it crashes. Thus the connection state of a node can be stored in volatile memory.

Before a source process makes its first call to a remote sink it must establish a connection with a two packet interchange. The source node first sends the sink the name of the process making the call, the source node identifer, and a proposed connection identifer (one packet), and then the sink acknowledges receipt of the connection identifer and other information (one packet). Each source process keeps track of its outgoing remote connections in a table that is indexed by remote node

identifer. Each sink node keeps track of its incoming remote connections in a table that is indexed by connection identifer.

Connection state can be garbage collected by both sources and sinks. A source can discard its connection state with a sink when no calls are in progress with the sink by simply forgetting the corresponding connection identifer. The source will have to reestablish a connection before it makes its next call to the sink. A sink can discard *source connection identifiers* if no calls are in progress with the corresponding source process. As outlined below, the next time that the source makes a call to the sink the source will discover that the connection has been garbage collected and create a new connection.

If a sink receives a call message with an unknown connection identifer then the sink sends back a distinguished *unknown connection* message to the source. The sink includes the unique identifier of the call message, the source process identifer, and the unknown connection identifer in the message.

A source will find itself in one of three states upon receipt of an unknown connection message:

- The first case is that the call message received by the sink was sent before the last source crash. In this case the source process identifier will be unknown to the source, and the unknown connection message is ignored.

- The second case is that the unknown connection message is in response to the first transmission of a call message and when the call was made the source process had no other outstanding calls to the sink. In this case a new connection is established with the sink and the call is retransmitted with the new connection identifer. Subsequent calls will use the new connection identifer.

- If neither the first or the second case apply, then the source must assume that the sink has crashed and recovered since the last successful call. Once the *crashed* failure is returned as the result of a procedure call, a new connection with the sink is established which will enable future calls to be processed.

6.2. Channel Failure Detection and Recovery

We guarantee that as soon as a failure occurs on a channel, no further operations on channels in the same group will be performed until the calling process is advised of the error. Here we examine how this guarantee can be implemented. There are two types of failures that we will consider in our discussion: the failure that results when a destroyed channel is called (a *destroyed* failure), and the failure that results when a sink crash occurs during the processing of a call (a *crash* failure).

Procedure call failures are directly reported to the call site by a distinguished return value. Thus, a failure during a procedure call will not cause future operations on the same group to be ignored because the failure is immediately reported.

Pipe call failures can not be immediately reported to the call site because the calling process does not block and wait for a return value. Thus, if a pipe call fails, all subsequent pipe calls to channels in the same group will be ignored until a procedure is called that is in the same group as the failed pipe call. The pipe failure will be immediately returned as a distinguished return value from the procedure call, and then subsequent pipe and procedure calls on the group can be processed. Note that synchronize is a special form of procedure call, and thus synchronize can be used to poll for pipe failures.

In the case of a pipe *destroyed* failure the sink must ignore future pipe calls on channels in the same group until a procedure call on the same group is made. The sink can of course advise the source of the pipe error, but the source will not be able to report it to the calling process until it makes a call on a procedure in the same group as the failed pipe.

In the case of a pipe *crash* failure the source must ignore future pipe calls on channels in the same group until a procedure call on the same group is made. If the pipe calls were not ignored, then it would be possible for the sink to recover and process subsequent pipe calls before the pipe failure was reported.

6.3. Sequence Vectors Implement Channel Timing

A new mechanism that utilizes vectors of sequence numbers can be used to implement the sequencing semantics for groups. Recall that in the Remote Pipe and Procedure model a client can dynamically create groups in order to force the sequential processing of calls made on independent channels. A channel value starts with a single sequencing stamp that is unique to the channel, and copies of the channel value can be freely made that include additional sequencing stamps to denote group memberships. Two channel calls will be processed in order if and only if the channel values share a sequencing stamp. We call this sequencing property the channel timing invariant.

One way to implement the channel timing invariant is to generate sequence numbers for each sequence stamp on a per-process basis. Using this method, when a call is made on a channel a new sequence number is obtained for each of the channel value's sequence stamps. This set of sequence stamps and sequence numbers is sent in the call message to the sink, along with the process identifier of the calling process. In order to guarantee the channel timing invariant, the sink will only process a call when all of the call's sequence numbers are one greater than the sequence numbers for already processed calls from the process identified in the call message. By "one greater" we mean that for each sequence stamp in the call message, the sequence number associated

with the stamp is one greater than the sink's copy of the sequence number for the corresponding stamp. For a given stamp, if the sequence number is 1 and the sink has no previous record of this stamp, then it initializes its sequence number for the stamp to be 0. At the end of a call, the sink increments the sequence numbers of the stamps that were sent with the call.

For example, consider the the following sequence of channel calls made by a single process. Each call is shown with the list of sequence stamp and sequence number pairs that is included in the call message:

```
Put("a");              <[1, #1]>
PutSeq("b");           <[1, #2], [2, #1]>
ColorSeq(blue);        <[3, #1], [2, #2]>
Put("c");              <[1, #3]>
PutSeq("d");           <[1, #4], [2, #3]>
```

From the stamps that are sent in each call messages shown above, we can determine the sequencing semantics of the channel calls. Because `ColorSeq` and `Put` do not share a sequence stamp their processing is unordered. `PutSeq` refers to the same underlying channel as `Put` (identified by sequence stamp 1), and thus calls on these two channels will be ordered. In addition, `PutSeq` has been extended with the sequence stamp 2 which is used to group `PutSeq` and `ColorSeq` in order to guarantee that calls on these two channels will be ordered.

In sum, to implement the normal sequencing of calls, the following state must be maintained by a source and a sink:

- The source must keep for each outgoing connection a sequence stamp to sequence number table. When a local process makes a call appropriate entries in the outgoing sequence table are incremented, and the new sequence numbers are sent with the call to the sink. The first time that a sequence stamp is used by a local process an entry is placed in the process' table with sequence number 1.

- The sink must keep a sequence stamp to sequence number table for each incoming connection. Each request that is received is checked against the corresponding connection's table, and if the sequence stamps do not match the request is rejected. After a request is processed, appropriate entries in the connection's table are incremented. When a sink establishes a new connection it creates a fresh table with no sequence stamp entries. If a sequence stamp is not in a table, an entry with a sequence number of 1 will be automatically generated.

It is possible for either a source or a sink node to discard sequence number tables by

garbage collecting the corresponding connection. The rules for garbage collecting connections were outlined in the last section.

6.4. Normal Call Processing

Here we recap the information that is sent in every call message. A call message includes: (1) the sink node address (for the communication system), (2) the connection identifier (to identify the source node and calling process), (3) a unique call identifer that is different for each retransmission of the call (to determine if a new connection can be opened if the call fails), (4) a sequence vector (to sequence the call), (5) the identifer of the channel being called, and (6) a byte string that represents the data for the channel. We assume that typed values can be converted to byte strings and back again via encode and decode operations [10]. Specific versions of encode and decode must be designed to work with each programming language in order to properly handle the language's type space and exception handling discipline.

A call message is retransmitted until a corresponding return message is received. A return message includes: (1) the source node address (for the communication system), (2) the connection identifer (to identify the calling process), (3) the unique call identifer (to identify the call); and (4) a byte string that represents the result data.

6.5. The Performance of the Model Implementation Can Be Improved

The model implementation we have described is intended only to be suggestive; a practical implementation of the Remote Pipe and Procedure Model would require performance optimizations. Important optimizations include:

- *Combine pipe calls*: Multiple pipe calls destined for the same sink node can be buffered at a source and transmitted as a single message in order to reduce message handling overhead and improve network throughput. The amount of time that a pipe call is buffered before it is sent presents a tradeoff between low pipe latency and efficient bulk communication. A moving window flow control algorithm can be employed [Postel79] to manage the transfer of buffered pipe calls between a source and a sink.

- *Combine pipe calls with procedure calls*: A procedure call message will always be transmitted immediately, and any buffered pipe calls to the same sink should be prepended to the procedure call message whenever possible.

- *Combine pipe returns*: Because returns from pipe calls are only used to acknowledge the completion of processing, multiple pipe call returns can be combined into a single message that acknowledges the processing of a set of calls.

- *Preallocate Processes*: Processes can be preallocated into a process pool at node startup so that performance of a FORK operation for each incoming remote call is not required. Eliminating FORK overhead on is especially important for a collection of pipe calls that arrive in a single message, because the overhead per pipe call is limited to approximately the cost of a procedure call, as opposed to a process creation. A process allocated from a pool would return itself to the pool when the process had finished processing its assigned call message.

- *Explicitly Acknowledge Messages*: At times both call and return messages should be explicitly acknowledged in order to improve performance. A call message should be explicitly acknowledged by a sink when the sink has been processing a call for a predetermined interval without a result having been produced. This acknowledgment informs the source that the call has been successfully received, and that the source does not need to retransmit the call message. A procedure return message from a sink should be explicitly acknowledged by a source when the same source process does not make a subsequent procedure call to the sink within a predetermined interval. This informs the sink that the return message has been received by the source, and that the sink can discard the result contained in the return message.

- *Factor Packages and Groups*: In order to save space, information that is common to all of the channels in a package or group value need only be represented once.

7. PRACTICAL EXPERIENCE AND CONCLUSIONS

We conclude with

- analytical performance bounds and empirical performance results,

- experience with an application of the Remote Pipe and Procedure Model that has proven certain of its elements practical,

- and discussion about general application of the model.

7.1. Performance Bounds and Results

We show here that k pipe calls can be at most k times as fast as k identical procedure calls, and we compare this performance bound to empirical data gathered from an implementation of the Remote Pipe and Procedure Model. Experimental data confirms the analytical model as long as the source does not generate data faster than they can be transmitted to the sink. Performance is substantially degraded from analytical

predictions when the source is generating data faster than they can be transmitted to the sink.

First we derive elementary performance bounds on the ratio of time spent processing k procedure calls vs. k pipe calls followed by a `synchronize`. Figure X schematically breaks down the time spent processing a pipe and a procedure call into individual components as follows:

- Each procedure call has a processing cost at the source (send and receive), a round trip network latency delay, and a processing cost at the sink node (receive, execute call, and send). If we let s be the cost of the source send, and r be the cost of the source receive plus the cost of a round trip network latency plus the processing cost at the remote node, then k procedure calls will take $ks + kr$ units of time.

- A series of k pipe calls requires k source sends, and a following `synchronize` requires a round trip network latency plus the processing cost at the remote node plus a source receive. Thus the cost of k pipe calls followed by a `synchronize` is $ks + r$. This assumes that r is larger than s. If s is larger than r then the cost is $s + kr$. We can ignore this case without loss of generality by symmetry.

With these estimates of the cost of pipe and procedure calls we can estimate the time ratio between k procedure calls and k pipe calls followed by a `synchronize` as:

$$A(k,s,r) = \frac{ks + kr}{ks + r}$$

This equation for $A(k,s,r)$ shows that for fixed r, as the source send time decreases, the performance ratio between pipes and procedures is bounded by

$$\lim_{s \to 0} A(k,s,r) = k$$

Furthermore, for fixed s and r, the performance ratio is bounded by

$$\lim_{k \to \infty} A(k,s,r) = 1 + \frac{r}{s}$$

Combining these two independent bounds yields the single bound

$$A(k,s,r) \leq \max\left(1 + \frac{r}{s},\ k\right)$$

This performance bound can not be realized when the source generates data faster

than it can be transmitted to the sink. This will occur when data is generated at a rate that is greater than the bandwidth of the channel to the sink or when insufficient buffering is provided at the source. In this case additional queuing delays will result that add to the network delay time, and the performance ratio of pipes to procedures will be reduced.

A series of experiments were run on an implementation of the Remote Pipe and Procedure Model to measure the relative performance of pipe and procedure calls. The implementation tested uses TCP as its underlying transport layer, and limits all channels that are exported as a service to be in a single group. Further details of the implementation are given in [9].

The experiments that we ran consisted of making 10, 50, 100, 500, and 1000 pipe and procedure calls to test the effect of varying k. For each number of calls 0, 100, and 1000 byte arguments were used to test the effect of argument size. Each experiment that tested a combination of number of calls, argument size, and pipe or procedure was repeated 10 times.

We first ran these experiments in a low network delay environment. Low-delay was provided by running the tests on two Microvax-II workstations on the same 10 MBit/second Ethernet network. In this environment, the average time to perform a 0 byte argument remote procedure call was 13.3 milliseconds, while the average time to perform a 0 byte argument remote pipe call was 4.3 milliseconds.

As shown in the following table, the measured pipe to procedure performance ratio increases with k as predicated by our simple model. The value for $A(k,s,r)$ was computed by assuming that for 1000 calls that the procedure time was $s + r$ and the pipe time was s. Note that the performance improvements are all greater than a factor of two.

We repeated the same experiments in an environment with a large communication channel latency by employing one node at MIT as the sink and a second node at a distant Arpanet site as the source. The results are shown in the table below. The average time for a null procedure call was 748 milliseconds, which is far longer than the 13 milliseconds measured in the Ethernet case. Because the source was generating data faster than the Arpanet could absorb it we did not expect the measured performance ratios to be consistent with our simple performance model. In fact, the underlying TCP protocol exhibits pathological behavior under these circumstances as shown by the net performance reduction of pipes when compared with procedures in the case of 1000 byte arguments. This pathological behavior results from TCP's window based flow control algorithm. Instead of TCP's window based approach, a rate based flow control algorithm should be employed under these circumstances.

In sum, k pipe calls can perform at most $max(1 + \frac{r}{s}, k)$ times better than k procedure calls, where r is the remote communication and processing time and s is the source send

Low Latency Experiments Single Ethernet							
Bytes per Call	Number of Calls k	Procedure Call Time (seconds)	90% Conf. Interval	Pipe Call Time (seconds)	90% Conf. Interval	Pipe/ Procedure Ratio	A(k,s,r)
0	10			0.056	0.006	2.37	2.54
0	50			0.297	0.131	2.24	2.94
0	100			0.439	0.060	3.09	3.00
0	500			2.155	0.010	3.08	3.05
0	1000	13.277	0.034	4.341	0.008	3.06	3.05
100	10			0.062	0.003	2.28	3.23
100	50			0.255	0.004	2.77	4.02
100	100			0.437	0.006	3.23	4.15
100	500			1.178	0.013	4.11	4.26
100	1000	14.136	0.029	3.300	0.010	4.28	4.27
1000	10			0.088	0.005	2.06	2.19
1000	50			0.374	0.005	2.43	2.45
1000	100			0.777	0.065	2.34	2.49
1000	500			3.615	0.076	2.51	2.52
1000	1000	18.153	0.213	7.175	0.015	2.53	2.52

Table 9-1: Low Latency Experiments: Single Ethernet

time. This bound can be approached in practice when sufficient bandwidth is available. Thus, pipes can provide a substantial performance advantage over procedures for many applications.

High Latency Experiments Long-Haul Arpanet						
Bytes per Call	Number of Calls k	Procedure Call Time (seconds)	90% Conf. Interval	Pipe Call Time (seconds)	90% Conf. Interval	Pipe/ Procedure Ratio
0	10			3.781	0.349	1.98
0	50			3.931	0.308	9.52
0	100			3.876	0.277	19.3
0	500			6.799	0.351	55.0
0	1000	748.22	60.760	16.289	1.726	45.9
100	10			4.601	1.177	1.62
100	50			20.364	6.047	1.83
100	100			55.907	7.619	1.33
100	500			165.30	23.89	1.42
100	1000	745.84	29.81	531.24	128.18	1.42
1000	10			113.91	25.41	0.09
1000	50			152.95	23.94	0.34
1000	100			251.83	15.05	0.41
1000	500			1252.1	70.4	0.30
1000	1000	1034.3	54.9	3504.5	1329.6	0.41

Table 9-2: High Latency Experiments: Long-Haul Arpanet

7.2. The Elements of the Model Have Been Proven Practical

In order to gain experience with the Remote Pipe and Procedure Model we have used it to implement a distributed database system. The database system we implemented provides query based access to the full-text of documents and newspaper articles, and is presently in use by a community of users. The database system is divided into a user interface portion called Walter that runs on a user's local node, and a confederation of

remote database servers which are accessed by Walter via the DARPA Internet. Walter employs a query routing algorithm to determine which server contains the information required for processing of a given user query.

The protocol that Walter uses to communicate with a database server is built using the Remote Pipe and Procedure Model. A stub generator automatically generates both source and sink node stubs from an interface file. This interface file (rewritten from C into Modula-II) is shown below:

```
REMOTE INTERFACE DataBase;
  PIPE EstablishQuery(c: ARRAY[CHAR]);
    (* establishes a new query *)
  PROCEDURE CountMatchingRecords(): INT;
    (* Returns the number of records that have matched so far *)
    (* Number is positive if query processing is complete *)
  PROCEDURE FetchSummary(r: Range, dest: PIPE[Summary]);
    (* Causes summaries in range r to be sent to dest *)
  PROCEDURE FetchRecord(rec: INT, r: Range, dest: PIPE[Line]);
    (* Causes lines in range r of record rec to be sent
       to dest *)
  PIPE Abort();
    (* Causes the server to abort the query and any data that it
       is sending down a pipe *)
END DataBase.
```

When a user supplies a query the procedure `EstablishQuery` is called. `EstablishQuery` initiates processing of a query at a server. The server procedure `FetchSummaries`, which computes the summaries for a range of articles matching the current query is then called. As the summaries are computed they are sent down the pipe supplied in the `FetchSummaries` call. The pipe sink procedure that receives the summaries displays them as they arrive. All of the summaries generated by `FetchSummaries` are guaranteed to arrive before `FetchSummaries` returns. In order to view an entire database record the server procedure `FetchRecord` is used in precisely the same manner as `FetchSummaries` is used.

A second process is conceptually running concurrently with the information which is arriving down a pipe and being displayed. This process checks for the abort user request, which aborts the query in progress. If such a keyboard request is received, the `Abort` procedure is called to abort the `FetchSummaries` or `FetchRecord` operation in progress.

The use of pipes in this database application has provided two distinct advantages over remote procedures. First, pipes permit both `FetchSummaries` and `FetchRecord` to send variable amounts of bulk data to Walter simply. Second, since pipe calls do not block a server can continue computing after it has sent a datum. If a procedure instead

of a pipe were used to return data the server process would suspend processing while waiting for a response from Walter. The concurrency provided by pipes has proven to be important to Walter's performance in practice.

7.3. The Remote Pipe and Procedure Model Has Many Advantages

We have proposed three major ideas:

- *Channel values*: Channels should be first-class values which can be freely transmitted between nodes. If a communication model does not permit channel values to be transmitted between nodes, then its application will be limited to a restricted set of protocols. An application of channel values is the return of incremental results from a service to a client.

- *Pipes*: A new abstraction called a pipe should be provided in the communications model. A pipe permits bulk data and incremental results to be transmitted in a type safe manner in a remote procedure call framework. Existing remote procedure call models do not address the requirements of bulk data transfer, or the need to return incremental results.

- *Channel groups*: A new sequencing technique, the channel group, is important in order to permit proper sequencing of channel calls. A channel group is used to enforce serial sequencing on its members with respect to a single source process.

As we have explained these three ideas form the basis for the Remote Pipe and Procedure Model. We expect that this model will find a wide variety of applications in distributed systems.

Acknowledgments: The ideas in this paper benefited from meetings with fellow MIT Mercury Project members Toby Bloom, Dave Clark, Joel Emer, Barbara Liskov, Bob Scheifler, Karen Sollins, and Bill Weihl. I am especially indebted to Bob Scheifler for posing a question that resulted in the notion of a channel group. Barbara Liskov, John Lucassen, Bob Scheifler, Mark Sheldon, Bill Weihl, and Heidi Wyle commented on drafts of the paper.

References

1. Bershad, B., et al. "A Remote Procedure Call Facility for Heterogeneous Computer Systems." Technical Report 86-09-10, Computer Science Department, University of Washington, September 1986.

2. Birrell, A. and B. Nelson. "Implementing Remote Procedure Calls." *ACM Transactions on Computer Systems*, 2, 1, (February 1984), 39-59.

3. Birrell, A. "Secure Communication Using Remote Procedure Calls." *ACM Transactions on Computer Systems*, 3, 1, (February 1985), 1-14.

4. Brown, M., et al. "The Alpine File System." *ACM Transactions on Computer Systems*, 3, 4, (November 1985), 261-293.

5. Cheriton, D. "VMTP: Versatile Message Transaction Protocol." Computer Science Department, Stanford University, Stanford, CA, January 12, 1987.

6. Gifford, D. "Information Storage in a Decentralized Computer System." Report CSL-81-8, Xerox Palo Alto Research Center, Palo Alto, CA.

7. Gifford, D., et al. "An Architecture for Large Scale Information Systems." *Proceedings of the Tenth ACM Symposium on Operating Systems Principles* and *ACM Ops. Systems Review*, 19, 5, 161-170.

8. Gifford, D. "Remote Pipes and Procedures for Efficient Distributed Communication." MIT/LCS/TR-384, MIT Laboratory for Computer Science, Cambridge, MA, October 1986.

9. Glasser, N. "The Remote Channel System." M.S. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

10. Herlihy, M. and B. Liskov. "A Value Transmission Method for Abstract Data Types." *ACM Transactions on Programming Languages and Systems*, 4, 4, (October 1982), 527-551.

11. Jones, M., et al. "Matchmaker: An Interface Specification Language for Distributed Processing." *Proceedings of the 12th Annual ACM Symposium on Principles of Programming Languages*, January 1985, 225-235.

12. Liskov, B. and R. Scheifler. "Guardians and Actions: Linguistic Support for Robust, Distributed Programs." *ACM Transactions on Programming Languages and Systems*, 5, 3, (July 1983), 381-404.

13. Needham, R. and M. Schroeder. "Using Encryption for Authentication in Large Networks of Computers." *Communications of the ACM*, 21, 12, (December 1978), 993-998.

14. Nelson, B. "Remote Procedure Call." Report CSL-81-9, Xerox Palo Alto Research Center, Palo Alto, CA, May 1981.

15. Postel, J. "Internetwork Protocols." *IEEE Transactions on Communication*, COM-28, 4, 604-611.

16. Redell, F. "Experience with Processes and Monitors in Mesa." *Communications of the ACM*, 23, 2, (February 1980), 105-117.

17. Voydock, V. and S. Kent. "Security Mechanisms in High-Level Network Protocols." *Computing Surveys*, 15, 2, (June 1983), 135-171.

18. White, J. "A High-Level Framework for Network-Based Resource Sharing." *Proceedings of the National Computer Conference*, 1976, AFIPS Press, 561-570.

19. White, J. and Y. Dalal. "Higher-Level Protocols Enhance Ethernet." *Electronic Design*, 30, 8, (April 1982), 33-41.

20. "Courier: The Remote Procedure Call Protocol." Xerox System Integration Standard XSIS 038112, Xerox Corporation, Stamford, CT, December 1981.

Publications

1. Gifford, D.K. and A. Z. Spector. "The Evolution of the IBM System 360/370 Architecture." *Communications of the ACM*, 30, 4, (April 1987), 291-307.

2. Gifford, D.K. "Processor Lattices." December 1986.

3. Gifford, D.K. "Remote Pipes and Procedures for Efficient Distributed Communication." MIT/LCS/TR-384, MIT Laboratory for Computer Science, Cambridge, MA, October 1986.

4. Gifford, D.K., J.M. Lucassen, S.T. Berlin, D.E. Burmaster, J.B. Henderson and D.A. Segal. "Boston Community Information System - User Manual." MIT/LCS/TR-373, MIT Laboratory for Computer Science, Cambridge, MA, September 1986.

5. Gifford, D.K. and J.M. Lucassen. "Integrating Functional and Imperative Programming." *Proceedings of the 1986 ACM Conference on LISP and Functional Programming*, August 1986.

Theses Completed

1. Baldwin, R. "Rule Based Analysis of Large Protection Systems." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

2. Cote, R. "An Automatic News Article Editor." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

3. Glasser, N. "The Pipe Extension to Remote Procedure Calls." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

4. Rao, R. "A Flexible and Portable Window System for Common LISP." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

5. Slivan, S. "An Interactive Graphics Interface for Clinical Applications Software." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, August 1986.

Theses in Progress

1. Braunstein, A. "New File System." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1987.

2. Brondmo, H.P. "User Interface of a Dynamic Proton Synchrotron Control System." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1987.

3. Lucassen, J. "Types and Effects -- Integrating Functional and Imperative Programming." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected August 1987.

4. O'Toole, J. "Type Inferences in Fluent Programming Languages." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected January 1988.

5. Sheldon, M. "Standard Types for a Fluent Language." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1987.

Talks

1. Lucassen, J. "Types and Effects -- Integrating Functional and Imperative Programming." Bell Laboratories, Holmdel, NJ, June 1987.

2. Gifford, D. "Processor Lattices." MIT Parallel Computing Research Meeting, Endicott House, Dedham, MA, February 1987.

3. Gifford, D. "Applications of the MIT Common System." DARPA Presentation, December 1986.

4. Gifford, D. "Communication Models for Distributed Computation." Advanced Study Institute, Izmir, Turkey, August 1986.

5. Gifford, D. "An Architecture for Large Scale Information Systems." TTI Study Mission on Videotex - MIT-ILO Visit by Japanese Companies, MIT, Cambridge, MA, May 1986.

6. Lucassen, J. "An Architecture for Large Scale Information Systems." MIT-ILO Visit by IBM, MIT, Cambridge, MA, March 1986.

REAL TIME SYSTEMS

Academic Staff

M. Dertouzos
S. Ward, Group Leader
R. Zippel

Research Staff

J. Pezaris
M. Singh
R. Zak

Graduate Students

R. Anderson
M. Blair
S--L. Ku
J. Miller
M. Powell
J. Sieber
C. Staelin
M. Wong
A. Ayers
D. Goddeau
B. Kuszmaul
J. Morgan
S. Seda
P. Spacek
S. Weiner
F. Zhao
R. Baldwin
B. Guharoy
W. Lar
P. Osler
C. Shepard
M. St. Pierre
J. Wolfe

Undergraduate Students

L. Candell
C. Forsythe
R. Hagen
J. Hunt
J. Kipnis
M. Matchett
R. Opie
J. Tabor
S. Chang
K. Gartner
J. Halebian
R. Jenez
D. Kwon
J. Nguyen
M. Powell
K. Tamura
J. Chung
S. Geels
H. Houh
B. Kartzman
K. Mackenzier
J. Ofori
B. Swiston

Support Staff

S. Thomas

Visitors

D. Cerys
R. Saenz

1. INTRODUCTION

Consolidation of RTS research goals has marked the 1986-7 academic year, owing primarily to personnel changes. The departure last year of Profs. Terman and Zippel is leading RTS research in the areas of VLSI CAD tools and the database accelerator to an orderly finish. The divestment of Prof. Halstead's project as a separate research group leads progress on CONCERT and MULTILisp to be reported elsewhere.

Major foci of RTS research efforts have been continued work on (1) the L architecture and (2) alternative memory system architectures, with seed projects in learning and modeling. An undercurrent of NuBus-related projects continues an eight-year RTS tradition, encouraged somewhat by several conspicuous recent proselytes.

2. THE L ARCHITECTURE

Continuing development of L, a minimalist architectural model for multithreaded, object oriented computation, was a primary focus of research by Ayers, Saenz, Singh, and Ward.

2.1. Microcoded L Simulation

The microcoded emulation of a single-processor L system became operational during the reporting period, due primarily to the work of Ayers, together with primitive versions of a number of associated software tools [Ayers, Saenz, Singh]. The current implementation runs on T.I. Explorer processors, co-resident with the T.I. Lisp machine microcode. While the basic interpretive structure of L (including chunk management and elementary fetch/execute mechanism) is directly supported, the current system freely uses escapes to Lisp to bridge various gaps in the implementation. While this approach has been a valuable bootstrapping and prototyping tool, it is encumbered by resource and performance limitations which we hope to mitigate in the next implementation (see following section).

The current system integrates volatility-based garbage collection with a local chunk cache in a scheme which we hope to extend to more ambitious implementations. The cache is non-associative, and constitutes a chunk namespace local to the processor. Local names are assigned chunks when they are (1) locally allocated or (2) imported from an external namespace object. Chunks of the former class need not have external names at all, allowing them to be garbage collected without having undergone the expense of exportation. The highest volatility level is thus kept entirely within the local cache. The processor deals only with local names, which are typed pointers into the local cache. Processor references to chunks by external names are dynamically trapped, causing importation of the referenced chunk if necessary and substitution of its local name in the cached chunk containing the offending reference.

This approach has a number of virtues. Various aspects of storage management

overhead (such as synchronization, virtual memory management, protection checking, etc) can be bundled into the importation and exportation operations and thereby removed from critical paths of instruction execution. So long as the working set of an active computation fits into the processor cache, its memory references can be direct and unencumbered, leading to a streamlined execution pipeline and fast clock speeds.

2.2. Multiprocessor L Prototype

The foundations have been layed for a reimplementation of the L emulation in a less restrictive context [Ayers, Pezaris, Singh]. This approach is again based on T.I. Explorer processors, but with the entire resources of a single processor board devoted to the emulation of a single L processor. Pursuant to the new implementation, we have configured a large NuBus chassis to accept multiple cards, and have arranged for a single processor running Lisp to bootstrap secondary L-based processors. This approach lends considerable flexibility to our L implementation, which can now expand beyond the microstore corner unused by Lisp. It also makes a clean break from our current Lisp dependency, perhaps leading eventually to standalone L systems runnable on ordinary explorer hardware. Most importantly, it provides a testbed for extension of the L system to modest multiprocessor dimensions.

2.3. L Base Language

Although the L architecture is designed to support a wide range of programming languages, our current language technology focuses on a single Lisp-like *base language* tuned to exploit the unique characteristics of L. To a rough first approximation, the language is *SCHEME* adapted to a compiler-intensive implementation. In particular, it deviates from *SCHEME* at the source language level primarily in its facilities for optional type declarations. While defaulting to *SCHEME* semantics in the absence of declarations, our dialect is intended to afford the efficiency of, say, *C* to programmers willing to adhere to *C* programming idioms.

The base language implementation goes to unusual lengths to semantically integrate compile- and run-time environments. It is completely devoid of interpretive mechanism; even the most trivial of evaluations are performed by compilation followed by execution. A single lexical environment governs execution and compilation: a form is both compiled and executed relative to an environment, and the compiler is effectively distributed about the environment by the same mechanism that governs other semantic bindings. Thus the form *(fn args)* is compiled relative to an environment by looking in that environment for a *form compiler* for *fn* and invoking it; the meanings of *quote*, *lambda*, and *cond* are dictated by environmental bindings to compile-time mechanism. As a special case, traditional Lisp *SUBRs* are bound to form compilers which invoke shared interpretive means (presumably by producing calls to a shared procedure); macros, inline-coded functions, and various language extensions are similarly simple special cases. Of particular interest is the ability this approach affords to sophisticated

optimizations, particularly those crossing the boundaries of a procedure call. In commonly occurring cases, a form compiler which deals with a call to a procedure can exploit detailed knowledge about the internal structure of that procedure, alternative entry points for polymorphic procedures, organization of local environment, etc.

2.4. Metacache Architecture and Implementation

Work by Singh has led to a preliminary VLSI implementation of a prototype *metacache*, an architectural feature intended to improve the performance of L and similar processors heavily dependent on accessing data through bounded-length indirection paths. The unit of storage in L is the chunk, a 9-tuple whose elements may each be a scalar or a reference to another chunk. Since objects in L are made of interconnected chunks, a natural memory accessing mechanism is the "metaname". A metaname is a path into memory starting at some initial chunk and ending at some final point. A typical metaname in L is of the form (1 4 5 6). Accessing is always done relative to a root chunk which is usually dictated by the context of the access (typically the STATE chunk corresponding to a computation). Given this type of memory structure, it is straightforward to design a cache that simply stores values based on some function of a given metaname, where the metaname is provided by the L processor during a memory access.

There are at least two interesting features specific to this cache however. The first is that since memory accesses will most likely display some sort of locality of reference, it might be worthwhile on a cache miss to iteratively cache the object pointed to by each partial metaname. Secondly, care must be taken in how the metacache handles the problem of objects accessed via multiple paths. This problem is of course general in the sense that there may be more than two paths which converge at any level in the metaname and writes to the terminal locations of these paths must be carefully treated.

An approach to this problem is to use a good hashing scheme based on an "encoded dependency word" to allow the invalidation of entries based on path convergence. The dependency word for each cache entry yields a conservative approximation of that entries reliance on the contents of several chunks, by ORing hashed derivatives of identifiers of these chunks. When a chunk element is changed a corresponding invalidate code is broadcast, causing each cached value possibly dependent on the modified element to be marked invalid for future reference.

Simulation of systems based on this approach led to a promising design, a modest sample of which (8 words) has been implemented in CMOS.

3. MEMORY SYSTEMS ARCHITECTURE

Continuing work on memory systems architecture by Ward and Zak has led to a nonstandard architecture for main memory systems which promises potentially

improved cost/performance characteristics. The typical cached memory system uses a modest amount of fast-access static RAM as a buffer between the processor and a much larger array of slower, denser dynamic RAM devices. The benefits of a large cache are average access times approaching that of the fast static memory. Its costs include the static RAM whose contents are redundant with the selected portions of dynamic memory, additional *tag* memory used to record addresses of cache contents, as well as comparators and other associated logic.

Modern static column dynamic RAM (SCRAM) chips combine a fast static *row buffer* with a much larger dynamic charge array. An entire row of the two-dimensional array (dictated by high-order address bits) is loaded into the row buffer on the RAS signal. Subsequent accesses to memory elements within that row may be made at static RAM speeds, since they deal only with the static buffer. The effect is to achieve very fast access times to consecutive locations within the same row, while exploiting DRAM densities for the bulk of the memory.

The static row buffer is reminiscent of a cache, and its use as such has been proposed by Goodman and Chiang. An alternative system architecture, reported as RTS progress last year and refined further recently, extends SCRAM cache usefulness by achieving the performance characteristic of a virtual rather than physical cache. Each of these schemes associates a register (or, equivalently, a location in a dedicated *tag memory*) with each of several memory banks. The contents of each register reflects the current row of the associated bank, allowing a sequence of accesses to the same row (within a given bank) to be detected and the intervening RAS cycles avoided. The effect of this approach, is equivalent to the use of a direct-mapped cache whose line size is equal to the size of the row buffers of a single bank.

While these techniques can be usefully employed with conventional static-column RAM parts, they suffer from two limitations: (1) the total effective cache size is dictated by the size of the RAM static row buffers, which is typically smaller than desirable; and (2) the direct-mapped nature of the effective cache results in serious contention problems, lowering overall performance. These latter contention problems arise when two or more frequently accessed locations reside in different rows within the same memory bank. If they are accessed (say) alternately, then the time overhead associated with a RAS cycle will burden each memory access since no two consecutive accesses to that bank refer to the same row.

3.1. Set-associative DRAMs

The basis for our proposed memory system is a new dynamic memory device, the *set-associative dynamic RAM* or *SADRAM*, which features a multiplicity of on-chip static row buffers as shown in Figure 10-1.

The SADRAM chips will typically exploit technology similar to that used in conventional static-column parts, and have similar timing and I/O provisions. However,

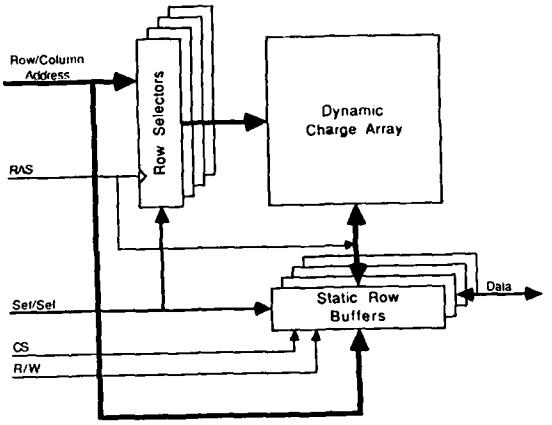


Figure 10-1: Set Associative DRAM

each chip will have 2^n independent row buffers, and n additional input pins to select between them. In compliance with accepted cache terminology, we refer to 2^n as the *set size.*

The binary data on the n *set select* inputs is significant during static accesses as well as each edge of RAS. During static accesses, the additional inputs select which of the row buffers is to be accessed. On the rising edge of RAS, they select the row buffer to be *filled* from the row (dictated by the address lines) of the dynamic charge array; on the falling edge of RAS, they select the row buffer to be *written* into the addressed row of the charge array. Thus by use of appropriate off-chip circuitry, the 2^n static buffers of each SADRAM may be used to hold a corresponding number of arbitrarily chosen rows of memory.

One use of these devices involves the straightforward extension of a previous system to n-way set associativity, as depicted in Figure 10-2. Systems restricted to caching physical locations may be implemented somewhat more simply.

Note that in this illustrative system, we have chosen a set size of 4 and assume the use of 256K by 4-bit devices. Each virtual memory address is simultaneously compared with the contents of a register associated with each *row* of each memory bank; thus the comparator logic grows in proportion to the product of the number of banks and the set size. If any comparison reveals a match, the corresponding bank and row buffer are selected and the row buffer is directly accessed. If no match occurs, a *miss* condition causes access via the mapped (*physical*) address to the appropriate bank, cycling RAS to rewrite the current row and to select a new one containing the accessed location. Note that the new row can be read into any of the 4 row buffers of the selected bank; this choice is made according to a *replacement strategy* whose details reflect logic not

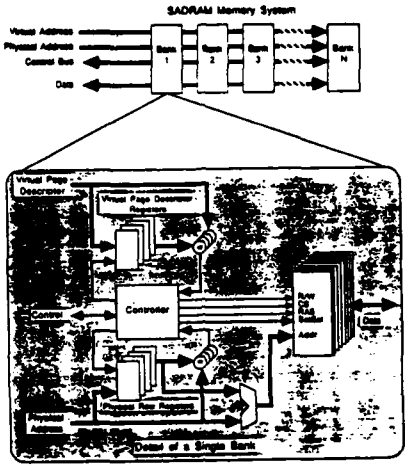


Figure 10-2: Virtual Set-Associative SADRAM Memory System

shown. Standard cache replacement strategies (and their usual implementations) are appropriate here, including *least recently used* or *pseudo-random* replacements.

We have demonstrated many elements of the above system in a prototype 68020 system which outperforms commercial equivalents having dramatically higher parts counts. While we are not in a position to prototype actual SADRAM chips, extrapolation of empirical performance data suggests very promising results using these parts, as a primary cache in small systems or as a secondary cache in larger ones. We are exploring additional techniques for collecting performance data under a variety of practically interesting assumptions, in an effort to reinforce our argument for the commercial assimilation of these ideas. For similar reasons, we have applied for a patent.

Aside from its intrinsic architectural novelty, we view our proposed system as an example of a class of technological innovation which is both important to the domestic semiconductor industry and difficult for that industry to assimilate. Its importance stems from the alternative it offers to the permanent abdication of the commodity chip market to international competitors offering production advantages. It defines a new commodity, moving the competitive focus to the architectural arena where we seem better equipped. The difficulty of such proposals is that they dictate simultaneous innovation at the system and the component levels, in rude defiance of the "socket counting" marketing policy that the semiconductor industry has retrenched to.

4. SEED PROJECTS

The exploration of alternative models of computation continues as the common theme among RTS projects. The following paragraphs sketch two new efforts along these lines.

4.1. Machine Learning

The goal of this new research by Goddeau is to develop a memory/computational model capable of learning to perform certain cognitive tasks by example. That is, having been taught with a set of examples of input/output (problem/solution) pairs from a problem domain, the system should be able to solve additional problems from that domain. This research focuses primarily on exploring the underlying mechanisms of learning and the architecture of learning systems.

Although this research is not directed towards human learning, the model is inspired by observations on human abilities. One of these is the importance of memory relative to computation. Humans are clearly able to remember and retrieve vast amounts of information with little conscious effort. By contrast, the low-level computational mechanisms (relative to cognition rather than vision or hearing) are very difficult to characterize. One of the underlying principles of this research is to concentrate on memory and retrieval mechanisms, and minimize the complexity of the required computational machinery.

To contrast memory-based learning by example with a computation-based approached consider the problem of predicting the horizontal travel of a projectile launched with a given speed and angle of inclination. A computation-based approach might assume the function $d = f(speed, angle)$ could be approximated by a polynomial, the coefficients of the polynomial would be determined from the examples presented and new problems solved by evaluating the polynomial given data. A memory-based approach might simply remember all of the examples and use linear interpolation based on the known examples to solve new problems. Assuming the examples are well chosen, both methods will converge on the correct answer as the number of examples increase (assuming the function is continuous over the region of interest).

The problem domain chosen for development work is the syntax of natural languages. The knowledge to be acquired in this domain is the ability to translate a sentence in a given natural language into a Meaning Representation Language expression or vice versa. The system should be capable of learning any human language by example with no prior knowledge of the language.

Clearly memory alone is not sufficient for learning language, since the system must be able to interpret and produce sentences it has not encountered before. The characterization of the computational abilities needed to perform this task is one of the major goals of the research. The underlying principle in developing computational mechanisms is that new sentences are interpreted based on similar, known sentences. These mechanisms include interpretation through "analogy" with known examples, abstraction of examples into patterns, classification of subunits, and association of subunits with corresponding translations. Since natural languages are often less consistent than the examples given above, the structure of patterns learned will be more complex, with exceptions to patterns, exceptions to exceptions, and hierarchies of

patterns. Furthermore the learning system must converge on a correct set of rules even though some of the training examples may demonstrate spurious relationships. The system must have the ability to forget some forms as additional data shows them to be incorrect.

An initial version of the learning system has been built during 1987. This includes an "associative" memory/retrieval system, abstraction and classification mechanisms, and a control structure which uses them to provide translations. This initial system has been tested on the problem of learning verb conjugation and on translating and generating simple sentences, with encouraging results. Future work includes research on exception hierarchies, multiple paradigms, recursive patterns, "forgetting" and system convergence, as well as extensive testing on a diverse set of languages.

4.2. Modeling

New research by Sieber is aimed at developing the first phase of a system that can "understand" some portion of the world. The rather amorphous notion of "understanding" is based upon the ability to combine available information about the local structure of the parts of the world to build a description of its overall structure. This overall structural description can then be analyzed to determine the behavioral characteristics of the system.

This ability to create a picture of the whole given an understanding of its parts will be combined with the ability to compare two views of the same system and see the ways in which they differ. The result will be a system that can build a model of the world and either simplify it to a similarly behaving model with a less complex description, or determine what aspects of the world the simpler model doesn't capture. This allows us to build plausible theories and then verify whether they can in fact explain the behavior of the system.

Initially this goal will be accomplished by developing a framework in which interesting aspects of the world can be formally described by a human modeler with the help of a computer. The second phase of the project will allow the computer to take a more active role in manipulating (analyzing and modifying) the model to extend its understanding.

The tangible product of the first phase will be to develop a dynamic system modeling tool. This tool will make it relatively easy to describe and manipulate (model and simulate/analyze) both simple systems and those that are so complex that they can only be modeled by building upon previously developed models.

There are two reasons that this project is interesting. The first is that the resulting tool will be a step in a direction that is useful to people who actually do modeling. The second is that rather than being simply an updated implementation of existing modeling tools, it will form the basis for a system that will exhibit the ability to "understand"

the way the world works and use that understanding to interpret data available from the real world.

5. MISCELLANEOUS PROJECTS

A number of RTS projects were directed at concluding or continuing previously reported research; following subsections briefly note several efforts in this category.

5.1. Data Base Accelerator

Continuing work by Osler and Zippel on the Smart Memory Project has focused on completion of a Content Addressable Memory (CAM) system called the Data Base Accelerator. The goals of the project are to produce a memory system which has extendability, storage capacity, density, and per-bit-costs similar to RAM based memory systems, but with significantly more computing power.

The basic element in this memory system is a 32K-digit chip called the DBA. This chip will be fabricated at MIT's M.T.F., using an enhanced 2μm CMOS process employing titanium silicide polysilicon gate electrodes and N-channel buried contacts. The first step towards that goal is the fabrication of a prototype chip, of 2K-digits. The data path of this chip has been fabricated using MOSIS's 2μm CMOS process. Verification of this data path has occurred, and the design of the associated control logic has been begun. Working 2K prototype chips are expected in the fall.

A circuit board has been designed, fabricated, and debugged. This circuit board has been designed to accept up to 16 of either the 2K or the 32K DBA chips. The circuit board plugs into a Symbolics Lisp machine. When working prototypes become available, this board will be the medium for system testing of the DBA chips.

5.2. NuBus Gadgetry

The RTS undercurrent of NuBus-related projects has continued, including development of bus windows [Wolfe], processor cards [Zak], and sound synthesizers [Tabor]. Of particular interest is a new project by Gartner involving the design and implementation of an "architectural sandbox" device containing an array of software programmable gate array chips. The device incorporates a low-level microprogrammed engine and recently available SRAM-based Gate Arrays to arrive at a highly reconfigurable system which operates at reasonable speeds. A large collection of these gate arrays, totalling about 30,000 usable gates is surrounded and controlled by standard wide-word microcode. Data may enter the array simultaneously from two fast 32 bit SRAM stores. A single 32 bit result may be fed back to the array, saved or transmitted to a remote device through the PORT Transceiver. Various bits of state may be dynamically set or read from the keyboard console of the Lisp Machine. This board interfaces with the Lisp Machine as a slave peripheral through the use of a 10 Mhz Nubus.

Publications

1. Dertouzos, M. L. "Personal Computers." *1986 Edition of World Book Encycolopaedia*, Chicago, IL, 1986

2. Dertouzos, M. L. "Future Information Technology and Some Thoughts on T.I.'s Role." Texas Instruments, Dallas, TX, September 1986.

Talks

1. Dertouzos, M. L. "Multiple Cooperating Computers." Data General Corporation, June 1986.

2. Dertouzos, M. L. "Core IT Issues." MIT Sloan School of Management, Colloquium: Management in the 1990s, Cambridge, MA, July 1986.

3. Dertouzos, M. L. "The Frontiers of Information Technology by 2000 A.D." Harvard University 350th Anniversary Celebration, Cambridge, MA, September 1986.

4. Dertouzos, M. L. "Long Term Trends In Information Technology." Internal Revenue Service, Annual Meeting, Washington, DC, October 1986.

5. Dertouzos, M. L. "Technology Overview." Fondazione Luigi Einaudi, Seminar on Advanced Information Technology: The Europe-Japan-USA confrontation, Milano, Italy, October 1986.

6. Dertouzos, M. L. "Dominant Technological Trends +2000 Versus 1986--What Will Make the Difference." The Society for Information Management, Annual Conference, September 28-October 2, 1986.

7. Ward, S. A. "The L Architecture." Texas Instruments, Austin, TX, July 1986.

8. Ward, S. A. "The L Architecture." Texas Instruments, Dallas, TX, July 1986.

9. Ward, S. A. "The L Architecture." IBM/MIT Technical Review, April 1987.

10. Ward, S. A. "A View From The Outside." Texas Instruments, Dallas, TX, September, 1986.

11. Ward, S. A. "Set Associative Dynamic Random Access Memories." Texas Instruments, Dallas, TX, June 1987.

12. Ward, S. A. "Systems, Semiconductors, Strategies." Intel-Asilomar III Conference, Rippling River, Oregon, June 1987.

13. Ward, S. A. and W. K. Stuart. "A Solution to a Special Case of the Synchronization Problem." *IEEE Transactions on Computers*, to appear.

Theses Completed

1. Anderson, R. "High Performance Raster Graphics Architectures." S.B. and S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

2. Baldwin, R. "Rule Based Analysis of Computer Security, Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

3. Forsythe, C. "A Simplified High-Power Graphics Architecture." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

4. Gartner, K. "DREAM: A Dynamical/Reconfigurable Electrically Alterable Module." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

5. Haleblian, J. "G: An Object-Oriented Graphics Editor for the IBM-PC." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

6. Hunt, J. "Design and Construction of a Board to Facilitate Debugging of 3081/E Emulators." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

7. Ku, S.L. "A High Level Microcode Development System." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

8. Matchett, M. "The L--Project Debugger--Inspector." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

9. Nguyen, J. "Mused: A Music Editor." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

10. Spacek, P. "A Network-Based Real-Time Interactive Simulation." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

11. Wu, C. "Persistent Object Storage for Object-Oriented Database Systems." S.B. & S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

12. Zak, R. "Column Addressable Dynamic Memory: A Novel Design to Increase Microprocessor Performance." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, December 1986.

Theses in Progress

1. Ayers, A. "Architectural Simulation for L." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected August 1987.

2. Chang, S. "A Structure Compiler for L." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected May 1988.

3. Goddeau, D. "A Memory Based Model of Natural Language Acquisition Interpretation, and Generation." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1988.

4. Kartzman, B. "An 8086 Simulator in Lisp -- Can It Perform?" S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1987.

5. Osler, P. "A Prototype Content Addressable Memory System." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected July 1987.

6. Seda, S. "Compacting VLSI Layouts Containing Octagonal Geometry." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected August 1987.

7. Sieber, J. "Gnostic: A System That Can Understand How The World Works." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1987.

8. Wolfe, J. "Explorer Nubus to PC/AT Input/Output Bus Converter Card." S.B. and S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected October 1987.

9. Zhao, F. "An O(n) Algorithm For Three-Dimensional N-Body Simulations." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1987.

SYSTEMATIC PROGRAM DEVELOPMENT

Academic Staff

J. V. Guttag, Group Leader

Research Staff

S. J. Garland

Graduate Students

D. S. Hinman
D. N. Jackson
M. T. Vandevoorde
K. A. Wienhold
K. A. Yelick
J. L. Zachary

Undergraduate Students

J. Lin
B. Teitelbaum

Support Staff

A. L. Rubin

SYSTEMATIC PROGRAM
DEVELOPMENT

1. INTRODUCTION

The activities of the Systematic Program Development Group during 1986-87 have been rather diverse. Most of these activities were part of ongoing work on formal specifications and inference systems. Other activities concerned logic programming and parallel compilation.

2. THE LARCH SPECIFICATION SYSTEM

The main thrust of our research continues to be the development of the Larch Specification System as a vehicle for the specification of software. The Larch Project is a collaborative effort between MIT's Laboratory for Computer Science and DEC's Systems Research Center. The project is developing both a family of specification languages [8] [9] and a set of software tools to support their use. The support tools include language-sensitive editors and semantic checkers based on a powerful theorem prover. Work this year has included language design, tool building, and experimentation with constructing and using specifications.

Larch is a two-tiered approach to formal specifications. One tier is written in a language-independent shared language, the other in a language-dependent interface language. Shared language specifications provide a vocabulary of terms to interface language specifications, which describe the externally visible effects of program components and deal with features (such as parameter mechanisms, exception handling, and concurrency) specific to the programming languages in which systems are implemented.

2.1. Specification of Concurrency

In the last year we have begun studying the specification of concurrent programs. In part, this activity is an outgrowth of work done by John Guttag while he was on leave at DEC's System Research Center. Together with Jim Horning, he developed a formalism and a language for specifying interfaces to concurrent systems. Because concurrency is implemented in different ways on different systems, they added a specific Larch/Concurrent interface language to the Larch family of specification languages. Using this language, they worked with others at DEC to write formal specifications of parts of programs in which concurrency plays a major role.

Experience with these specifications has been very encouraging. A Larch/Concurrent specification of the Topaz synchronization primitives [2] became the reference of choice, both for programmers using those primitives and for programmers responsible for their implementation. A prior prose specification [16] [17] gave an indication of how the primitives were to be used, but left too many questions unanswered about the guaranteed behavior of the interface. A second, semiformal operational specification was more precise and accurate; but it was too subtle, and important information was rather widely distributed. By contrast, both users and implementors seem able to read the Larch/Concurrent specification and to understand its implications.

Another encouraging aspect of our experience is the role played by our specification in insulating clients from the implementation of the synchronization primitives. Not only does the specification abstract from the details of the implementation to provide a simpler model, but it has enabled the underlying implementation to be reworked several times.

2.2. Specification of Text Primitives

Modula-2+ is an extended version of Modula-2 developed at DEC SRC and used for all systems programming there. The compiler provides a number of interfaces that supply frequently used routines. One of these, the text interface, contains routines for manipulating immutable strings. Previously, these routines had been specified informally by short natural language descriptions accompanying the source code. Daniel Jackson wrote formal specifications [11] of these routines using the Larch Shared Language and the Larch/Modula-2+ interface language. He did not specify those routines which, for reasons of efficiency, violate the text data abstraction. The completed formal specifications, with an explanatory introduction and accompanying notes, are now being used by programmers instead of the informal specifications. Reactions have been enthusiastic, and programmers have been happy to help maintain the formal specification. Larch is now more widely used at SRC, and we anticipate that eventually the informal descriptions of all the interfaces will be replaced by formal specifications.

2.3. Specification Tools

We have continued the development of a syntax/semantics-directed specification editor for the Larch Shared Language. Ben Teitelbaum enhanced the editor to take advantage of the features of a bit-mapped display provided by the X Window System. Kay Wienhold, by subjecting the editor to a thorough workout, generated a list of ways in which the editor could provide better support for a specification effort.

2.4. Larch Interface Languages

In his Master's thesis, David Hinman designed a Larch interface language for Modula-2 and provided a guide for the development of further interface languages. Prior to Hinman's work, the only completely defined interface language was for CLU [15] [20]; a partially-defined interface language also existed for Pascal. The existence of a Modula-2 interface language makes the Larch methodology fully accessible to a larger number of programmers.

The Modula-2 interface language demonstrates that interface languages can be designed for programming languages quite different from CLU. It shows how to support programming language features such as multiple parameter passing mechanisms and explicit heap allocation. Furthermore, its definition shows how to formalize the semantics of interface languages in general.

3. INFERENCE SYSTEMS

Our interest in inference systems stems from a desire to provide mechanical assistance for specifiers in detecting errors--assistance analogous to that provided for programmers by compilers [7]. While many errors in programming are easily detected by running a program (that is, by testing), the Larch style of specification emphasizes brevity and clarity at the expense of executability. To compensate for our inability to execute specifications, Larch permits us to make significant assertions about logical properties of specifications; these properties are related to consistency, completeness, and independence.

Our experience in hand checked specifications has convinced us that such assertions are useful in establishing the validity of a specification. Unfortunately, they cannot be checked completely by machine because they are all undecidable in the general case. But we are making progress toward designing useful approximations to the desired checks, and toward building mechanical inference systems that will help us carry out the checks.

3.1. Reve

Much of Larch is based on equational logic because of the promise of an approach to theorem proving based on term rewriting. In conjunction with our work on Larch, we have developed the Reve Term Rewriting Laboratory [5], which is in use in approximately thirty university and industrial laboratories in the United States and Western Europe. This program includes a robust implementation of the Knuth-Bendix [12] completion procedure, partial support for inductionless induction using the Huet-Hullot method [10], and a laboratory environment for rewriting and unifying terms; it can handle associative-commutative operators (using the methods of Peterson and Stickel [19]) and several orderings (rdos [14], dsmpos, and polynomial interpretations) to prove the termination of rewriting systems.

We are about to issue a final release, Reve 2.5, of the Term Rewriting Laboratory. This release differs from the previous one in the following ways.

- Reve 2.5 supports proofs by traditional induction in addition to proofs by inductionless induction. We have had some success (see below) in showing that this mechanism is both more powerful and easier to work with than inductionless induction.

- Reve 2.5 provides an automated test [13] that constructors for proofs by inductionless induction satisfy the required principle of definition (i.e., are sufficiently complete).

- The polynomial interpretation has been extended to use sequences of polynomials, rather than a single polynomial, to interpret an operator [1].

As a result, it is now possible to use a polynomial interpretation to prove the termination of a rewriting system for the natural numbers under addition and multiplication.

- The user interface in Reve 2.5 is more flexible and consistent than before.
- The library of examples that accompanies Reve has been expanded and annotated.
- A half dozen subtle, but significant errors in the previous release have been fixed.

Our evaluation of Reve as a tool for analyzing specifications has convinced us to bring the development of Reve to a close and to begin development of a new inference system, described below, in which the Knuth-Bendix completion procedure plays a less central role.

3.2. Induction

The inductive theory of a set of equations (i.e., the equations true in its initial model) is generally larger than its equational theory (i.e., the equations true in any model); it is also larger than the set of theorems provable using any particular method of induction. Our experience with inductionless induction in Reve gave us cause for concern that this method of induction, though of considerable theoretical interest, presented far too many practical difficulties. Hence we implemented facilities for constructing proofs by traditional induction in Reve, and we used these facilities to contrast the two methods [6].

Our experiments lead us to conclude that term-rewriting systems are well-suited to constructing traditional inductive proofs. For the kinds of examples we are most interested in, traditional induction seems superior to inductionless induction. While the set of provable theorems is not necessarily larger, the proofs are often--though not always--easier to find.

Proofs by traditional induction have a simpler theoretical basis--one which is more familiar to nonspecialists. It is easier to control such proofs, to see when and why they fail, and to supply the lemmas necessary to make them succeed. Proofs by inductionless induction, though intended to reduce the amount of interaction required from the user, actually require more interaction (of a less obvious sort) in many applications.

Proofs by traditional induction impose no special requirements on a set of equations. Inductionless induction requires that a set of equations to be Hilbert-Post complete, a property which is difficult to show and which makes the method inapplicable when we wish to reason about incompletely specified systems. Inductionless induction requires that a set of equations be converted into an equivalent, terminating, confluent set of

rewrite rules. Traditional induction can often succeed when a set of equations is converted into a set of rewrite rules that lacks one or more of these properties.

These experiments persuade us that the interesting theoretical issues raised by inductionless induction have unjustly distracted attention from a more traditional approach to induction. We intend to pursue this more natural approach in our future work.

3.3. The Larch Prover

This spring, we started work on a new theorem proving system, called The Larch Prover, or lp for short. This system is designed to support our work on analyzing specifications of software. It builds on our earlier work on Reve, but departs from Reve in some radical and interesting ways. Whereas Reve was designed to analyze and construct term rewriting systems for proving theorems in equational logic, lp is designed to use these systems in constructing proofs for formulas in quantifier-free first-order logic. This change of focus allows us to prove more theorems; it even seems to reduce the time required to prove equational theorems.

In specific terms, the current version of the Larch Prover differs from Reve 2.5 in the following ways.

- Special mechanisms exist in lp for reasoning about equality, logical connectives, and conditionals. Some of these mechanisms are simply built-in sets of rewrite rules. Others (such as a simplification rule for conditionals) are implemented as metarules. And still others are implemented as transformations on a rewriting system which enlarge its equational theory while respecting the intended interpretation of the logical primitives.

- Proofs in lp can be carried out by case analysis in addition to equational reasoning, traditional induction, and inductionless induction. Case analysis enables us to apply rewriting techniques in the proof of quantifier-free first-order formulas.

- The term-rewriting capabilities of lp have been enhanced to handle operators that are commutative, but not associative.

- The user interface of lp has been streamlined and enhanced to facilitate proof construction, to give users greater control over term rewriting systems, and to facilitate connection to our specification environment.

- The performance of lp has been improved considerably by analyzing its behavior on several large proofs.

Next year, we plan to continue development of lp and to experiment further with its

use. A major goal is to make lp easy to use by nonspecialists. A related theoretical goal is to use more fully the notion of "critical pairs," which arises in the Knuth-Bendix completion procedure for term rewriting systems, to enhance the power of, and to provide a cleaner logical foundation for, inference systems based on term rewriting. Fulfilling these goals should cause lp to outperform, be easier to use, and be conceptually simpler than the Boyer-Moore theorem prover [3] [4], which uses an intricate set of heuristics to control the application of term rewriting to proofs in quantifier-free first-order logic.

3.4. Verification

As an outgrowth of our work on specifying concurrent systems, we became involved in verifying them. In particular we have started to use the Larch Prover to construct mechanical proofs of the correctness of concurrent algorithms. We successfully completed a proof of the correctness of a pipeline architecture, and we have started verifying the correctness of an algorithm developed by Nancy Lynch and Leslie Lamport.

Others have used Reve to verify properties of mathematical systems. For example, Ursula Martin at the University of Manchester has used Reve to investigate axiomatizations for algebraic systems [18].

4. EQUATIONAL LOGIC PROGRAMMING

In his Ph.D. thesis, Joe Zachary investigated ways of using procedural and data abstraction in logic programming. His goal was to carry over techniques for programming in the large that have been developed for use with imperative languages. He presented Denali, a language based upon pure Prolog and equational logic, as a vehicle for illustrating our ideas.

In Denali, predicate abstraction is analogous to procedural abstraction in conventional programming languages. The interface of a predicate abstraction must fix the domain of objects over which the predicate is defined. Because the objects within a logic program can contain variables, completely describing the domain of a predicate requires a two-dimensional type system. Consequently, a Denali type is composed of a conventional sort component and a novel multi-valued mode component. A multi-valued mode specifies the degree to which an object must be instantiated. The modes are an improvement upon the bi-valued modes used by some logic languages.

The objects of a program written in an equational logic language are related by an equational theory. A program in such a language typically consists of a set of Horn clauses and a set of equations that present the theory. In a resolution interpreter for such a language, an equational unification procedure that respects this theory must be substituted for the classical unification algorithm. Unfortunately, obtaining an

equational unification procedure for an arbitrary equational theory is an undecidable problem. Moreover, unification algorithms are known for only a handful of simple theories. Although unification procedures can be synthesized dynamically for a restricted class of theories, the resulting procedures are extremely inefficient and are almost always non-terminating.

By contrast, the equational theory that underlies a Denali program is not explicitly presented. Instead, the programmer must implement, for each sort, a unification predicate for the objects of that sort. The language implementation, in turn, combines these individual predicates into an overall unification procedure. The most critical aspect of the design of the data abstraction mechanism in Denali, then, is facilitating the construction of unification predicates by the programmer. In particular, the design takes care to ensure that Denali programs can be constructed by non-specialists in equational logic.

The implementation of unification is facilitated in two ways. First, Denali supports the layering of data abstractions. Consequently, existing unification algorithms can be built into the language and exploited by user-defined implementations. Second, the mode mechanism can be used by the programmer to impose restrictions upon both the domain and range of a unification predicate. This simplifies the problem of implementing unification, at the cost of some expressive power.

5. PARALLEL COMPILATION

In his SM thesis, Mark Vandevoorde described the extension of a C compiler to run in parallel on a tightly-coupled multiprocessor. The parallel compiler (PTCC) consists of a two-stage pipeline. The first stage performs lexical analysis for the second stage, which does the parsing and assembly code generation. Lexical analysis is extended to include matching paired delimiters so that the parser can quickly advance past units of the source program. The second stage processes units of the source program as fine as a single statement in parallel.

To avoid unproductive parallelism, a new scheduling abstraction, called WorkCrew, is used in PTCC. In the WorkCrew model of computation, a set of workers cooperate to perform tasks specified by the client. It differs from other worker/tasklist models in that clients specify how tasks may be divided. Such tasks are only divided if and when a processor becomes idle. Thus, WorkCrews favor serial execution when parallel execution is unproductive. They also favor coarser grains of parallelism over finer ones.

Several experiments were performed to measure the performance of PTCC. With five processors, PTCC performed 2.5 to 3.3 times better than a similar sequential compiler on source files approximately 1000 lines long. To measure the benefits of different levels of parallelism, multiple versions of PTCC were created. Each exploits additional opportunities for parallelism. Procedure-level parallelism is usually sufficient to keep a five-processor Firefly fully utilized. Statement-level parallelism, however, sometimes

significantly reduces compilation time. Furthermore, it was never found to increase compilation time.

The primary factor limiting PTCC's performance is processor idle time during the serial processing of declarations. Contention for the shared memory slows the processors by roughly 10%. Overhead to allow (but not exploit) parallelism makes PTCC run approximately 7% slower than its serial counterpart.

References

1. Ben Cherifa, A. and P. Lescanne. "Termination of Rewriting Systems and Polynomial Interpretations." Centre Recherche en Informatique de Nancy, France, April 1986.

2. Birrell, A. D., J. Guttag, J. Horning and R. Levin. "Synchronization Primitives for a Multiprocessor: A Formal Specification." *Proceedings of the Symposium on Operating Systems Principles*, Austin, TX, to appear November 1987.

3. Boyer, R. S. and J.S. Moore. A Computational Logic. New York, Academic Press, 1979.

4. Boyer, R. S. and J.S. Moore. "Integrating Decision Procedures into Heuristics Theorem Provers: A Case Study with Linear Arithmetic." Machine Intelligence. Oxford University Press, to appear.

5. Forgaard, R. and J.V. Guttag. "REVE: A Term Rewriting System Generator with Failure-Resistant Knuth-Bendix." *Proceedings of an NSF Workshop on the Rewrite Rule Laboratory*, General Electric Corporate Research and Development Report No. 84GEN008, Schenectady, NY, April 1984, 5-31.

6. Garland, S. J. and J. V. Guttag. "Why Induct Inductionlessly When You Could Induct Inductively?" MIT Laboratory for Computer Science, Cambridge, MA, 1987.

7. Garland, S.J. and J. V. Guttag. "Automated Analysis of Formal Specifications." MIT Laboratory for Computer Science, Cambridge, MA, 1987.

8. Guttag, J. V. and J. J. Horning. "Report on the Larch Shared Language." and "A Larch Shared Language Handbook." *Science of Computer Programming*, 6, 2, (March 1986), 103-157.

9. Guttag, J. V., J.J. Horning and J. M. Wing "An Overview of the Larch Family of Specification Languages." *IEEE Software*, (September 1985), 24-35.

10. Huet, G. and J.M. Hullot. "Proofs by Induction in Equational Theories with Constructors." *Proceedings of the 21st Symposium on the Foundations of Computer Science*, Los Angeles, CA, October 1980, 96-107.

11. Jackson, D. N. and J.J. Horning. "The Modula-2+ Text Interface." Internal Memorandum, Digital Equipment Corporation, Systems Research Center, Palo Alto, CA, October 1986.

12. Knuth, D. E. and P. Bendix. "Simple Word Problems in Universal Algebras." in Computational Problems in Abstract Algebra. J. Leech (ed.), Pergamon Press, Oxford, 1969, 263-297.

13. Lazrek, A., P. Lescanne and J.-J. Thiel. "Proving Inductive Equalities: Algorithms and Implementation." Report 86-R-087, Centre Recherche en Informatique de Nancy, France, September 1986.

14. Lescanne, P. "Uniform Termination of Term Rewriting Systems: Recursive Decomposition Ordering with Status." *Proceedings of the 6th Colloquium on Trees in Algebra and Programming*, Bourdeaux, France, Cambridge University Press, March 1984. Also appears as "How to Prove Termination? An Approach to the Implementation of a New Recursive Decomposition Ordering." *Proceedings of an NSF Workshop on the Rewrite Rule Laboratory*, General Electric Corporate Research and Development Report No. 84GEN008, Schenectady, NY, April 1984, 109-121.

15. Liskov, B. H. and J. V. Guttag. Abstraction and Specification in Program Development. Cambridge, MA, MIT Press, 1986.

16. Rovner, P., R. Levin and J. Wick. "On Extending Modula-2 for Building Large, Integrated Systems." Report 3, Digital Equipment Corporation, Systems Research Center, Palo Alto, CA, 1985.

17. Rovner, P. "Extending Modula-2 to Build Large, Integrated Systems." *IEEE Software*, 3, 6, (November 1986), 46-57.

18. Martin, U. "Doing Algebra with REVE." Technical Report UMCS-86-10-4, Department of Computer Science, University of Manchester, June 1986.

19. Stickel, M. E. "A Unification Algorithm for Associative-Commutative Theories." *Journal of the ACM*, 28, 3, (July 1981), 423-434.

20. Wing, J. M. "A Two-Tiered Approach to Specifying Programs." Ph.D. dissertation MIT/LCS/TR-299, MIT Laboratory for Computer Science, Cambridge, MA, May 1983.

Publications

1. Birrell, A. D., J.V. Guttag, J.J. Horning and R. Levin. "Synchronization Primitives for a Multiprocessor: A Formal Specification." *Proceedings of the Symposium on Operating Systems Principles*, Austin, TX, to appear November 1987.

2. Garland, S. J. and J. V. Guttag. "Why Induct Inductionlessly When You Could Induct Inductively?" MIT Laboratory for Computer Science, Cambridge, MA, 1987.

3. Garland, S.J. and J. V. Guttag. "Automated Analysis of Formal Specifications." MIT Laboratory for Computer Science, Cambridge, MA, 1987.

4. Jackson, D. N. and J.J. Horning "The Modula-2+ Text Interface." Internal Memorandum, Digital Equipment Corporation, Systems Research Center, Palo Alto, CA, October 1986.

5. Yelick, K. A. "Unification in Combinations of Collapse-Free Regular Theories." *Journal Symbolic Computation*, 3, (1987), 153-181.

6. Zachary, J. L. "Multi-valued Modes for Logic Programming Languages." MIT Laboratory for Computer Science, Cambridge, MA, 1987.

Theses Completed

1. Hinman, D. S. "On the Design of Larch Interface Languages." S. M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, January 1987.

2. Vandevoorde, M. T. "Parallel Compilation on a Tightly-Coupled Multiprocessor." S. M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

3. Wienhold, K. A. "Hybrid Task Allocation for a Multi-Processor System." S. M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

Thesis in Progress

1. Zachary, J. L. "Integrating Logic Programming and Data Abstraction." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected July 1987.

Talks

1. Garland, S. J. "Reasoning about Sets of Equations with the Reve Term Rewriting Program." MIT Laboratory for Computer Science, Cambridge, MA, February 1987.

2. Garland, S. J. "Term Rewriting, Induction, and Reve." MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

3. Guttag, J. V. "The Larch Approach to Specification." York University Distinguished Lecture Series, March 1987.

4. Guttag, J. V. "Using Specifications to Achieve Modularity." University of Texas Year of Programming, April 1987.

5. Guttag, J. V. "Modularity in the Larch Shared Language." University of Texas Year of Programming, April 1987.

6. Guttag, J. V. "Concurrent Specifications in Larch." University of Texas Year of Programming, April 1987.

7. Zachary, J. L. "Abstraction Mechanisms for Equational Logic Programming Languages."
   MIT, March 1987;
   Dartmouth College, March 1987;
   University of Washington, March 1987;
   Carnegie-Mellon University, March 1987;
   University of Utah, April 1987.

THEORY OF COMPUTATION

Academic Staff

B. Awerbuch	P. Elias	S. Goldwasser	L. Heath
F.T. Leighton	C. Leiserson	N. Lynch	A. Meyer
S. Micali	R. Rivest, Leader	D. Shmoys	M. Sipser

Visitors and Post-Docs

Paul Beame	H. Bodlaender	Ravi Bopanna	B. Chor
Johan Hastad	O. Goldreich	A. Sherman	Kurt Sieber
B. Trakhtenbrot	M. Wand		

Graduate Students

W. Aiello	R. Ashcroft	M. Bellare	B. Berger
B. Bloom	V. Breazu-Tannen	T. Cormen	C. Crepeau
A. Dhagat	P. Feldman	L. Fortnow	J. Fried
A. Goldberg	S. Goldman	R. Greenberg	M. Grigni
R. Hirschfeld	A. Ishii	B. Kaliski	J. Kilian
S. Kipnis	P. Klein	R. Koch	B. Maggs
F.M. Maley	S. Malitz	S. Mentzer	M. Newman
X. Pan	J. Park	C. Phillips	S. Plotkin
S. Rao	M. Reinhold	J. Riecke	R. Schapire
E. Schwabe	D. Short	J. Siskind	R. Sloan
J. Wein	S.-M. Wu		

Undergraduate Students

T. Heigham	T. Leung	B. Rogoff	J. Tang	P. Wang

Support Staff

A. Benford S. Bemus B. Hubbard L. Melcher

1. INTRODUCTION

The principal **research areas** investigated by members of the Theory of Computation Group are:

- algorithms: combinatorial, geometric, graph-theoretic, number theoretic
- cryptology,
- computational complexity,
- distributed computation: algorithms and semantics,
- machine learning,
- randomness in computation,
- semantics and logic of programs,
- VLSI design theory.

Group members were responsible for over one hundred publications and several dozen public lectures around the world during the past year. The *individual reports* by faculty and students in the next sections, and the *annotated reference and lecture lists* offer further descriptions of the year's activities.

The following **major research contributions** merit highlighting:

- Awerbuch's "compiler" which produces distributed network protocols for unreliable networks from protocols which work only on reliable networks [7].
- Breazu-Tannen and Meyer's conservative extension results for polymorphic types [35][36].
- Leighton's results on stable "backoff" protocols for N-station ethernets [80].
- Rivest's general purpose procedure for inferring finite state environments [124].

Group members received the following **awards**:

- Goldwasser and Shmoys received Presidential Young Investigator awards from the NSF.
- Hastad won the ACM Ph.D. thesis award.

- Cormen and Leiserson's paper [46] on their hyperconcentrator switch (patent pending) won the Best Presentation Award at the 1986 International Conference on Parallel Processing.

- Maggs and Leiserson's paper on the distributed random-access machine (DRAM) for parallel computing, which abstracts essential properties of fat-trees won the Most Original Paper Award at the 1986 International Conference on Parallel Processing (cf. [103]).

- Meyer was elected to membership in the American Academy of Arts and Sciences, May '87.

2. FACULTY REPORTS

Baruch Awerbuch

Awerbuch has been working on designing efficient and reliable distributed protocols. His main emphasis was on studying issues related to complexity of protocols in distributed dynamic systems.

Together with S. Even (Duke U.), he investigated reliable broadcast protocols in unreliable networks. He showed [10] that reliable broadcast is possible in networks under very weak connectivity requirements; such networks may never be connected as a whole.

Together with Micali, he discovered [13] a new protocol for detecting and resolving deadlocks, which can deal with the most general form of deadlocks. Its requirements in communication, time, and memory are optimal.

Another problem is making an arbitrary protocol adaptive to dynamic input and network topology, without using unbounded counters or time stamps. The most general solution to that problem is a "compiler" which receives as input an arbitrary protocol and produces as output an adaptive version of the former protocol. Such a compiler is called an *Adaptor*. Designing efficient adaptors is one of the oldest and the most fundamental problems in communication networking. It has been open since 1976. Moreover, there was no meaningful definition of what constitutes a "truly" bounded-complexity adaptor. Awerbuch defined a "truly-bounded" adaptor as one whose overhead is bounded by the overhead of the original non-adaptive protocol, which constitutes the true input to the problem. That is Awerbuch succeeded in developing such an adaptor. It also has practical value since it is very simple and easy to program; its on one page.

In [8],[2], Awerbuch raised a fundamental question, neglected so far in the literature: how to make a distributed algorithm robust against input errors and wrong probabilistic assumptions about the distribution of the inputs or link delays. He

introduced a notion of complexity-preserving protocol controller: this is an automatic procedure that controls worst-case execution of any distributed algorithm. He suggested a controller with poly-logarithmic overhead, and then generalized it to solve the more general problem of dynamic resource management.

Together with Plotkin, Awerbuch worked on the problem of Leader Election in a distributed asynchronous faulty network. They have developed a technique to approximate up to a constant factor the size of a dynamically growing asynchronous distributed network with amortized message complexity of $O(\log^2|V|)$ per node, where $|V|$ is the final size of the network. They showed how to apply this technique to construct an efficient distributed algorithm for Leader Election in a faulty network [14]. The algorithm has a message complexity $O(|E| + |V| \log^2 |V|)$, which is an improvement of $O(\log^2 |V|)$ over the previously known algorithms.

Another area of Awerbuch's research was distributed graph algorithms.

Together with R. Gallager (MIT), Awerbuch found a new distributed algorithm to find Breadth First Search Trees in an asynchronous communication network [14]. The communication complexity of this algorithm is $O(E + V^{1.6})$ messages, which is optimal for dense networks.

In [9], Awerbuch discovered optimal algorithms for a number of problems, including Minimum Spanning Tree, Leader Election, Counting, and other problems in distributed computing. Each one of those algorithms has $\Theta(E + V \log V)$ communication complexity and $\Theta(V)$ time complexity. Previous algorithms required $\Theta(V \log^* V)$ time.

Together with Goldreich and Vainish, [12], Awerbuch showed that at least $\Omega(E)$ messages are needed to construct a spanning tree in a network, even if the identity of every neighbor is known.

Awerbuch also worked on the problem of breaking symmetry in a network. He showed that it is impossible to find a maximal independent set on a ring or on a line in less than $\Omega(\log^* n)$ time; this lower bound is tight. Also, working with Beame, he has extended this result to several infinite families of graphs for which known symmetry-breaking algorithms are optimal.

Another area of Awerbuch's research was protocols for shared memory registers. Together with P. Vitanyi, [129] Awerbuch investigated the problem of constructing multi-user atomic shared registers. The problem addressed is rooted in hardware design of concurrent registers access by asynchronous components and also in asynchronous interprocess communication. He succeeded in constructing multi-valued registers which can be read and written asynchronously by many processors in a consistent fashion.

Awerbuch plans to continue research on practical distributed algorithms and communication protocols.

Peter Elias

The research on two universal coding schemes mentioned in the last annual progress report was published in January 1987 [54]. More recent work was suggested by a question from Leonid Levin about the capacity of a binary channel under jamming, and is being prepared for publication.

In a binary jamming channel (BJC) with parameter p a codebook assigns $M=2^{NR}$ messages each to an N-bit codeword. The transmitter selects one for transmission. A jammer changes at most Np of the N bits. The jammer knows the codebook, the transmitter's selection and the (fixed) rule by which the receiver assigns messages to each of the possible N-bit received noisy sequences. If the receiver assigns a single message to each sequence, then arbitrarily reliable bits can be received over the channel at a positive rate R for all sufficiently large N, despite the jammer, only if $p<1/4$. The largest such R as a function of p -- the *capacity* of the BJC -- is not known precisely, but is bounded above and below by functions which are both strictly positive for $0\leq p<1/4$, and both vanish for $1/4<p<1/2$: the jammer can guarantee incorrect decoding of almost all messages for any positive R when $p>1/4$.

The new results apply list decoding, useful in analyzing noisy channels and feedback [52][53][130][58], to the BJC. In list-of-L decoding the transmitter selects a message from a set of size $M=L2^{NR}$ and sends the corresponding codeword. The jammer still changes up to Np bits. The receiver now maps the received sequence into a list of L messages: there is a decoding error if the transmitted message is not on that list. If decoding is successful, $log(M/L)=NR$ bits have been transmitted. One result which follows quite simply from earlier work [52][131] but may be new is that the capacity of the BJC under list decoding is $C_1(p)=1-H(p)$, where $H(p)$ is the binary entropy function and $C_1(p)>0$ for $p<1/2$. For every R less than $C_1(p)$ there is an L so large that for all sufficiently large N codes of length N and rate R exist for which the transmitted message will be on the receiver's list of L whenever the jammer has changed no more than Np bits. That result is not constructive. Another result is the construction of a family of linear codes which allow reliable communication at any rate $R<C_2(p)$ using a list size $L(p,R)$, where $0<C_2(p)<C_1(p)$ for all $p<1/2$.

Shafi Goldwasser

On leave Spring '87.

Lenwood S. Heath

Heath has been working on problems related to fault-tolerance in processor networks. In particular, he has made progress on two different aspects of Rosenberg's DIOGENES design methodology [126]. First, Heath developed algorithms and NP-completeness results concerning the implementation of networks in linear arrangements [83]. Second, he and Istrail disproved a conjecture concerning genus g graphs by developing an efficient algorithm for embedding the class of genus g graphs in a book having a bounded number of pages [83].

Heath also investigated problems of mapping one network structure into another, which can be seen as mapping an algorithm into a parallel processor network. He and Rosenberg have obtained an efficient algorithm for mapping the FFT algorithm into the Hypercube network with optimal dilation and expansion [82].

Heath will continue to work on fault-tolerance and network mapping problems.

Tom Leighton

Most of Leighton's research time during the past year was devoted to problems relating to the design and analysis of parallel algorithms and architectures. In particular, he spent a substantial amount of time studying the computational properties of the binary hypercube, currently one of the most popular architectural choices for large scale parallel machines. Working with Sandeep Bhatt (Yale), Fan Chung (Bell Core), Hastad, Heath, Newman, Arnie Rosenberg (UMass-Amherst) and others, they showed that the hypercube is a far more versatile and universal structure than previously realized. For example, they proved that the N-node hypercube can simulate (with only a small constant slowdown) any N-node binary tree, mesh of trees, grid graph, multigrid, pyramid network, butterfly or cube-connected-cycles. Moreover, these simulations can still be performed even if up to 50% of the cells of the hypercube fail at random! The details of this work can be found in a recent STOC paper with Hastad and Newman [79], a FOCS paper with Bhatt, Chung and Rosenberg [18], and a forthcoming paper with Heath and Rosenberg.

Leighton also continued working on problems related to the average case analysis of algorithms. The highlight of research in this area during the past year is the work with Hastad and Rogoff on analysis of backoff protocols for communication in multiple access channels such as Ethernet [80]. Ethernets currently use a protocol known as exponential backoff to govern the transmission of broadcasts. In exponential backoff, a station attempts to transmit with probability 2^{-b} where b is the number of times the station has tried to send but failed since the last successful transmission. The exponential backoff protocol seems to work well enough for low densities of message traffic, but does not perform well for high density situations. In their work, they formally prove that exponential backoff is unstable if the arrival rate of messages exceeds a certain threshold. On the other hand (and more importantly), they show that polynomial backoff protocols (such as quadratic backoff) are stable up to nearly 100% utilization, the best possible. Hence, quadratic backoff is dramatically superior to exponential backoff in systems with lots of message traffic. The details of this work can be found in the latest ACM STOC proceedings.

Aside from research, Leighton spent a great deal of time working on service-related activities such as chairing the program committee for the next IEEE FOCS conference, and on writing his book. Unfortunately, Leighton did not progress as much as he had hoped on the book, and it now looks like the first volume won't get done until next summer. The first volume covers parallel algorithms and architectures. The second

(and last) volume is on VLSI computation and design. Considering that it has taken a year and a half to write the first 300 pages, it may take a long time before Volume II is finished!

Charles E. Leiserson

Leiserson has continued his research on the theory of computing machinery. His work has centered on understanding the impact of communication constraints in parallel algorithms and architectures. His earlier work on volume- and area-universal networks, such as fat-trees, has been extended with the help of his students.

Details of Leiserson's research with students can be found in their individual progress reports. A brief summary follows:

- T. H. Cormen: A U. S. patent application [45] has been submitted for the hyperconcentrator switch they designed. A paper [46] on the switch won the Best Presentation Award at the 1986 International Conference on Parallel Processing.

- A. T. Ishii: A rigorous understanding of the timing of level-clocked circuits is under way. They have discovered fast, graph-theoretic algorithms for determining whether a circuit operates properly. The correctness of these algorithms rests on sufficient conditions for proper operation of a circuit, which can be shown to be sufficient if the function of the circuit is interpreted syntactically.

- J. Kilian and S. Kipnis: They have investigated the power of bussed interconnection schemes for realizing permutations among chips and have established a correspondence between permutation architectures and difference covers for permutation sets [94]. As an example of the practical implications of this theoretical work, a fast, efficient cyclic shifter on n chips can be built with only $\sqrt{n}$ pins per chip such that any of the shifts can be realized in only one clock cycle.

- B. M. Maggs: The distributed random-access machine (DRAM) for parallel computing, which abstracts essential properties of fat-trees, has been substantially refined [103]. The earlier version of the paper, implementing efficient algorithms in the model, won the Most Original Paper Award at the 1986 International Conference on Parallel Processing.

- J. K. Park: A deterministic on-line message-routing algorithm for fat-trees has been discovered that works asymptotically as well as previous probabilistic algorithms. The algorithm also works on a butterfly variant of fat-trees that uses constant-sized switches.

- C. A. Phillips: They have discovered an algorithm for contracting bounded-degree planar graphs [104]. The algorithm leads to fast EREW-PRAM and DRAM algorithms for the problem of region labeling in vision systems.

In other work, Leiserson has completed a paper with Leighton surveying algorithms for integrating wafer-scale systolic arrays [99]. He is also writing a textbook on algorithms with Cormen and Rivest, tentatively entitled, *Introduction to Algorithms*.

Theses that were completed under Leiserson's supervision include:

- Thomas H. Cormen, *Concentrator Switches for Routing Messages in Parallel Computers*, M. S., August 1986.
- Bruce M. Maggs, *Communication-Efficient Parallel Graph Algorithms*, M. S., August 1986.
- Daniel Weise, *Hierarchical, Multilevel Verification of MOS/VLSI Circuits*, Ph.D., August 1986 (reader).
- Alan T. Sherman, *Cryptology and VLSI*. Ph.D., October 1986 (reader).
- Marie J. Sullivan, *Parallel Graph Algorithms on the Connection Machine*, B. S., February 1987.
- Andrew V. Goldberg, *Efficient Graph Algorithms for Sequential and Parallel Computers*, Ph.D., February 1987.

Leiserson is a Presidential Young Investigator, an editor of the *Journal of Parallel and Distributed Systems*, and a Corporate Fellow of Thinking Machines Corporation, Cambridge, Mass.

Nancy A. Lynch

The main project Lynch worked on this year was the development of a theory for nested transactions, a joint project with Dr. Michael Merritt (AT&T Bell Laboratories). The concept of nested transactions is central to new languages and systems for distributed computing. The theory allows careful description of the semantics of nested transactions, as well as description and correctness proofs for algorithms which implement them. The work began with a paper [107] last summer, in which they presented a semantic model and stated the basic correctness conditions to be satisfied by nested transaction systems. In this paper, they presented and proved correctness of an exclusive locking algorithm for nested transactions.

With colleagues at MIT and elsewhere, they continued by treating an interesting variety of algorithms. With Dr. Maurice Herlihy (CMU) and Bill Weihl (MIT), they presented proofs for two algorithms for management of "orphan" transactions [84].

With Alan Fekete (MIT) and Weihl, they extended the work in [107] to read-write locking [55] and are now working on a further extension to a new algorithm for general commutativity-based locking. Lynch and K. Goldman (MIT) presented a proof for a replicated data-management algorithm [67]. Lynch has been helping Jim Aspnes (MIT) to carry out a similar treatment of timestamp-based algorithms [6]. The model has provided a vocabulary for describing an amazing variety of interesting algorithms, and has permitted proofs which follow the natural modularity which system designers use in talking about the algorithms informally.

At bottom, all of this work is based on the "I/O automaton" model of concurrent computation, a new model which emphasizes the distinction between input and output events. This model was developed by Lynch and Mark Tuttle (MIT) [108]. In addition to using it to model concurrency control algorithms, Lynch and Tuttle used it to carry out an interesting hierarchical proof of a distributed network resource-allocation algorithm. Lynch and her students are continuing to develop this model and to apply it to many other problems in concurrency. For instance, she helped Bloom to describe his new algorithm for implementing two-writer atomic registers from one-writer atomic registers, by using I/O automata as a framework for presenting all the definitions, algorithms and proofs [19]. She is helping Jennifer Welch (MIT) to use this model to present algorithms for mutual exclusion and Dining and Drinking Philosophers. She has worked with Fekete and Dr. Liuba Shrira (MIT) to present a modular decomposition of a network synchronizer algorithm of Awerbuch [56].

She has also been working with Dr. Leslie Lamport (DEC West) on a new technique for carrying out lattice-structured correctness proofs for distributed network algorithms. They have been using it for reasoning about algorithms for minimum spanning tree computation and for bank audit.

She plans to continue the work on nested transaction theory, continue developing the I/O automaton model, and continue trying to apply this model to study a variety of application areas.

Albert R. Meyer

Meyer's research continues to focus on semantics and logic of programming languages. The various topics under investigation are enumerated below. Further technical information is in the annotated references and reports of collaborators.

Research

- *Empty types.* The question of whether simulations of Booleans and integers by constructs in a pure "polymorphic" calculus, which Meyer posed to his e-mail list on types in programming ([111]), led to a remarkable theoretical development of proof and model theory of polymorphic types [37][114][33][35]. Unexpectedly, and in contrast to non-polymorphic type disciplines, the decision to allow types to be *empty* has crucial technical consequences.

- *Conservative extension.* Polymorphism is orthogonal to other programming constructs, that is, the logic of other constructs is unaffected by embedding them in a polymorphic context. This is formalized in a Ph.D. thesis just completed under Meyer's supervision [36].

- *Hoare-style partial correctness logic* for ALGOL-like languages. joint with Sieber and German (GTE). See Sieber's report.

- *Semantics of concurrency* with Bloom and Istrail (Wesleyan). Questions the foundations of Hoare's CSP and Milner's CCS theories of concurrency. See Bloom's report.

- *Fundamental theorem of Scott domains* with S. Cosmadakis (IBM Watson Res.Ctr.) Explains the general connection between operational and denotational semantics.

- *Monotone models* with D. Vellemin (Amherst College) and G. Plotkin (Edinburgh). Demonstration that *continuity is not needed* for semantics of first order functional languages---the most familiar case. Allows a simpler pedagogical treatment introducing the theory and incidentally implies incompleteness of usual LCF style reasoning at third-order types where continuity *is* needed.

- *Lambda calculus and Cartesian closed categories.* Further work with Breazu-Tannen following [34].

- *Continuations.* Fundamental theory of continuation transforms of simply typed functional expressions. Joint with Riecke following earlier work with Wand (Northeastern) [116].

Professional Activities

- Moderator for research e-mail lists on

  1) Types in Programming,

  2) Semantics of Concurrency,

  3) Logic of Programs.

- These lists are coming to play a significant role in Meyer's research activities. Most of the empty types and conservative extension results grew out of e-mail collaborations involving researchers in England, France, Italy, and U.S.

- Editor(-in-Chief) of *Information and Computation* (formerly Information and Control).
- One of five Editors of a thousand-plus page Handbook of Theoretical Computer Science to be published in 1988 by North-Holland.
- Editor of *JCSS, SICOMP, TCS, Advances in Applied Math.*
- Student Supervision:
  - *Ph.D.*: Breazu-Tannen, Jan. 87 [32]; Bloom, expected June '88.
  - *Ph.D. Reader*: D.A. MacAllester, June '87; J. Lucassen, expected Sept. '87.
  - *S.M.*: Reinhold, expected Sept. '87; Riecke, expected June, '88.
- Developing an undergrad theory course in semantics and logic of programming (6.044)--an alternative to the usual required course in computability and complexity (6.045).
- Elected to membership in American Academy of Arts and Sciences, May '87.

Silvio Micali

Micali has been working on determining the power of theorem-proving procedures and, more generally, determining what theorems can be proved in any other efficient way.

Micali has continued work on his particular interest in a new "compiler-type" algorithm that seems fundamental in the design of cryptographic protocols with more than two participants. This algorithm, given the specification of *any* protocol for totally honest parties (which is usually trivial to design), outputs an alternative and equivalent protocol that *remains correct* even if up to *half* of its participants deviate from their prescribed programs in an *arbitrary* (but polynomial-time bounded) way. Micali has been working on extending his "compiler" to handle the case of two-party protocols.

He has been working on developing a theory in the field of asynchronous distributed systems when all participants are reliable. Here the problem is the arbitrary arrival time of messages. The only guarantee is that all messages will be eventually all delivered. However, messages sent first may arrive later and there is no upper bound on the time for a message to reach its recipient. Clearly, finding robust good solutions in this extreme model will, *a fortiori*, imply having good solutions for more reasonable models.

Together with Beame, he has been investigating the problem of quantifying the amount of knowledge which a randomized interactive protocol releases. They have obtained a definition which appears to satisfy intuitive notions of the amount of usable information released and which has some nice basic information-theoretic properties.

Micali plans to work on signature schemes together with Dhagat this summer.

Ronald L. Rivest

Rivest has developed a new method for searching game trees [121]. This method, an alternative to minimax search with alpha-beta pruning, uses p-th arithmetic means as an approximation to the *min* and *max* functions, and then takes derivatives to identify the unexpanded leaf upon whose value the value at the root most strongly depends. This leaf is then chosen be expanded next. Extensive empirical results demonstrate that the new method can outperform traditional alpha-beta pruning.

Rivest has also developed a new method for controlling transmissions in a slotted multi-access broadcast channel [121] with ternary feedback (collision, successful broadcast, or hole). This method, called "Pseudo-Bayesian Broadcast", is an approximation of the ideal Bayesian estimators for the number N of stations wishing to transmit a packet. Each station keeps an estimate v of N, and updates v after each time slot as follows:

- If there was a hole or a successful broadcast, v is decreased by one, otherwise (if there was a collision) v is increased by $\frac{1}{e-2}=1.392211....$ v is increased by $\frac{1}{e}$ or any other available upper bound on the average arrival rate of new packets into the system. If v is less than one, it is replaced by one.

Then, any station with a packet to transmit, transmits during the next time slot with probability $\frac{1}{v}$. (Note that newly arrived packets and backlogged packets are treated identically.) It is possible to show that this rule is stable for any average arrival rate less than $\frac{1}{e}$. Empirical results demonstrate that this rule provides excellent performance.

Rivest has shown how one can learn "decision-lists" efficiently, within the model of learning proposed by Valiant. A decision-list is a representation of a Boolean formula, which classifies given sample data points as positive or negative instances of the concept being learned. A decision-list is a sequence of *rules*, each of which specifies a *term* (i.e. a conjunction of some literals) and a *value* (either *true* or *false*). The last rule represents a default rule; its term is *true*. A data point is classified as a positive (*true*) or negative (*false*) instance by finding the first rule in the list which applies to that data point. The class of decision-lists with terms of size at most k is called k-DL; it is shown that k-DL generalizes the usual classes of k-CNF and k-DNF. It is shown that k-DL is polynomially learnable in the sense of Valiant.

Rivest, together with Leighton, has investigated the accuracy with which a finite-state device can measure a probability [100]. More specifically, the device has access to a coin with bias p in favor of heads, and can make state transitions which depend on whether the coin comes up heads or tails. It is shown how to estimate p to within $O(1/n)$, where n is the number of states, and that this is optimal (to within a constant factor).

Rivest, together with Goldwasser and Micali, has refined and published their signature scheme which is provably secure against chosen-message attacks [73].

Rivest, in collaboration with Goldman, has investigated efficient algorithms for computing the probability distribution of maximum entropy, subject to a number of given constraints [69][70]. The problem is approached by viewing the coordinates of the underlying space as vertices in a hypergraph, and viewing the constraints as hyperedges. It is shown that *adding* hyperedges to the graph can make the hypergraph acyclic; and that acyclic hypergraphs admit efficient algorithms for computing the maximum entropy probability distribution.

Together with Schapire, Rivest has developed new techniques for inferring the structure of a finite-state automaton by systematic experimentation [124]. Their procedures make use of a new representation for finite automata which is dramatically more compact than the usual state-transition table when the automaton exhibits lots of "regularity". Experimental results have been quite encouraging; for example, their procedure can infer the structure of Rubik's Cube in under two minutes of CPU time. (Rubik's Cube can be easily transformed into a finite automaton with over 10^{19} states.)

Sloan and Rivest have developed a new model for inductive inference, based on a Bayesian model with a countable number of hypotheses, but where there are explicit costs associated with making a prediction and with running an experiment [125]. This model can be used to explore a number of tradeoffs in doing science "efficiently", such as when to run a "crucial experiment" (i.e. finding an experiment which will distinguish between the best two competing theories). This model can also be used as a formal model for a theory of subjective probability, and as such can handle a number of problems not well addressed by other approaches.

Rivest, in collaboration with Prof. Ross Quinlan (New South Wales Institute of Technology, Australia), has explored the application of Rissanen's "Minimum Description Length Principle" to the construction of decision trees [118]. They develop a specific algorithm based on this principle, and compare it against other well-known decision-tree construction algorithms on a variety of "real-life" data bases. The results are quite promising; the trees produced are competitive with those produced by the other algorithms both in terms of size and accuracy on new data.

David B. Shmoys

Shmoys studied a wide range of questions in the design and analysis of efficient algorithms. The problems considered are from the areas of parallel and distributed

computation, as well as from the area of worst-case analysis of approximation algorithms for NP-hard combinatorial optimization problems.

Together with Hochbaum (UC/Berkeley), Shmoys focused on obtaining techniques for approximation algorithms that can be used for a wide range of problems [86][88]. The problem of minimizing the maximum completion time of a set of independent jobs when scheduled on a set of identical machines was the initial problem for which worst-case analysis was first proposed. In [88] a polynomial approximation scheme is given for it, which answers a question that had been open for twenty years.

In [85][106], further improvements are given for two of the most important generalizations of the problem considered in [88]. In work done with Hochbaum [85], a polynomial approximation scheme is given for the case that allows machines to run at different speeds, but requires these differences to be uniform for all jobs. With Lenstra (CWI) and Tardos (Eotvos University) [106], an efficient algorithm is given for the next natural generalization, where the machines are entirely unrelated, so that running times for the jobs can vary arbitrarily from machine to machine. The algorithm delivers solutions no worse than twice the optimum, whereas no bounded guarantees for polynomial-time algorithms had been previously known. Furthermore, there is evidence to suggest that this algorithm may be extremely effective in practice.

Shmoys has also been active in the design of protocols for distributed systems, especially for important primitives, such as tossing a coin or reaching agreement on a value in the presence of failures. In [51], that is joint work with Dwork and Stockmeyer (IBM Almaden), procedures are given for tossing a common coin in constant expected time that can tolerate $O(n/\log n)$ failures in a complete network of n processors. These procedures can then be used to build protocols for Byzantine agreement. If constant expected time is required, then the procedure can tolerate $O(n/\log n)$ failures, whereas if a constant fraction of the processors could possibly fail, the protocol takes $O(\log\log n)$ time.

The final area in which Shmoys has been doing work, is in the design of parallel algorithms for combinatorial problems. In work done jointly with Karloff (Univ. Chicago), edge coloring problems were considered [93]. In work not yet submitted for publication, parallel approximation algorithms for the traveling salesman and maximum flow problems were considered. In the former case, an *RNC* algorithm asymptotically approaching the Christofides method is given. In the latter case, an *RNC* approximation scheme is given for this P-complete problem.

Michael Sipser

Sipser worked on developing new methods for proving the inherent computational complexity of a variety of problems in different settings. This past fall, Sipser conducted a seminar to survey his latest results and discussed further directions.

3. STUDENT REPORTS

William Aiello

Aiello has been studying the complexity of interactive proofs and zero-knowledge interactive proofs. In joint work with Goldwasser and Hastad [3], Aiello gives evidence that interactive proofs with an unbounded number of rounds recognize more languages than interactive proofs with a constant number of rounds. It is shown that for any unbounded function F and any function $G=o(F)$ there is an oracle A such that the class of languages recognized by interactive proofs with oracle A and $F(|x|)$ rounds is not contained in the class of languages recognized by interactive proofs with oracle A and $G(|x|)$ rounds. However, when one requires an interactive proof to be perfect zero-knowledge then an unbounded number of rounds do not appear to be much help. Aiello and Hastad [4] show that any language recognized by an unbounded round perfect zero-knowledge proof can also be recognized by a two round interactive proof (which is not necessarily zero-knowledge).

R.A. Ashcroft

Ashcroft arrived at MIT in January after a semester at UC Berkeley. His major accomplishment has been to get a summer internship doing science journalism for *The Economist* in London. This position was obtained largely on the strength of an article on Zero-Knowledge written for the non-computer scientist; this appeared in the June issue of *The Economist*[5].

Paul W. Beame

Beame has been working on methods for proving lower bounds on the resources needed by parallel computers in order to solve various computational problems. With Hastad he has shown [15] that the concurrent-read concurrent-write parallel random access machine (CRCW PRAM) requires Θ (log n/loglog n) time to compute the parity of, sort, or add n-bits as well as to multiply or divide two n-bit integers along with a number of related problems. In addition, Beame and Hastad have demonstrated a time hierarchy for such machines as well given stronger lower bounds which apply to almost all Boolean functions of n bits. Beame has since extended the results in [15] using modified techniques which yield lower bounds for additional graph-theoretic problems, such as finding small cliques in graphs.

Beame has improved the running time of the best known parallel machine algorithms for symmetry-breaking in constant-degree graphs to $O(\text{loglog}^* n)$. Also, with Awerbuch, he has shown that in distributed computation there are several infinite families of graphs for which known symmetry-breaking algorithms are optimal.

Beame and Micali have been investigating the problem of quantifying the amount of knowledge which a randomized interactive protocol releases. They have obtained a definition which appears to satisfy intuitive notions of the amount of usable information released and which has some nice basic information-theoretic properties.

Beame plans to continue work on lower bounds for CRCW PRAM's as well as to investigate the computational advantages of concurrent-read memory access for parallel computers. In September 1987, he will join the faculty of the University of Washington in the Department of Computer Science.

Bard Bloom

Concurrent Process Semantics: Bloom, working with Meyer and Sorin Istrail (Wesleyan) is studying the denotational semantics of parallel and nondeterministic processes. Dana Scott's very successful models for the semantics of sequential, deterministic programs do not extend naturally to the more general domain. There are a number of proposals for a replacement; Meyer and Bloom are investigating several of these models. One question in semantics is, "when shall we consider two programs equivalent?" Two proposed notions are *trace congruence* (used in Hoare's language CSP and variants) and *bisimulation* (used in Milner's SCCS). Bloom, Meyer, and Istrail have found an extension of SCCS (without recursion) in which the two notions coincide. The new operation is somewhat peculiar in nature; they have shown that no finite set of operators defined in a clean way can cause the two to coincide.

Meyer and Bloom are also investigating the complexity of the operational semantics of Milner's SCCS. They have shown that bisimulation in SCCS with unguarded recursion is not arithmetical, and trace congruence is Π_3^0; therefore the two differ for that language and any extension as well.

Fully Abstract Lattice Semantics: A classic paper in denotational semantics (Gordon Plotkin's *LCF Considered as a Programming Language*) gives two kinds of semantics for a simple but extremely powerful language based on typed lambda calculus. One semantics is *operational*, describing how a particular interpreter computes; the other kind is *denotational*, assigning meaning to the programs in moderately familiar mathematical terms, using several varieties of Scott domains. The paper shows that the two semantics coincide in a weak sense (*computational adequacy*: two integer terms evaluate to the same constant if they have the same denotational meaning), but not in a stronger sense (*full abstraction*: two routines behave identically in all contexts if they have the same denotational meaning). The programming language can be extended by the addition of a "parallel conditional" such that the extended language is fully abstract for one of the denotational models. The classic paper shows that this extension is not fully abstract for the other languages.

However, one of the other denotational models (Scott domains built from complete lattices rather than cpo's) is mathematically appealing, and it is somewhat surprising that the classic paper did not find a fully abstract extension of LCF using this model. However, this is not the author's oversight. Bloom has shown that there is *no* fully abstract extension of LCF with a reasonable evaluator for which this model is fully abstract, where "reasonable" means that an arithmetic expression can evaluate to, at most, one value. If the evaluator is not required to be reasonable in this sense, there is

a simple extension of LCF after the spirit of the classic paper which is fully abstract for the lattice model.

Atomic Read/Write Registers: Bloom has shown how to construct a two-writer, n-reader atomic memory register from two one-writer, $(n+1)$-reader atomic memory registers. This work was suggested by N. Lynch (MIT) and is part of a continuing program started by Leslie Lamport (DecCirc) investigating the relations between various models of shared memory. The most useful kind of shared memory is *atomic* memory, defined as memory to which all references appear sequential even though they may overlap arbitrarily. Most previous research was devoted to single-writer simulations, e.g., showing how to simulate one-writer, n-reader atomic memory using roughly n one-writer, one-reader atomic memory cells. This paper describes an optimal construction of a two-writer, n-reader atomic register from two one-writer, $(n+1)$-reader registers.

Mihir Bellare

Bellare arrived at MIT in February after a semester at UC Berkeley. He is working with Rivest on the complexity of a learning algorithm. This work has focused on determining lower bounds to the complexity of an algorithm trying to learn an equivalence relation.

Bonnie Berger

Berger is working with Goldwasser on cryptographic applications for Smart Cards. A Smart Card is a plastic credit card with one or two chips embedded in it to provide it with computing and memory capabilities.

Hans L. Bodlaender

Bodlaender studied the complexity of (NP-complete) graph problems, when restricted to certain classes of graphs. In [22] a general approach is taken. It is shown that there are large classes of problems, called ECC and LCC, that are solvable in polynomial time for graphs with a constant upperbound on the treewidth (and, for problems in LCC, on the degree). Several important classes of graphs can be shown to have a constant upperbound on the treewidth. A large number of important NP-complete graph problems is shown to be in LCC and/or ECC. Also, polynomial algorithms are obtained for the CHROMATIC INDEX problem and the GRAPH ISOMORPHISM problem on graphs with a constant upperbound on the treewidth.

The complexity of these problems in a parallel context has also been studied. It can be shown that the algorithms can be carried out in $O(\log\ n)$ and $O(\log^2\ n)$ time on a EREW PRAM, with a polynomial number of processors.

A similar framework is being developed, with similar results for the class of the cographs.

Bodlaender also is working on the problem of distributed leader finding in rings of

processors. For this problem several new lowerbounds are obtained, that are better than existing ones.

Ravi B. Boppana

Boppana has been working on the design of average-case algorithms for certain NP-complete problems. In particular, he investigated the Graph Bisection problem: Given a graph, partition its vertices into two equal-size pieces so as to minimize the number of edges between the two pieces. Graph Bisection has applications in solving VLSI design and layout problems. Boppana [26] presented a new algorithm for Graph Bisection that, for almost all graphs in a certain class, will output the minimum-size bisection. The algorithm will also yield, for almost all such graphs, a proof that the output bisection is optimal.

Boppana has also been studying interactive proof systems. Interactive proof systems form a randomized generalization of the complexity class NP. Together with Hastad and Stathis Zachos (Brooklyn College), Boppana [27] showed that if every language in co-NP had a short interactive proof, then the polynomial-time hierarchy would collapse. As a corollary: if the Graph Isomorphism problem is NP-complete, then the polynomial-time hierarchy collapses.

Boppana plans to continue studying average-case algorithms for other NP-complete problems. Starting September 1987, he will join the faculty of Rutgers University, Department of Computer Science.

Val Breazu-Tannen

In collaboration with Meyer, Breazu-Tannen has investigated the conservative extension properties of the polymorphic type discipline as embodied in the Girard-Reynolds second-order lambda calculus. These properties state that reasoning about polymorphic programs is conservative over reasoning about the underlying data types or other basic features to which polymorphism was added. Given the known computational power of the Girard-Reynolds calculus, this research gives strong theoretical support to the idea of designing programming languages that avoid unrestricted recursion--all programs terminate--and thus gain in reasoning simplicity, while incorporating versatile features such as "types-as-values" and various tame forms of recursion [35][36].

In collaboration with Thierry Coquand from INRIA, France, Breazu-Tannen has described a general method for constructing models of the Girard-Reynolds polymorphic lambda calculus [33], also used in some of the proofs of the conservative extension results mentioned above.

All this is contained in Breazu-Tannen's Ph.D. thesis completed in January 1987 [32].

Future plans include attacking some conjectures suggested by the thesis research, as

well as working on type theories with equations between types and on the relationships between typed lambda calculi and higher-order categorical logic.

Thomas H. Cormen

Cormen has continued his research on concentrator switches. In [44], he showed that any n-input, n-output synchronous switch that ϵ-nearsorts its incoming valid bits (in the sense that each bit is within ϵ positions of its correct position in a fully sorted sequence) is also an $(n, m, 1 - \epsilon/m)$ partial concentrator switch if just the first m outputs are used. He then constructs two multichip partial concentrator switches based on the first few steps of the Revsort [127] and Columnsort [97] algorithms for sorting on a mesh. Both switches use the hyperconcentrator switch of Cormen and Leiserson [43][46] as a subcircuit and can be implemented efficiently in three dimensions. Signals incur at most $4 \lg n + O(1)$ gate delays in passing through these switches.

Cormen also received the award for the best presentation at the 1986 International Conference on Parallel Processing. The talk was entitled "A Hyperconcentrator Switch for Routing Bit-Serial Messages."

Claude Crepeau

Crepeau moved to MIT in September with a Master's degree from Universite de Montreal. His major achievements this year are related to Zero-Knowledge and other types of cryptographic protocols. His work on Zero-Knowledge protocols was done in collaboration with Gilles Brassard (U. de Montreal) and led to two papers [29][28]. Research on the "All or Nothing Disclosure of Secrets" protocol led to some results, both from a cryptographic point of view [30] and an information theoretic point of view [31]. This work was done in collaboration with Gilles Brassard and Jean-Marc Robert (McGill U.).

Crepeau's Master's thesis was about a Zero-Knowledge poker protocol that achieves confidentiality of the players' strategy [49][49]. Since his arrival at MIT, he has worked mostly on specific problems relating to Zero-Knowledge protocols, first with Goldwasser and then Micali. Later in the year, some results about the equivalence between the two known versions of Oblivious Transfer were achieved [50].

Aditi Dhagat

Dhagat started her first year as a graduate student at MIT in September. She worked with Goldwasser during the first semester in number theory and cryptography. During the second semester, she worked with Rivest on inference of finite automata by learning programs. She will return to cryptography during the summer working on signature schemes with Micali.

Paul N. Feldman

Feldman has spent this year polishing up the two major results of his thesis, namely,

an efficient scheme for verifiable secret sharing and a Byzantine Agreement protocol running in constant expected time. He collaborated with Manuel Blum (Berkeley) and Micali to show *How to Prove a Theorem so Nobody Else Can Claim It* [20].

Lance J. Fortnow

Fortnow transferred to MIT in September after one year at the University of California at Berkeley. His main result concerns perfect zero-knowledge, i.e., interactive protocols which disclose no knowledge in an information theoretic sense. His paper [57] shows that for any language which has a perfect zero-knowledge protocol, its complement has a short interactive proof. In particular, this result implies that if NP-complete problems had perfect zero-knowledge protocols then the polynomial time hierarchy would collapse. Some of this work was done while he was at Berkeley.

Fortnow also looked at other complexity issues including: The relativization of the conjecture that all NP-complete sets are isomorphic, limit points of sequences, monotone branching programs, and various other complexity issues dealing with interactive protocols and zero-knowledge.

Jeff Fried

Fried has been working on cost-universal elements for interconnection networks. With Leiserson and Park, he investigated several types of elements, addressing schemes, and on-line routing algorithms which were both simple and applicable to a family of interconnection networks including butterfly-fat-trees. This research also includes a revision of the classical model of VLSI communication cost (area or volume) to include variations in the cost of communication at different levels of any packaging hierarchy; for example wires between chips are much more costly than intra-chip wires. The goal of this work is to develop a very concrete notion of what 'cost-universal' networks (and elements) must do. As a class project with Cormen and Brad Kuszmaul (MIT), Fried designed a processor chip which should be capable of efficiently simulating a wide variety of low-degree interconnection networks. Called the NAP (for No Alu Processor), this chip will be fabricated over the summer.

Fried is also continuing research (begun at GTE Laboratories) on Optical Interconnects for VLSI, including a complexity model for optical interconnects.

Andrew V. Goldberg

Goldberg has been working in the area of algorithms, in particular network flow algorithms and parallel algorithms.

Working together with Robert Tarjan (Princeton), Goldberg has developed a successive approximation framework for solving the minimum-cost circulation problem [63][66]. This framework allowed them to obtain better bounds on the problem. Given a network with n vertices, m edges, and integer costs with absolute values bounded by

C, the algorithms described in [63][66] achieve an $O(log(nC)min(n^3,n^{5/3}m^{2/3},nmlogn))$ running time bounds, significantly improving the previous bounds. Recently, Goldberg and Tarjan have improved these results in two ways [64]. The running time bound has been improved to $O(log(nC)nmlog(n^2/m))$, and a natural modification of the minimum-cost framework has been shown to produce strongly polynomial algorithms. Unlike previous strongly polynomial algorithms, these algorithms use no rounding of costs or capacities.

Goldberg and Tarjan continued their work on the maximum flow problem [65]. During the summer of 1986, Goldberg has been working at Thinking Machines, Inc., implementing a parallel version of the algorithm on the Connection Machine. The experimental results [63] show that the algorithm is practical.

Goldberg has been working together with Plotkin on efficient parallel algorithms for coloring and matching in sparse graphs. They have developed a new algorithm to color a maximum degree Δ graph with $(\Delta+1)$ colors in $O(\log^* n)$ time on EREW PRAM, where n is the size of the graph [60][61]. The usefulness of their technique is exhibited in [62], where they show how to apply it to significantly improve the running times of a number of algorithms, including 5-coloring planar graphs and finding a maximal matching in planar graphs.

Sally A. Goldman

Goldman continued her work with Rivest on theoretical aspects of artificial intelligence [69][70] and completed a Master's thesis [68] based on this work. They developed a new algorithm for computing the maximum entropy probability distribution satisfying a set of constraints. Unlike previous approaches, their method is integrated with the planning of data collection and tabulation. They show how adding constraints and performing the associated additional tabulations can substantially speed up computation by replacing the usual iterative techniques with a noninteractive technique. Goldman is beginning new research with Rivest on the problem of machine learning.

Ronald I. Greenberg

Greenberg has been studying the issue of hardware-efficient parallel computation from the standpoint of physical layout considerations. The fat-tree architecture proposed by Leiserson and studied by Leiserson and Greenberg is a step in this direction. It can simulate any other network with comparable physical volume without requiring much more time (in a unit-delay model). Greenberg has been working on improving the mathematical analyses which yield these results and on including consideration of such issues as wire length and fault-tolerance.

Michelangelo Grigni

Grigni arrived in September with a B.S. from Duke University; he spent most of his

first year in classes. He is currently working on a project to implement the Quadratic Sieve factoring algorithm on the Connection Machine--a project begun with Klein and Walter Gillett (MIT AI Lab). His adviser is Sipser.

Johan Hastad

Hastad's main interest has been lower bounds for concrete problems. Hastad together with Beame proved in [15] that very powerful parallel machines require almost logarithmic time to compute simple functions like parity. They also establish that there is a very fine hierarchy: For slowly growing functions T there are languages that can be recognized using parallel resources in time T but not in time T-2. The techniques of this paper are very close to those of Hastad's thesis (supervised by Goldwasser) which was published as a book by MIT press [76]. The main results of this thesis are close to optimal lower bounds for small depth circuits. Also related to parallel computing, in [77] Hastad proves that there are functions which are one-way with respect to parallel computation. In particular there are permutations computable by constant depth circuits which are P-complete to invert.

Hastad has also been studying interactive proof systems. Interactive proof systems form a randomized generalization of the complexity class NP. Together with Boppana and Stathis Zachos (Brooklyn College), Hastad [27] showed that if every language in co-NP had a short interactive proof, then the polynomial-time hierarchy would collapse. As a corollary: if the Graph Isomorphism problem is NP-complete, then the polynomial-time hierarchy collapses. Hastad also studied other aspects of the computational complexity of interactive proofs. In [3], which is joint work with Goldwasser and Aiello, it is proved that, given an unbounded function F, there is an oracle A such that the class of languages recognizable by an interactive proof relative with $F(x)$ rounds relative to A is not contained in the polynomial time hierarchy relative to A. A limitation for the power of interactive proofs was proved together with Aiello [4]. They prove that given any language which has a zero-knowledge interactive proof of an unbounded number of rounds also has a two round interactive proof.

Hastad, also interested in the role of randomness in computation, together with Leighton and Newman, investigated how to cope with randomly occurring errors in large networks [79]. They prove that in the case of the hypercube an error-free network can be simulated by a network containing random errors with only a slight loss in efficiency. Randomness plays a quite different role in [80] where Hastad, Leighton and Rogoff analyze randomized protocols for communication. The communication model is that of a broadcast network and backoff protocols are analyzed. They prove that the previously proposed protocols linear backoff and exponential backoff have some theoretical drawbacks. Instead they propose quadratic backoff which has good provable theoretical properties and also seems to perform well in practice.

Finally Hastad has been working in the area of geometry of numbers. In [78] he proves that given a lattice L and a point x outside the lattice the is a vector in the dual lattice

which gives a fairly good estimate for the distance from x to L. This implies that the NP-complete nearest lattice point problem has an approximate counterpart which belongs to co-NP.

Rafael Hirschfeld

Hirschfeld has finished his Master's thesis research in pseudorandom number generators and complexity theory under Micali's supervision. He investigated pseudorandom generators that cannot feasibly be distinguished from true random sources, and their implications for the relationship between deterministic and probabilistic computation. Boppana and Hirschfeld [25] wrote an expository paper on this and related work. Hirschfeld is presently exploring directions for doctoral research.

Alexander T. Ishii

Ishii and Leiserson are preparing a joint paper describing the analysis of timing in VLSI circuits where latches are controlled by the level (high or low) of a clock signal rather than a transition (edge) of the clock. Such level-clocked circuits are frequently used in MOS VLSI design. A level-clocked circuit is modeled as a graph $G=(V,E)$, where V consists of components--latches and functional elements--and E represents the connections among the components. An algorithm is presented for determining whether a circuit operates properly that runs in $O(|V||E|+|V|^2 lgV)$ time. The work presented represents results that are to appear in Ishii's Master's thesis.

The timing analysis methods used are based on concepts from elementary logic and graph algorithms. The "intended" semantics of a designer's circuit are modeled using a syntactic interpretation of the functional elements of the circuit. Sufficient conditions are given for a circuit to operate properly under any interpretation of the functional elements, and under the assumption that the circuit must operate properly for all interpretations of the functional elements, the conditions are shown to be necessary as well. The conditions are reducible to a collection of simple linear constraints that can be solved using standard graph algorithms.

Burton S. Kaliski

Kaliski completed his Master's thesis describing the hardware used for the DES experiments reported in [92] and [128]. He also continues to work on the use of elliptic curves in cryptography.

Shlomo Kipnis

Kipnis together with Kilian and Leiserson have been investigating the power of bussed interconnection schemes for realizing permutations among chips [94]. They established a correspondence between bussed permutation architectures and difference covers for permutation sets. As an example, only $\sqrt{n}$ pins per chip are needed to realize all cyclic shifts among n chips in one clock cycle. Using point to point wires, $n-1$ pins per chip

are required. They also show that $4\sqrt{n}$ pins per chip suffice to realize any abelian group of permutations and that any general group of permutations requires $2\sqrt{n \lg n}-1$ pins per chip. Some other results include bussed interconnection schemes for hypercubes (d+1 pins per chip), shuffle-exchange graphs (3 pins per chip), and d-dimensional meshes (d+1 pins per chip).

Joe Kilian

Kilian, together with Kipnis and Leiserson, has been investigating the power of bussed interconnection schemes for realizing permutations among chips [94].

Together with Martin Abadi (MCC) and Joan Feigenbaum (U. Chicago), Kilian examined what can be informally considered an *oblivious transfer of capability* [1]. They considered protocols in which a polynomially bounded party enlists the aid of a highly powerful party to solve a problem. The polynomially bounded party trusts the powerful party, but wishes to conceal from him certain details about the problem. For example, the weaker person may want to compute $\log_g x \bmod p$, but does not want to reveal any information about x. An important subclass of these protocols constitutes an interactive generalization of the complexity class *P/poly*.

Recently, Kilian has worked on slightly altered models of interactive proof systems.

Philip N. Klein

Klein has been researching efficient use of parallel computation to solve graph problems. Recently he has been considering the usefulness of preprocessing in facilitating parallel speed-up. For example, searching a directed graph has proved a difficult problem to parallelize: to get a speed-up of s, previous methods seem to require about $n^2 s$ processors, where n is the number of nodes in the graph. Klein developed a technique whereby a speed-up of $s \leq \sqrt{m}$ can be achieved with only s processors, following a preprocessing stage whose output represents the graph using storage $ns+m$. (Here m is the number of edges in the graph.) The technique can be used to find, for example, all the nodes reachable from a given set of nodes in the graph. The technique has applications in artificial intelligence.

In October 1986, at FOCS, Klein delivered a paper ([95]), coauthored with John Reif, on an efficient parallel algorithm for determining whether a graph is planar, and, if so, finding a planar embedding. The algorithm is about two orders of magnitude more efficient than previous such algorithms; in fact, it is almost optimal. Klein gave a talk at LCS on the same topic the previous month.

Klein plans to continue investigating parallel algorithms and the role of preprocessing in efficient parallel computation.

Richard Koch

Koch is currently conducting research for his doctoral dissertation. He investigated the analysis of protocols for Ethernet-type communication channels. He is currently investigating a universal routing problem which would have have applications for distributed routing and the graph bisection problem.

Bruce Maggs

Maggs and Leiserson have refined the Distributed Random-Access Machine (DRAM) model [103] in which the communication requirements of a parallel algorithm can be evaluated. Maggs and Koch are looking for a "universal routing algorithm", an algorithm that determines how to quickly route a set of messages in an arbitrary network. A universal routing algorithm would have applications in parallel computing and possibly in finding a small bisector of a graph.

F. Miller Maley

Maley continued his work on the theory of single-layer wire routing with homotopy constraints. The theory, which applies to several mathematically natural wiring models, is the first rigorous treatment of general river routing problems. It derives necessary and sufficient conditions for routability, and leads to efficient methods for constructing optimal routings. Maley is currently working on extending these results to models of greater practical interest.

Stuart Mentzer

Mentzer has been working with Shmoys on the approximability of NP-hard combinatorial optimization problems. He has studied the question of whether graph problems in a metric plane allow better approximations than their more general triangle inequality versions. Results for six metric P-center problems have been obtained.

P-center problems are a type of emergency facility location problem where you wish to choose a set of p points so as to minimize the maximum distance from any point of a given set to the closest chosen point. In discrete versions the chosen point set is a subset of the given points. A simple algorithm delivers a 2-approximation (within a factor of two of the optimum) to any instance where the distances (given by a symmetric nonnegative distance matrix) obey the triangle inequality, and it is well known that doing better than a factor of two is NP-hard for the triangle inequality version.

Mentzer has shown (publication under preparation) that it is also NP-hard to better than 2-approximate the discrete and continuous P-center problems in the L^1 and L^∞ metric planes ($\frac{3}{2}$ was known for the continuous versions). In addition, he showed that it is NP-hard to better than $\sqrt{3}$-approximate these problems in the Euclidean plane ($\frac{2}{\sqrt{3}}$ was known for the continuous version). Techniques for improving this lower bound

have been found. Thus, of the problems studied, only the Euclidean versions leave open the possibility of better than 2-approximations (if P $\neq$ NP).

James K. Park

Park has been working with Leiserson on message-routing algorithms for the fat-tree [102] and related interconnection networks. They have developed a simple new routing algorithm for the butterfly-fat-tree, a variant of the fat-tree using only constant-sized switches. It is a deterministic, on-line algorithm, requiring $O(\lambda(M) \log n)$ delivery cycles and $O(\lambda M l \log n + \lambda(M) \log^2 n + \log^3 n)$ bit operations to deliver a message set M, where $\lambda(M)$ is the load factor the message set induces on the network and l the length in bits of the longest message.

Cynthia A. Phillips

Phillips and Leiserson developed a simple parallel algorithm for contraction of n-node bounded-degree planar graphs which solves the problem of region labeling in vision systems and leads to algorithms to compute spanning trees and biconnected components of bounded-degree planar graphs [104]. The connected components algorithm runs in $O(\lg n)$ randomized time or $O(\lg n \lg^* n)$ deterministic time on the restrictive exclusive-read exclusive-write PRAM model. They are currently trying to extend these results to planar graphs of arbitrary degree. Phillips is trying to develop an $O(\lg n)$-time parallel algorithm for region labeling in n-voxel 3D images by exploiting the restrictive topology and geometry of the spatial tesselation. She is also investigating the use of randomized search techniques for fast, simple connected components algorithms for general graphs.

Serge A. Plotkin

Plotkin together with Goldberg has been working on efficient parallel algorithms for coloring and matching in sparse graphs. They developed a new algorithm to color a maximum degree Δ graph with $(\Delta+1)$ colors in $O(\log^* n)$ time on EREW PRAM, where n is the size of the graph [60][61]. The usefulness of their technique is exhibited in [62], where together with Shannon (Purdue Univ.), they show how to apply it to significantly improve the running times of a number of algorithms, including 5-coloring planar graphs and finding a maximal matching in planar graphs.

In addition to his work on parallel algorithms, Plotkin has been working with Awerbuch on the problem of Leader Election in distributed asynchronous faulty network. They have developed a technique to approximate up to a constant factor the size of a dynamically growing asynchronous distributed network with amortized message complexity of $O(log^2|V|)$ per node, where $|V|$ is the final size of the network. They demonstrated how to apply this technique to construct an efficient distributed algorithm for Leader Election in a faulty network [14]. The algorithm has a message complexity $O(|E| + |V| \log^2 |V|)$, which is an improvement of $O(\log^2 |V|)$ over the previously known algorithms.

Satish B. Rao

Rao just completed his Master's thesis [119] supervised by Leighton on finding approximately optimal separators in planar graphs. He presented an algorithm to find optimal quotient separators (minimizing the ratio of the cost over the weight separated), and an approximate solution for optimal separators. The approximate solution in some sense trades balance for the size of the separator. The algorithm may have applications in various divide-and-conquer approaches to graph problems such as VLSI placement and routing.

Mark B. Reinhold

Dependent function types, which originated in the type theory of intuitionistic mathematics, have recently appeared in programming languages such as CLU, Pebble, and Russell. (A function has a dependent type when the type of its result depends upon the value of its argument.)

In [115], Reinhold and Meyer investigate the consequences of assuming that there exists a *type of all types* in a λ-calculus with dependent function types. The type of all types is the type of every type, including itself. When a language with dependent function types is enriched with the type-of-all-types assumption, enormous expressive power is gained at very little apparent cost. By reconstructing and analyzing a paradox of Girard, they show that this combination leads to several serious problems. The most significant of these are that for such a language (1) typechecking is undecidable, and (2) classical reasoning about programs is not sound.

Reinhold has produced a machine-verified construction of Girard's paradox. This construction, and the undecidability result, will be described in his forthcoming Master's thesis.

Jon G. Riecke

Riecke has been finishing research begun jointly with Kim B. Bruce. In [38], they examine the semantics of types in the programming language Miranda. The work focuses on a novel feature of Miranda, algebraic types, which create new types from old. Algebraic types, they argue, may be thought of as user-defined type constructors, and they show how to extend a denotational model to include interpretations for these type constructors.

Riecke plans to study the connections and differences between two styles of denotational semantics--the direct and continuation styles. Plotkin's notion of *full abstraction* will be particularly emphasized in this project.

Robert E. Schapire

Schapire, working with Rivest, has been investigating a new approach to the problem of inferring a finite state automaton based only on its input/output behavior [124].

They have developed and extended an algorithm which infers an FSA in time polynomial in its diversity, a new measure which may be exponentially smaller than the number of global states of the automaton.

Eric Schwabe

Schwabe started his first year as a graduate student at MIT in September. Having spent most of his year involved in course work, he will begin work in parallel computation with Leighton this summer, initially considering problems taken from a discussion of space-saving strategies for queue management in packet-routing algorithms.

Alan T. Sherman

Sherman completed his dissertation [128] and is now an Assistant Professor at Tufts University.

David Short

Short has been studying learning algorithms for artificial intelligence with Rivest, and is now studying mixed Eulerian graphs with Shmoys.

Kurt Sieber

Sieber is working on semantics and program verification for ALGOL-like programming languages with higher order procedures. The main goal is to develop Hoare-style calculi for reasoning about free procedures, i.e., procedures which are not *bound* by declarations, but only *specified* by semantic properties. Such reasoning is necessary if correctness proofs are to correspond to the structure of programs. In these proofs the correctness of a program should follow from procedure specifications and not depend on the particular procedure declarations.

In his current research two main directions can be distinguished:

- the development of new semantic models which can deal with free procedures *and* local variables,
- completeness questions about Hoare-style calculi for the language known as L4 [41].

New methods for defining the semantics of local variables are necessary, because traditional denotational semantics (based on complete partial orders and continuous functions) only work if a declaration is available for every procedure, i.e., they fail in the presence of free procedures. A first *ad hoc* solution to this problem was presented in [75]. Sieber is now developing a more systematic approach, based on so called locally complete partial orders and locally continuous functions. Moreover he is investigating the question of full abstraction for his new models.

In the language L4, procedures are not allowed to have access to global variables. In this case the local variables are harmless and the above mentioned semantic problems disappear. But L4 is still powerful enough to make traditional Hoare-style calculi fail. A relatively complete Hoare calculus has only recently been presented in [59]. In collaboration with Meyer and S. German (GTE Research Lab), Sieber is now trying to find generalizations of this result for Hoare formulas with free procedures.

Jeffrey Mark Siskind

Siskind has been continuing the research in the area of logic programming discussed in last year's report. This research has focused on the incorporation of dependency-directed backtracking and related search pruning techniques and heuristics into the control strategy of a Prolog implementation. Analysis of an abstract model of Prolog execution termed *decision problems* has highlighted the following four distinct sources of potential domain-independent search space reduction techniques.

- Constraint Propagation
- Selective Backtracking
- Caching *Nogoods*
- Incremental Context Switch

The literature on intelligent backtracking in Prolog discusses only selective backtracking and is unaware of the potential improvement afforded by the additional three techniques. On the other hand, a long tradition of propositional logic reasoning systems based on efficient graph representation techniques known as *Truth Maintenance Systems* have made use of some or all of these techniques. These systems however, solve only SAT problems and either do not handle variables and quantification at all or do so incompletely via a demonic invocation mechanism.

Siskind's research has shown how a symbolic unraveling of a resolution proof tree for a theorem in first-order logic can produce a corresponding infinite stream of SAT problems which contains a satisfiable SAT problem if the theorem has a proof. This unraveling process makes use of novel graph techniques to encode the constraints on unification between terms as part of the generated SAT problem. The result of this is that any pure Prolog program can be translated into a finite template which represents a corresponding infinite stream of SAT problems. The unraveling process and unification constraint graph techniques have been incorporated into a working implementation of pure Prolog called ConLog. As ConLog uses a TMS to solve the resulting SAT problems, it functions as an implementation of pure Prolog which provides all four of the above search space reduction techniques. ConLog has been tested on a number of examples from the literature and shows promise of reducing the amount of search and backtracking required on these examples.

This work is being supervised by Charles E. Leiserson.

Robert H. Sloan

Sloan has been working on problems in cryptography [117] together with Micali, and problems in learning theory [124] in collaboration with Rivest.

Joel Wein

Wein spent the year involved in course work. He will spend the summer working with Shmoys. Possible topics include parallel algorithms and approximation algorithms to NP-Complete problems.

Su-Ming Wu

Wu has continued to work on proving the correctness of the Micali-Vazirani algorithm for maximum matching in general graphs. A crucial component of the Micali-Vazirani matching algorithm is the labeling of the vertices the algorithm performs while finding a maximum matching. For his Master's thesis, supervised by Micali, Wu has proved that this labeling procedure is correct. He is considering how the techniques used in this proof can be extended to a proof of correctness of the entire algorithm.

References

1. Awerbuch, B. and S. Even. "Reliable Broadcast Protocols in Unreliable Networks." *Networks*, 16, (1987), 381-396.

2. Awerbuch, B. and R. Gallager. "A New Distributed Algorithm to Find Breadth First Search Trees." *IEEE Transactions on Information Theory*, (1987), 33, 315-322.

3. Awerbuch, B., O. Goldreich and R. Vainish. "On the Message Complexity of Broadcast: Basic Lower Bound." MIT/LCS/TM-325, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

4. Awerbuch, B. and S. Micali. "Dynamic Deadlock Resolution Protocols." *Proceedings of the 27th Symposium of the Foundation of Computer Science*, IEEE, 1986, 196-207.

5. Awerbuch, B. and S. Plotkin. "Approximating the Size of a Dynamically Growing Distributed Network." MIT/LCS/TM-328, MIT Laboratory for Computer Science, Cambridge, MA, April, 1987.

6. Beame, P.W. and J. Hastad. "Optimal Bounds for Decision Problems on the CRCW PRAM." *Proceedings of the 19th Symposium on Theory of Computing*, ACM, 1987, 83-93.

7. Berger, B., M. Brady, D. Brown and T. Leighton. "Nearly Optimal Bounds and Algorithms for Channel Routing." *Journal of Circuits and Systems*, submitted for publication.

8. Berman, F., D. Johnson, T. Leighton, P. Shor and L. Snyder. *Journal of Algorithms*, to appear.

9. Bhatt, S., F. Chung, T. Leighton and A. Rosenberg. "Optimal Simulation of Tree Machines." *Proceedings of the 27th Symposium of the Foundation of Computer Science*, IEEE, 1986, 274-282.

10. Bloom, B. "Constructing Two-Writer Atomic Registers." *Proceedings PODC*, ACM, to appear, 1987.

11. Blum , M. P. Feldman and S. Micali "How to Prove a Theorem so Nobody Else can Claim It." To appear, 1987.

12. Bodlaender, H.L. "A Better Lower Bound for Distributed Leader Finding in Bidirectional Asynchronous Rings of Processors." To appear, 1987.

13. Bodlaender, H.L. "Dynamic Programming on Graphs with Bounded Tree Width." MIT/LCS/TR, MIT Laboratory for Computer Science, Cambridge, MA, May 1987.

14. Bodlaender, H.L. "New Lower Bounds for Distributed Leader Finding in Asynchronous Rings of Processors." Submitted to *GI-Jahrestagung*.

15. Booth, K. and G. Lueker. "Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms." *Journal of Computer System Scinek*, 13, (1976), 335-339.

16. Boppana, R. and R. Hirschfeld. "Pseudorandom Generators and Complexity Classes." *Randomness and Computation.* Also *Advances in Computing Research*, JAI Press, 1987.

17. Boppana, R. B. "Eigenvalues and Graph Bisection: An Average-Case Analysis." Submitted for publication, 1987.

18. Boppana, R. B., J. Hastad and S. Zachos "Does Co-NP Have Short Interactive Proofs?" To appear, 1986.

19. Brassard, G. and C. Crepeau. "Non-Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond." *27th Symposium Foundations of Computer Science*, IEEE 1986,188-195.

20. Brassard, G. and C. Crepeau. "Zero-Knowledge Simulation of Boolean Circuits (Extended abstract)." *Advances in Cryptology: Proceedings Crypto '86.* Springer-Verlag, Lecture Notes in Computer Science, to appear, 1986.

21. Brassard, G. and C. Crepeau and J.-M. Robert. "All-or-Nothing Disclosure of Sec." *Advances in Cryptology: Proceedings Crypto '86*, Springer-Verlag, Lecture Notes in Computer Science, to appear, 1986.

22. Brassard, G. and C. Crepeau and J.-M. Robert. "Information Theoretic Reductions among Disclosure Problems." *27th Symposium Foundations of Computer Science*, IEEE, 1986, 168-173.

23. Breazu-Tannen V. "Conservative Extensions of Type Theories." MIT Department of Mathematics, Cambridge, MA, 1987.

24. Breazu-Tannen, V. and T. Coquand. "Extensional Models For Polymorphism." *TAPSOFT'87 -- Colloquium on Functional and Logic Programming and Specifications*, Springer-Verlag, Lecture Notes in Computer Science, 250, 1987, 291-307.

25. Breazu-Tannen, V. and A.R. Meyer. "Lambda Calculus With Constrained Types (Extended Abstract)." Logics of Programs, Springer-Verlag, Lecture Notes in Computer Science, 193, R. Parikh (ed.), 1985, 23-40.

26. Breazu-Tannen, V. and A.R. Meyer. "Computable Values Can be Classical." *14th Symposium on Principles of Programming Languages*, ACM, 1987, 238-245.

27. Breazu-Tannen, V. and A.R. Meyer. "Polymorphism is Conservative Over Simple Types." *2nd Symposium Logic in Computer Science*, IEEE, 1987, 7-17.

28. Bruce, K.B. and A.R. Meyer. "The Semantics of Second-Order Polymorphic Lambda Calculus." *Semantics of Data Types*, Springer-Verlag, 1984, Lecture Notes in Computer Science, G. Kahn, D. B. MacQueen and G. Plotkin (eds.), 173, 131-144. Also to appear in *Information and Computation*, 1988, coauthored with J.C. Mitchell.

29. Bruce, K. B. and J.G. Riecke. "The Semantics of Miranda's Algebraic Types." *Workshop on Mathematical Foundations of Programming Language Semantics*, Springer-Verlag, Lecture Notes in Computer Science.

30. Bui, T., S. Chaudhuri, T. F. Leighton and M. Sipser. "Graph Bisection Algorithms With Good Average Case Behaviour." *Combinatorica*,, to appear, 1987.

31. Bui, T., S. Chaudhuri, T. Leighton and M. Sipser. "Graph Bisection Algorithms With Good Average Case Behaviour." *Combinatorica*, to appear, 1987.

32. Chung, T., T. Leighton and A. Rosenberg. "Embedding Graphs in Books: A Layout Problem With Applications to VLSI Design." *SIAM Journal Algebraic and Discrete Methods*, 8, 1987, 33-58.

33. Clark Jr., E.M. "Programming Language Constructs for Which It Is Impossible to Obtain Good Hoare Axiom Systems." *JACM*, 26, (1979) 129-147.

34. Coffman, E. and T. Kadota, T. Leighton and L. Schepp "Stochastic Analysis of Storage Fragmentation. *Proceedings Int. Sem. on Teletraffic Analysis and Computer Performance Evaluation*, 1986.

35. Cormen, T.H. "Concentrator Switches for Routing Messages in Parallel Computers." MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, 1986.

36. Cormen, T.H. "Efficient Multichip Partial Concentrator Switches." *1987 International Conference on Parallel Processing*, IEEE Computer Society Press, to appear, 1987 Available as MIT/LCS/TM-322, 1987.

37. Cormen, T.H. and C.E. Leiserson. U.S. Patent Application: "Hyperconcentrator Switch for Message Routing." 1986.

38. Cormen, T.H. and C. E. Leiserson. "A Hyperconcentrator Switch for Routing Bit-Serial Messages." *1986 International Conference on Parallel Processing*, IEEE Computer Society Press, 1986, 721-728. Also available as MIT/LCS/TM-321, 1987.

39. Crepeau, C. "A Secure Poker Protocol that Minimizes the Effects of Player Coalitions." *Advances in Cryptology: Proceedings Crypto '85*, Springer-Verlag, Lecture Notes in Computer Science, 218, H.C. Williams (ed.), (1985), 73-86.

40. Crepeau, C. "Protocole Cryptographique De Poker a L'Aveugle Permettant la Confidentialiede la Strategie." S.M. thesis, Dept. d'Informatique et de R.O., Universite de Montreal, 1986.

41. Crepeau, C. "A Zero-Knowledge Poker Protocol that Achieves Confidentiality of the Players. Strategy or How to Achieve an Electronic Poker Face." *Advances in Cryptology: Proceedings Crypto '86*, Springer-Verlag, Lecture Notes in Computer Science, to appear, 1986.

42. Crepeau, C. "Equivalence Between Two Flavours of Oblivious Transfers (Abstract)." 1987, submitted to Crypto '87.

43. Dwork, C., D.B. Shmoys and L. Stockmeyer. "Flipping Persuasively in Constant Expected Time. *27th Symposium Foundations Computer Science*, IEEE, 1986, 222-232.

44. Elias, P. "List Decoding for Noisy Channels." 957 IRE Wescon Convention Record Part 2., Institute of Radio Engineers (now IEEE). 1957, 94-104.

45. Elias, P. "Zero Error Capacity for List Decoding." Quarterly Progress Report, MIT Research Laboratory of Electronics, Cambridge, MA, January 1958, 88-90.

46. Elias, P. "Interval and Recency Rank Source Coding: Two On-Line Adaptive Variable-Length Schemes." *IEEE Transactions on Information Theory*, 33, (1987), 3-10.

47. Fekete, A., N. Lynch, M. Merritt and W. Weihl. "Nested Transactions and Read-Write Locking." *6th Symposium Principles of Database Systems*, to appear, 1987.

48. Fekete, A., N. Lynch and L. Shrira. "A Modular Proof of Correctness for a Network Synchronizer." *2nd International Workshop on Distributed Algorithms*, Springer-Verlag, to appear, 1987.

49. Fortnow, L. "The Complexity of Perfect Zero-Knowledge." *Randomness and Computation*, JAI Press, 1987, S. Micali (ed.), *Advances in Computing Research*, to appear.

50. Gallager, R.G. Information Theory and Reliable Communication. John Wiley and Sons, 1968.

51. German, S.M., E.M. Clarke and J.Y. Halpern. "True Relative Completeness of an Axiom System for the Language *L4*." *Symposium Logic in Computer Science*, IEEE, 1986, 11-25.

52. Goldberg A. and S. Plotkin. "Efficient Parallel Algorithms for $(\Delta + 1)$ Coloring and Maximal Independent Set Problems." MIT/LCS/TM-320, MIT Laboratory for Computer Science, Cambridge, MA, 1987.

53. Goldberg, A. and S. Plotkin. "Parallel $(\Delta + 1)$ Coloring of Constant-Degree Graphs." *IPL*, 24, 4, 1987.

54. Goldberg A. and S. Plotkin and G. Shannon. "Parallel Symmetry Breaking in Sparse Graphs." *19th Symposium Theory of Computing*, ACM 1987, 315-324.

55. Goldberg, A.V. "Efficient Graph Algorithms for Sequential and Parallel Computers." MIT/LCS/TR-374, Laboratory for Computer Science, Cambridge, MA, January 1987.

56. Goldberg, A.V. and R. E. Tarjan. "Finding Minimum-Cost Circulations by Successive Approximation." To appear, 1987.

57. Goldberg, A.V. and R. E. Tarjan. "A New Approach to the Maximum Flow Problem." *Journal of the ACM*, 1987.

58. Goldberg, A.V. and R. E. Tarjan. "Solving Minimum-Cost Flow Problems by Successive Approximation." *19th Symposium Theory of Computing*, May 1987, 7-18.

59. Goldman, K. J. and N. A. Lynch. "Quorum Consensus in Nested Transaction Systems." *6th Symposium Principles of Distributed Computing*, ACM, to appear, 1987.

60. Goldman, S.A. "Efficient Methods for Calculating Maximum Entropy Distributions." MIT/LCS/TR-391, MIT Laboratory for computer Science. Cambridge, MA, 1987.

61. Goldman, S.A. and R.L. Rivest. "Making Maximum Entropy Computations Easier by Adding Extra Constraints (Extended Abstract)." *Proceedings 6th Workshop on Maximum Entropy and Bayesian Methods in Applied Statistics*, 1986, to appear.

62. Goldman, S.A. and R.L. Rivest. "Uncertainty in Artificial Intelligence." North-Holland. A Non-Iterative Maximum Entropy Algorithm, L.N. Kanal and J. Lemmer (eds.), to appear.

63. Goldreich, O., S. Micali and A. Wigderson. "Proofs That Yield Nothing But Their Validity and a Methodology of Cryptographic Protocol Design." *27th IEEE Symposium Foundations Computer Science*, IEEE, 1986, 174-186.

64. Goldreich, O. and S. Micali and A. Wigderson. "How to Play Any Mental Game or A Completeness Theorem for Distributed Protocols with Honest Majority." *19th Symposium Theory of Computing*, ACM, 1987, 218-229.

65. Goldwasser, S., S. Micali and C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems." submitted for publication, 1987.

66. Goldwasser, S., S. Micali and R. L. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks." *Japan-U.S. Conference on Algorithms*, Kyoto, Japan, 1986, 287-310.

67. Greenberg, R.I. and C. E. Leiserson. "Randomized Routing on Fat-Trees. " *Advances in Computing Research, Fourth volume: Randomness and Computation*, JAI Press, S. Micali (ed.), to appear. Earlier versions available in MIT/LCS/TM-307 and 26th IEEE Symposium Foundations of Computer Science, (1985), 241-249.

68. Halpern, J.Y., A.R. Meyer and B.A. Trakhtenbrot. "The Semantics of Local Storage, Or What Makes the Free-list Free?" *11th Symposium Principles of Programming Languages*, ACM, 1984, 245-257.

69. Hastad, J. "Computational Limitations of Small-Depth Circuits." MIT Press, Cambridge, MA, 1986.

70. Hastad, J. "Oneway permutations in NC^0." *IPL*, 1986, to appear.

71. Hastad, J. "Dual Vectors and the Nearest Lattice Point Problem." *Combinatorica*, to appear, 1987.

72. Hastad, J. F.T. Leighton, M. Newman. "Reconfiguring a Hypercube in the Presence of Faults." *19th Symposium Theory of Computing*. ACM, 1987, 274-284.

73. Hastad, J., F.T. Leighton and B. Rogoff. "Analysis of Backoff Protocols for Multiple Access Channels." *19th Symposium Theory of Computing*, 1987, ACM, 241-253.

74. Heath, L. S. and S. Istrail. "The Pagenumber of Genus g Graphs is $O(g)$". *19th Symposium Theory of Computing*, 1987, ACM, 388-397.

75. Heath, L. S. and A. L. Rosenberg. "An Optimal Mapping of the Algorithm FFT Onto the Hypercube Architecture." 1987, submitted for publication.

76. Heath, L. S., A. L. Rosenberg, B.T. Smith. "The Physical Mapping Problem for Parallel Architectures." 1986, submitted for publication.

77. Herlihy, M., N. Lynch, M. Merritt, W. Weihl, W. "On the Correctness of Orphan Elimination Algorithms. *17th International Symposium Fault-Tolerant Computing*, 1987, IEEE, to appear.

78. Hochbaum, D.S. and D.B. Shmoys. "A polynomial Approximation Scheme for Scheduling Onuniform Processors: Using the Dual Approximation Approach." *SICOMP*.

79. Hochbaum, D.S. and D.B. Shmoys. "A Unified Approach to Approximation Algorithms for Bottleneck Problems." *JACM*, 33, (1986), 533-550.

80. Hochbaum, D.S. and David B. Shmoys. " A Best Possible Parallel Approximation Algorithm for a Graph Theoretic Algorithm." *IFORS '87*, 1987, North-Holland, to appear.

81. Hochbaum, D.S. and D. B. Shmoys. "Using Dual Approximation Algorithms for Scheduling Problems: Practical and Theoretical Results." *JACM*, 34, (1987), 144-162.

82. JaJa, J and J. Simon. "Parallel Algorithms in Graph Theory: Planarity Testing." *SICOMP*, 11, 1982, 313-328.

83. Kaliski, B. S. "Design and Reliability of Custom Hardware for DES Cycling Experiments." M.S. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, 1987.

84. Kaliski, B. S. "A Pseudo-Random Bit Generator Based on Elliptic Algorithms." *Advances in Cryptology: Proceedings of Crypto 86,* Springer-Verlag, 1987.

85. Kaliski, B.S., R.L. Rivest and A.T. Sherman. "Is the Data Encryption Standard a Pure Cipher? (Results of more cycling studies on DES)." *Advances in Cryptology: Proceedings of Crypto 85,* Springer-Verlag, H.C. Williams (ed.), 212-226.

86. Karloff, H. J. and D.B. Shmoys. "Parallel Algorithms for Edge Coloring Problems." *Journal of Algorithms*, 8, (1987), 39-52.

87. Kilian, J., S. Kipnis and C.E. Leiserson. "The Organization of Permutation Architectures With Bussed Interconnections." 1987, submitted.

88. Klein, P.N. and J.H. Reif. "An Efficient Parallel Algorithm for Planarity." *27th Symposium on Foundations of Computer Science.* 1986, IEEE, 464-477.

89. Klein, P.N. and J.H. Reif. "Parallel Time $O(\log n)$ Acceptance of Deterministic CFL's on an Exclusive-Write PRAM." *SICOMP,* to appear, 1987.

90. Leighton, F.T. "Tight Bounds on the Complexity of Parallel Sorting. *IEEE,* C-34, (1985), 344-354.

91. Leighton, F.T. "A Survey of Bounds and Algorithms for Channel Routing." 1986, submitted to *Algorithmica.*

92. Leighton, F. T. and C. E. Leiserson. "A Survey of Algorithms for Integrating Wafer-Scale Systolic Arrays." *A VLSI Circuit and Architecture Design,,* Marcel-Dekker, Inc., E. Swartzlander, Jr. (ed.), To appear, 1987. An early version appears in the proceedings of the *IFIP Conference on Wafer-Scale Integration,* 1986, 177-195.

93. Leighton, F. T. and R.L. Rivest. "Estimating a Probability Using Finite Memory." *IEEE Transactions on Information Theory,* IE-32, 6, (November 1986), 733-742.

94. Leighton, F.T. and A. Rosenberg. "Three-Dimensional Circuit Layouts." *Sicomp,* 15, (1986), 718-813.

95. Leiserson, C.E. "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE*, C-34, (1985), 892-901.

96. Leiserson, C. E. and Maggs, B. M. *Algorithmica*. "Communication-Efficient Parallel Algorithms for Distributed Random-Access Machines." To appear, 1987. Earlier version available as MIT/LCS/TM-318, MIT Laboratory for Computer Science, Cambridge, MA., December 1986.

97. Leiserson, C. E. and C.A. Phillips. "Parallel Contraction of Planar Graphs (Extended Abstract)." 1987, submitted for publication.

98. Leiserson, C. E. and C.A. Phillips. "A Space-Efficient Algorithm for Finding the Connected Components of Rectangles in the Plane." MIT/LCS/TM-323, MIT Laboratory for Computer Science, Cambridge, MA, February 1987.

99. Lenstra, J.K., D.B. Shmoys, and E. Tardos. "Scheduling Unrelated Parallel Machines." 1987, submitted for publication.

100. Lynch, N. and M. Merritt. "Introduction to the Theory of Nested Transactions." *Int'l Conference Database Theory*, September 1986, Springer-Verlag, 278-305.

101. Lynch, N. and M. Tuttle. "Hierarchical Correctness Proofs for Distributed Algorithms." *6th Symposium Principles of Distributed Computing*, August To appear, 1987.

102. Maggs, B. and S. Plotkin. "Minimum-Cost Spanning Tree as a Path-Finding Problem in a Closed Semiring." 1986, submitted for publication.

103. Maley, F. M. "Compaction with Automatic Jog Introduction." MIT/LCS/TM-372, MIT Laboratory for Computer Science, Cambridge, MA, November 1986.

104. Meyer, A.R. "Communication in the Electronic Forum **types@xx.lcs.mit.edu**.." February 1986.

105. Meyer, A.R. "Floyd-Hoare Logic Determines Semantics." *TCS*, 1987, to appear. Earlier version available in *IEEE Symposium Logic in Computer Science*, (June 1986), 44-48.

106. Meyer, A.R. and J.Y. Halpern. "Axiomatic Definitions of Programming Languages: a Theoretical Assessment." *JACM*, 29, (1982), 555-576.

107. Meyer, A.R., J. C. Mitchell, E. Moggi and R. Statman. "Empty Types in

Polymorphic Lambda-Calculus." *14th Symposium Principles of Programming Languages*. ACM, 1987, 253-262.

108. Meyer, A.R. and M.B. Reinhold. "'Type' Is Not a Type: Preliminary Report." *13th Symposium Principles of Programming Languages*, ACM, 1986, 287-295.

109. Meyer, A.R. and M. Wand. Logics of Programs. Springer-Verlag, 193, "Continuation semantics in typed lambda-calculi (Summary). 1985, R. Parikh (ed.), 219-224.

110. Micali, S., C. Rackoff and R. Sloan. "The Notion of Security for Probabilistic Cryptosystems." *SICOMP*, To appear, 1987.

111. Quinlan, J.R. and R.L. Rivest. "Inferring Decision Trees Using the Minimum Description Length Principle." 1987, Draft manuscript available from authors.

112. Rao, S. "Finding Small Edge Separators in Planar Graphs." M.S. thesis 1987, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA,

113. Reed, D.P. "Naming and Synchronization in a Decentralized Computer System." Ph.D. dissertation, MIT/LCS/TR-205, MIT Laboratory for Computer Science, Cambridge, MA, 1978.

114. Rivest, R.L. "Game Tree Searching by Min/Max Approximation." *AI Journal*, to appear, 1987. Earlier version available as MIT/LCS/TR-311, September 1986.

115. Rivest, R.L. "Learning Decision-Lists." *Machine Learning*, To appear, 1987.

116. Rivest, R.L. Network Control by Bayesian Broadcast." *IEEE Transactions on Information Theory*, IT-33, 3, (May 1987), 323-328.

117. Rivest, R. L. and R.E. Schapire. "A New Approach to Unsupervised Learning in Deterministic Environments." *Proceedings Machine Learning Workshop*, Langley (ed.), To appear, 1987.

118. Rivest, R.L. and R. Sloan. "A New Model for Inductive Inference." 1987, submitted for publication.

119. Rosenberg, A.L. "The Diogenes Approach to Testable Fault-Tolerant Arrays of Processors." *IEEE*, C-32, 1983, 902-910.

120. Schnorr, C.P. and A. Shamir. "An Optimal Sorting Algorithm for Mesh Connected Computers. *18th Symposium Theory of Computing*, ACM, 1986, 255-263.

121. Sherman, A.T. "Cryptology and VLSI (a Two-Part Dissertation): I. Detecting and Exploiting Algebraic Weaknesses in Cryptosystems II. Algorithms for Placing Modules On a Custom VlSi Chip." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, 1986.

122. Vitanyi, P. and Awerbuch, B. "Atomic Register Access by Asynchronous Hardware. *27th Symposium Foundations Computer Science*, IEEE, 1986, 233-243.

123. Wozencraft, J.M. "List Decoding." Quarterly Progress Report, MIT Research Laboratory of Electronics, Cambridge, MA, January 1958, 90-95.

124. Zyablov, V.V. and M.S. Pinsker. "List Cascade Decoding." *Problemy Peredachi Informatsi*, 17, 1981, 29-33. English translation in Problems of Information Transmission, 236-240, 1982

Publications

1. Abadi, M., J. Feigenbaum and J.F. Kilian. "On Hiding Information From An Oracle." *Proceedings of the 19th Symposium on the Theory of Computation*, IEEE, 1987, 195-203.

2. Afek, Y., B. Awerbuch and E. Gafni. "Adapting Communication Protocols to Dynamic Input and Network Topology." *Proceedings of the 28 Symposium on Foundations of Computer Science*, IEEE, To appear, 1987.

3. Aiello, W.A., S. Goldwasser and J. Hastad. "On the Power of Interaction." *Proceedings of the 27th Symposium on Foundations of Computer Science*, IEEE, 1986, 368-379.

4. Aiello, W.A. and J. Hastad. "Perfect Zero-Knowledge Languages Can Be Recognized in Two Rounds." *Submitted to the 28th Symposium on Foundations of Computer Science*, IEEE, 1987.

5. Ashcroft, R. "When Ignorance Is Bliss." *The Economist*, 303, (1987), 93-94.

6. Aspnes, J.D. "Timestamp Ordering and Nested Transactions." E.E. and S.M. Thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, 1987.

7. Awerbuch, B. "Adapting Communication Protocols to Dynamic Input and Network Topology." MIT/LCS/TM-326, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

8. Awerbuch, B. "Controlling Worst-Case Performance of a Communication Protocol and Dynamic Resource Management." MIT/LCS/TM-326, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

9. Awerbuch, B. "Optimal Distributed Algorithms for Minimum Spanning Trees, Leader Election, Counting and Related Problems." *Proceedings for the 19th Symposium of the Theory of Computing*, ACM, 1987, 230-240.

10. Awerbuch, B. and S. Even. "Reliable Broadcast Protocols in Unreliable Networks." *Networks*, (1987), 16, 381-396.

11. Awerbuch, B. and R. Gallager. "A New Distributed Algorithm to Find Breadth First Search Trees." *IEEE Transactions on Inf. Theory*, (1987), 33, 315-322.

12. Awerbuch, B., O. Goldreich and R. Vainish. "On the Message Complexity of Broadcast: Basic Lower Bound." MIT/LCS/TM-325, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

13. Awerbuch, B. and S. Micali. "Dynamic Deadlock Resolution Protocols." *Proceedings of the 27th Symposium of the Foundation of Computer Science*, IEEE, 1986, 196-207.

14. Awerbuch, B. and S. Plotkin. "Approximating the Size of a Dynamically Growing Distributed Network." MIT/LCS/TM-328, MIT Laboratory for Computer Science, Cambridge, MA, April, 1987.

15. Beame, P.W. and J. Hastad. "Optimal Bounds for Decision Problems on the CRCW PRAM." *Proceedings of the 19th Symposium on Theory of Computing*, ACM, 1987,83-93.

16. Berger, B., M. Brady, D. Brown and T. Leighton. "Nearly Optimal Bounds and Algorithms for Channel Routing." *Journal of Circuits and Systems*, submitted for publication.

17. Berman, F., D. Johnson, T. Leighton, P. Shor and L. Snyder. *Journal of Algorithms*, to appear.

18. Bhatt, S., F. Chung, T. Leighton and A. Rosenberg. "Optimal Simulation of Tree Machines." *Proceedings of the 27th Symposium of the Foundation of Computer Science*, IEEE, 1986, 274-282.

19. Bloom, B. "Constructing Two-Writer Atomic Registers." *Proceedings PODC*, ACM, to appear, 1987.

20. Blum , M. P. Feldman and S. Micali "How to Prove a Theorem so Nobody Else can Claim It." To appear 1987.

21. Bodlaender, H.L. "A Better Lower Bound for Distributed Leader Finding in Bidirectional Asynchronous Rings of Processors." to appear 1987.

22. Bodlaender, H.L. "Dynamic Programming on Graphs with Bounded Tree Width." MIT/LCS/TR, MIT Laboratory for Computer Science, Cambridge, MA, May 1987.

23. Bodlaender, H.L. "New Lower Bounds for Distributed Leader Finding in Asynchronous Rings of Processors." Submitted to *GI-Jahrestagung*.

24. Booth, K. and G. Lueker. "Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms." *Journal of Computer System Scinek*, 13, (1976), 335-339.

25. Boppana, R. and R. Hirschfeld. "Pseudorandom Generators and Complexity Classes." *Randomness and Computation.* Also *Advances in Computing Research,* JAI Press, 1987.

26. Boppana, R. B. "Eigenvalues and Graph Bisection: An Average-Case Analysis." Submitted for publication, 1987.

27. Boppana, R. B., J. Hastad and S. Zachos "Does Co-NP Have Short Interactive Proofs?" to appear, 1986.

28. Boppana, R.B. and R. Hirschfeld. Randomness and Computation. *Advances in Computing Research*, 5, JAI Press, To appear, 1987. A preliminary version appeared as an MIT Master's thesis by the second author, Department of Electrical Engineering and Computer Science, June 1986.

29. Brassard, G. and C. Crepeau and J.-M. Robert. "Information Theoretic Reductions among Disclosure Problems." *27th Symposium Foundations of Computer Science,* IEEE, 1986, 168-173.

30. Brassard, G. and C. Crepeau and J.-M. Robert. *Advances in Cryptology: Proceedings Crypto '86.* Springer-Verlag. "All-or-Nothing Disclosure of Sec. , 1986, Series=Lecture Notes in Computer Science, to appear.

31. Brassard, G. and C. Crepeau. "Advances in Cryptology: Proceedings Crypto '86. {Springer-Verlag. "Zero-Knowledge Simulation of Boolean Circuits (Extended abstract). 1986, Lecture Notes in Computer Science, to appear.

32. Brassard, G. and C. Crepeau. "Non-Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond." *27th Symposium Foundations of Computer Science,* IEEE 1986,188-195.

33. Breazu-Tannen V. "Conservative Extensions of Type Theories." MIT Department of Mathematics, Cambridge, MA, 1987.

34. Breazu-Tannen, V. and A.R. Meyer. "Computable Values Can be Classical." *14th Symposium on Principles of Programming Languages,* ACM, 1987, 238-245.

35. Breazu-Tannen, V. and A.R. Meyer. "Polymorphism is Conservative Over Simple Types." *2nd Symposium Logic in Computer Science,* IEEE, 1987, 7-17.

36. Breazu-Tannen, V. and A.R. Meyer. Logics of Programs. Springer-Verlag, "Lambda Calculus With Constrained Types (Extended abstract). 1985, Lecture Notes in Computer Science, 193, R. Parikh (ed.), 23-40.

37. Breazu-Tannen, V. and Thierry Coquand. "Extensional Models For Polymorphism." *TAPSOFT'87 -- Colloquium on Functional and Logic Programming and Specifications,* Springer-Verlag, Lecture Notes in Computer Science, 250, Berlin, 1987, 291-307.

38. Bruce, K.B. and A.R. Meyer. "The Semantics of Second-Order Polymorphic Lambda Calculus." G. Kahn, D. B. MacQueen and G. Plotkin (eds.), *Semantics of Data Types.* Springer-Verlag, 1984, Lecture Notes in Computer Science, 173, 131-144. Also to appear in *Information and Computation,* 1988, coauthored with J.C. Mitchell.

39. Bruce, K. B. and J.G. Riecke. "The Semantics of Miranda's Algebraic Types." *Workshop on Mathematical Foundations of Programming Language Semantics,* Springer-Verlag, Lecture Notes in Computer Science.

40. Bui, T., S. Chaudhuri, T. F. Leighton and M. Sipser. "Graph Bisection Algorithms With Good Average Case Behaviour." *Combinatorica,* To appear, 1987.

41. Schnorr, C.P. and A. Shamir. "An Optimal Sorting Algorithm for Mesh Connected Computers. *18th Symposium Theory of Computing,* ACM, 1986, 255-263.

42. Chung, T., T. Leighton and A. Rosenberg. "Embedding Graphs in Books: A Layout Problem With Applications to VLSI Design." *SIAM Journal Algebraic and Discrete Methods,* 8, (1987), 33-58.

43. Chung, T., T. Leighton and A. Rosenberg. "Embedding Graphs in Books: A Layout Problem With Applications to VLSI Design." *SIAM Journal Algebraic and Discrete Methods,* 8, (1987), 33-58.

44. Clark Jr., E.M. "Programming Language Constructs for Which It Is Impossible to Obtain Good Hoare Axiom Systems." *JACM,* 26, (1979) 129-147.

45. Coffman, E., T. Kadota, T. Leighton and L. Schepp "Stochastic Analysis of Storage Fragmentation. *Proceedings Int. Sem. on Teletraffic Analysis and Computer Performance Evaluation,* 1986.

46. Cormen, T.H. "Efficient Multichip Partial Concentrator Switches." *1987 International Conference on Parallel Processing,* IEEE Computer Society Press, To appear, 1987. Available as MIT/LCS/TM-322, 1987.

47. Cormen, T.H. "Concentrator Switches for Routing Messages in Parallel Computers." MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, 1986.

48. Cormen, T.H. and C.E. Leiserson. U.S. Patent Application: "Hyperconcentrator Switch for Message Routing." 1986.

49. Cormen, T.H. and C. E. Leiserson. "A Hyperconcentrator Switch for Routing Bit-Serial Messages." *1986 International Conference on Parallel Processing,* IEEE Computer Society Press, 1986, 721-728. Also available as MIT/LCS/TM-321, 1987.

50. Crepeau, C. "Equivalence Between Two Flavours of Oblivious Transfers (Abstract)." 1987, submitted to Crypto '87.

51. Crepeau, C. "Protocole Cryptographique De Poker a L'Aveugle Permettant la Confidentialiede la Strategie." S.M. thesis, Dept. d'Informatique et de R.O., Universite de Montreal, 1986.

52. Crepeau, C. "A Secure Poker Protocol that Minimizes the Effects of Player Coalitions." *Advances in Cryptology: Proceedings Crypto '85,* Springer-Verlag, Lecture Notes in Computer Science, 218, H.C. Williams (ed.), (1985), 73-86.

53. Crepeau, C. "A Zero-Knowledge Poker Protocol that Achieves Confidentiality of the Players." Strategy or How to Achieve an Electronic Poker Face." *Advances in Cryptology: Proceedings Crypto '86*, Springer-Verlag, Lecture Notes in Computer Science, 1986, to appear.

54. Dwork, C., D.B. Shmoys and L. Stockmeyer. "Flipping Persuasively in Constant Expected Time. *27th Symposium Foundations Computer Science*, IEEE, 1986, 222-232.

55. Elias, P. "Zero Error Capacity for List Decoding." Quarterly Progress Report, MIT Research Laboratory of Electronics, Cambridge, MA, January 1958, 88-90.

56. Elias, P. "Interval and Recency Rank Source Coding: Two On-Line Adaptive Variable-Length Schemes." *IEEE Transactions on Information Theory*, 33, (1987), 3-10.

57. Elias, P. "List Decoding for Noisy Channels." 957 IRE Wescon Convention Record Part 2., Institute of Radio Engineers (now IEEE). 1957, 94-104.

58. Fekete, A., N. Lynch, M. Merritt and W. Weihl. "Nested Transactions and Read-Write Locking." *6th Symposium Principles of Database Systems*, To appear, 1987.

59. Fekete, A., N. Lynch and L. Shrira. "A Modular Proof of Correctness for a Network Synchronizer." *2nd International Workshop on Distributed Algorithms*, Springer-Verlag, to appear, 1987.

60. Fortnow, L. "The Complexity of Perfect Zero-Knowledge." *Randomness and Computation*, JAI Press, 1987, S. Micali (ed.), *Advances in Computing Research*, to appear.

61. Gallager, R.G. Information Theory and Reliable Communication. John Wiley and Sons, 1968.

62. German, S.M., E.M. Clarke and J.Y. Halpern. "True Relative Completeness of an Axiom System for the Language *L4*." *Symposium Logic in Computer Science*, IEEE, 1986, 11-25.

63. Goldberg A. and S. Plotkin. "Efficient Parallel Algorithms for $(\Delta + 1)$ Coloring and Maximal Independent Set Problems." MIT/LCS/TM-320, MIT Laboratory for Computer Science, Cambridge, MA, 1987.

64. Goldberg A. and S. Plotkin and G. Shannon. "Parallel Symmetry Breaking

in Sparse Graphs." *19th Symposium Theory of Computing*, ACM 1987, 315-324.

65. Goldberg, A. and S. Plotkin. "Parallel (Δ + 1) Coloring of Constant-Degree Graphs." *IPL*, 24, 4, 1987.

66. Goldberg, A.V. "Efficient Graph Algorithms for Sequential and Parallel Computers." MIT/LCS/TR-374, Laboratory for Computer Science, Cambridge, MA, January 1987.

67. Goldberg, A.V. and R. E. Tarjan. "Solving Minimum-Cost Flow Problems by Successive Approximation." *19th Symposium Theory of Computing*, May 1987, 7-18.

68. Goldberg, A.V. and R. E. Tarjan. "A New Approach to the Maximum Flow Problem." *Journal of the ACM*, 1987.

69. Goldberg, A.V. and R. E. Tarjan. "Finding Minimum-Cost Circulations by Successive Approximation." To appear, 1987.

70. Goldman, K. J. and N. A. Lynch. "Quorum Consensus in Nested Transaction Systems." *6th Symposium Principles of Distributed Computing*, ACM, to appear, 1987.

71. Goldman, S.A. "Efficient Methods for Calculating Maximum Entropy Distributions." MIT/LCS/TR-391, MIT Laboratory for computer Science, Cambridge, MA, 1987.

72. Goldman, S.A. and R.L. Rivest. "Making Maximum Entropy Computations Easier by Adding Extra Constraints (Extended Abstract)." *Proceedings 6th Workshop on Maximum Entropy and Bayesian Methods in Applied Statistics*, to appear, 1986.

73. Goldman, S.A. and R.L. Rivest. "Uncertainty in Artificial Intelligence." North-Holland. A Non-Iterative Maximum Entropy Algorithm, L.N. Kanal and J. Lemmer (eds.), to appear.

74. Goldreich, O., S. Micali and A. Wigderson. "Proofs That Yield Nothing But Their Validity and a Methodology of Cryptographic Protocol Design." *27th IEEE Symposium Foundations Computer Science*, IEEE, 1986, 174-186.

75. Goldreich, O. and S. Micali and A. Wigderson. "How to Play Any Mental Game or A Completeness Theorum for Distributed Protocols with Honest Majority." *19th Symposium Theory of Computing*, ACM, 1987, 218-229.

76. Goldwasser, S., S. Micali and C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems." submitted for publication, 1987.

77. Goldwasser, S., S. Micali and R. L. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks." *Japan-U.S. Conference on Algorithms*, Kyoto, Japan, 1986, 287-310.

78. Greenberg, R.I. and C. E. Leiserson. "Randomized Routing on Fat-Trees. " *Advances in Computing Research, Fourth volume: Randomness and Computation*, JAI Press, S. Micali (ed.), to appear. Earlier versions available in MIT/LCS/TM-307 and 26th IEEE Symposium Foundations of Computer Science, (1985), 241-249.

79. Halpern, J.Y., A.R. Meyer and B.A. Trakhtenbrot. "The Semantics of Local Storage, Or What Makes the Free-list Free?" *11th Symposium Principles of Programming Languages*, ACM, 1984, 245-257.

80. Hastad, J. "Oneway permutations in NC^0." *IPL*, to appear, 1986.

81. Hastad, J. "Dual Vectors and the Nearest Lattice Point Problem." *Combinatorica*, to appear, 1987.

82. Hastad, J. F.T. Leighton, M. Newman. "Reconfiguring a Hypercube in the Presence of Faults." *19th Symposium Theory of Computing*. ACM, 1987, 274-284.

83. Hastad, J., F.T. Leighton and B. Rogoff. "Analysis of Backoff Protocols for Multiple Access Channels." *19th Symposium Theory of Computing*, 1987, ACM, 241-253.

84. Hastad, J. "Computational Limitations of Small-Depth Circuits." MIT Press, Cambridge, MA, 1986.

85. Heath, L. S., A. L. Rosenberg, B.T. Smith. "The Physical Mapping Problem for Parallel Architectures." 1986, submitted for publication.

86. Heath, L. S. and A. L. Rosenberg. "An Optimal Mapping of the Algorithm FFT Onto the Hypercube Architecture." 1987, submitted for publication.

87. Heath, L. S. and S. Istrail. "The Pagenumber of Genus g Graphs is $O(g)$". *19th Symposium Theory of Computing*, 1987, ACM, 388-397.

88. Herlihy, M., N. Lynch, M. Merritt, W. Weihl, W. "On the Correctness of Orphan Elimination Algorithms. *17th International Symposium Fault-Tolerant Computing*, 1987, IEEE, to appear.

89. Hochbaum, D.S. and David B. Shmoys. " A Best Possible Parallel Approximation Algorithm for a Graph Theoretic Algorithm." *IFORS '87*, 1987, North-Holland, to appear.

90. Hochbaum, D.S. and D.B. Shmoys. "A polynomial Approximation Scheme for Scheduling Onuniform Processors: Using the Dual Approximation Approach." *SICOMP.*

91. Hochbaum, D.S. and D. B. Shmoys. "Using Dual Approximation Algorithms for Scheduling Problems: Practical and Theoretical Results." *JACM*, 34, (1987), 144-162.

92. Hochbaum, D.S. and D.B. Shmoys. "A Unified Approach to Approximation Algorithms for Bottleneck Problems." *JACM*, 33, (1986), 533-550.

93. JaJa, J and J. Simon. "Parallel Algorithms in Graph Theory: Planarity Testing." *SICOMP*, 11, 1982, 313-328.

94. Kaliski, B. S. "Design and Reliability of Custom Hardware for DES Cycling Experiments." M.S. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, 1987.

95. Kaliski, B. S. "A Pseudo-Random Bit Generator Based on Elliptic Algorithms." *Advances in Cryptology: Proceedings of Crypto 86*, Springer-Verlag, 1987.

96. Kaliski, B.S., R.L. Rivest and A.T. Sherman. "Is the Data Encryption Standard a Pure Cipher?" (Results of more cycling studies on DES)." *Advances in Cryptology: Proceedings of Crypto 85*, Springer-Verlag, H.C. Williams (ed.), 212-226.

97. Karloff, H. J. and D.B. Shmoys. "Parallel Algorithms for Edge Coloring Problems." *Journal of Algorithms*, 8, (1987), 39-52.

98. Kilian, J., S. Kipnis and C.E. Leiserson. "The Organization of Permutation Architectures With Bussed Interconnections." 1987, submitted.

99. Klein, P.N. and J.H. Reif. "Parallel Time $O(\log n)$ Acceptance of Deterministic CFL's on an Exclusive-Write PRAM." *SICOMP*, to appear, 1987.

100. Klein, P.N. and J.H. Reif. "An Efficient Parallel Algorithm for Planarity." *27th Symposium on Foundations of Computer Science*. 1986, IEEE, 464-477.

101. Leighton, F. T. and C. E. Leiserson. "A Survey of Algorithms for Integrating Wafer-Scale Systolic Arrays." *A VLSI Circuit and Architecture Design,*, Marcel-Dekker, Inc., E. Swartzlander, Jr. (ed.), To appear, 1987. An early version appears in the proceedings of the *IFIP Conference on Wafer-Scale Integration,* 1986, 177-195.

102. Leighton, F. T. and R.L. Rivest. "Estimating a Probability Using Finite Memory." *IEEE Transactions on Information Theory,* IE-32, 6, (November 1986), 733-742.

103. Leighton, F.T. "A Survey of Bounds and Algorithms for Channel Routing." 1986, submitted to *Algorithmica.*

104. Leighton, F.T. "Tight Bounds on the Complexity of Parallel Sorting. *IEEE,* C-34, (1985), 344-354.

105. Leighton, F.T. and A. Rosenberg. "Three-Dimensional Circuit Layouts." *Sicomp* 15, (1986), 718-813.

106. Leiserson, C. E. and Maggs, B. M. *Algorithmica.* "Communication-Efficient Parallel Algorithms for Distributed Random-Access Machines." To appear, 1987. Earlier version available as MIT/LCS/TM-318, MIT Laboratory for Computer Science, Cambridge, MA., December 1986.

107. Leiserson, C.E. "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE,* C-34, (1985), 892-901.

108. Leiserson, C. E. and C.A. Phillips. "A Space-Efficient Algorithm for Finding the Connected Components of Rectangles in the Plane." MIT/LCS/TM-323, MIT Laboratory for Computer Science, Cambridge, MA, February 1987.

109. Leiserson, C. E. and C.A. Phillips. "Parallel Contraction of Planar Graphs (Extended Abstract)." 1987, submitted for publication.

110. Lenstra, J.K., D.B. Shmoys, and E. Tardos. "Scheduling Unrelated Parallel Machines." 1987, submitted for publication.

111. Lynch, N. and M. Merritt. "Introduction to the Theory of Nested Transactions." *Int'l Conference Database Theory,* September 1986, Springer-Verlag, 278-305.

112. Lynch, N. and M. Tuttle. "Hierarchical Correctness Proofs for Distributed Algorithms." *6th Symposium Principles of Distributed Computing,* August To appear, 1987.

113. Maggs, B. and S. Plotkin. "Minimum-Cost Spanning Tree as a Path-Finding Problem in a Closed Semiring." 1986, submitted for publication.

114. Maley, F. M. "Compaction with Automatic Jog Introduction." MIT/LCS/TM-372, MIT Laboratory for Computer Science, Cambridge, MA, November 1986.

115. Meyer, A.R., J. C. Mitchell, E. Moggi and R. Statman. "Empty Types in Polymorphic Lambda-Calculus." *14th Symposium Principles of Programming Languages*. ACM, 1987, 253-262.

116. Meyer, A.R. "Communication in the Electronic Forum `types@xx.lcs.mit.edu`.." February 1986.

117. Meyer, A.R. "Floyd-Hoare Logic Determines Semantics." *TCS*, 1987, to appear. Earlier version available in *IEEE Symposium Logic in Computer Science*, (June 1986), 44-48.

118. Meyer, A.R. and J.Y. Halpern. "Axiomatic Definitions of Programming Languages: a Theoretical Assessment." *JACM*, 29, (1982), 555-576.

119. Meyer, A.R. and M. Wand. "Continuation Semantics in Typed Lambda-Calculi (Summary)." Logics of Programs., 193, Springer-Verlag, R. Parikh (ed.), 1985, 219-224.

120. Meyer, A.R. and M.B. Reinhold. "'Type' Is Not a Type: Preliminary Report." *15th Symposium Principles of Programming Languages*, ACM, 1986, 287-295.

121. Micali, S., C. Rackoff and R. Sloan. "The Notion of Security for Probabilistic Cryptosystems." *SICOMP*, To appear, 1987.

122. Plotkin, G.D. "LCF Considered As a Programming Language." *TCS*, 5, (1977), 223-257.

123. Quinlan, J.R. and R.L. Rivest. "Inferring Decision Trees Using the Minimum Description Length Principle." 1987, Draft manuscript available from authors.

124. Rao, S. "Finding Small Edge Separators in Planar Graphs." M.S. thesis 1987, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA,

125. Reed, D.P. "Naming and Synchronization in a Decentralized Computer

System." Ph.D. dissertation, MIT/LCS/TR-205, MIT Laboratory for Computer Science, Cambridge, MA, 1978.

126. Rivest, R.L. and R. Sloan. "A New Model for Inductive Inference." 1987, submitted for publication.

127. Rivest, R.L. Network Control by Bayesian Broadcast." *IEEE Transactions on Information Theory*, IT-33, 3, (May 1987), 323-328.

128. Rivest, R.L. "Game Tree Searching by Min/Max Approximation." *AI Journal*, to appear, 1987. Earlier version available as MIT/LCS/TR-311, September 1986.

129. Rivest, R.L. "Learning Decision-Lists." *Machine Learning*, To appear, 1987.

130. Rivest, R. L. and R.E. Schapire. "A New Approach to Unsupervised Learning in Deterministic Environments." *Proceedings Machine Learning Workshop*, Langley (ed.), To appear, 1987.

131. Rosenberg, A.L. "The Diogenes Approach to Testable Fault-Tolerant Arrays of Processors." *IEEE*, C-32, 1983, 902-910.

132. Sherman, A.T. "Cryptology and VLSI (a Two-Part Dissertation): I. Detecting and Exploiting Algebraic Weaknesses in Cryptosystems II. Algorithms for Placing Modules On a Custom VlSi Chip." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, 1986.

133. Sloan, R. H. "The Notion of Security for Probabilistic Public-key Cryptosystems." MIT/LCS/TR-379, MIT Laboratory for computer Science, Cambridge, MA.

134. Vitanyi, P. and Awerbuch, B. "Atomic Register Access by Asynchronous Hardware. *27th Symposium Foundations Computer Science*, IEEE, 1986, 233-243.

135. Wozencraft, J.M. "List Decoding." Quarterly Progress Report, MIT Research Laboratory of Electronics, Cambridge, MA, January 1958, 90-95.

136. Zyablov, V.V. and M.S. Pinsker. "List Cascade Decoding." *Problemy Peredachi Informatsi*, 17, 1981, 29-33. English translation in Problems of Information Transmission, 236-240, 1982

Talks

1. Aiello, W., S. Goldwasser and J.Hastad. "On the Power of Interaction." Lecture given at IBM Almaden Research Center, Stanford University, University of California at Berkeley, University of Southern California, February 1987.

2. Aiello, W., S. Goldwasser and H.J. "On the Power of Interaction." Lecture given at 27th IEEE Symposium Foundations of Computer Science,Toronto, October 1986.

3. Beame, P.W. and J.T. Hastad. " Nearly Optimal Lower Bounds for CRCW PRAM's." Lecture given at Harvard University, October 1986. A preliminary version of the results in the following lecture.

4. Boppana, R.B. " Eigenvalues and Graph Bisection: An Average-Case Analysis." Lecture given at Rutgers University, March 1987

5. Breazu-Tannen, V. "Conservative Extension: A Benefit of Recursion-Free, Polymorphic Computing." Lecture given at University of Torino (January), University of Pisa, IBM T.J. Watson Research Center (February), University of Chicago, University of Indiana, University of Pennsylvania, Cornell University (March), University of California Los Angeles, Carnegie-Mellon University (April), Massachusetts Institute of Technology (May), 1987.

6. Cormen, T.H. "Efficient Multichip Partial Concentrator Switches." Lecture given at MIT VLSI Research Review, Spring 1987, May 1987.

7. Cormen, T.H. and C.E. Leiserson. " A Hyperconcentrator Switch for Routing Bit-Serial Messages." Lecture given at 1986 International Conference on Parallel Processing, 1986.

8. Feldman, P.N. "An efficient Scheme for Verifiable Secret Sharing." Lecture given at MIT laboratory for Computer Science, November 1986.

9. Feldman, P.N. " A Protocol for Byzantine Agreement Running in Constant Expected Time." Lecture given At MIT Laboratory for Computer Science, April 1987.

10. Goldberg, A.V. "A New Approach to the Maximum Flow Problem." Lecture given at AT&T Bell Laboratories, IBM Yorktown Research Center, Harris Corporation, Cornell University, New York University, Columbia University, University of Texas at Austin, 1986-1987.

11. Goldberg, A.V. "On Finding and Exact Solution of a Zero-One Knapsack Problem." Lecture given at Cornell University, 1987.

12. Goldberg, A.V. "Solving Minimum-Cost Flow Problems By Successive Approximation." Lecture given at University of California at Berkeley, Stanford University, Princeton University, C.M.U., University of Toronto, MIT, 1987.

13. Goldman, S.A. "Making Maximum Entropy Computations Easier By Adding Extra Constraints." Lecture given at Brown University, Department of Computer Science, July 1986.

14. Goldman, S.A. "Making Maximum Entropy Computations Easier By Adding Extra Constraints." Lecture given at 6th Workshop on Maximum Entropy and Bayesian Methods in Applied Statistics, Seattle University, August 1986.

15. Greenberg, R.I. "Randomized Routing and Hardware-Efficient Parallel Computation." Lecture given at Department of Computer Science, Washington University, March 1987.

16. Hastad, J. "Randomized Routing and Hardware-Efficient Parallel Computation". Lecture given at Department of Computer Science, Washington University, March 1987.

17. Hastad, J. "Almost Optimal Lower Bounds for Small-Depth Circuits". Lecture given at Dartmouth College, Linkoping University, Stockholm University, January 1987.

18. Heath, L.S. "The Pagenumber of Genus G Graphs is O(G)." Lecture given at Department of Mathematics, MIT, March 1987.

19. Kaliski, B.S. "A Pseudo-Random Bit Generator Based on Elliptic Logarithms." Lecture given at Crypto 86 Conference, August 1986.

20. Killian, J., S. Kipnis and C. Leiserson. "Th Organization of Permutation Architectures With Bussed Interconnections." Lecture given at MIT. Fall 1987 VLSI Research Review, December 1986.

21. Klein, P.N. "An Efficient Parallel Algorithm for Planarity." Lecture given at MIT laboratory for Computer Science, 1986.

22. Leighton. "A Survey of Bounds and Algorithms for Channel Routing." Lecture given at Aegean Workshop on Computing, May 1986.

23. Leighton, F. "Simulating Special Purpose Networks with General Purpose Networks." Lecture given at LIDS workshop on Distributed Systems, October 1986.

24. Leighton, F. "An Introduction to Networks Parallel Computation and Vlsi Design." Lecture given at Army Math Steering Committee, Knoxville (November, 86), Mathematical Association of America National Meeting, San Antonio (January, 87), Cornell Distinguished Lecture Series, (April, 87) University Cincinatti Distinguished Lecture Series (April, 87), 1986, 1987.

25. Leighton, F.T. "Reconfiguring Networks Around Faults." Lecture given at Cornell Distinguished Lecture Series, April 1987.

26. Leighton, F.T. "Some Computational Properties of the Hypercube." Lecture given at BBN, (February, 87) Princeton workshop on Algorithms and Complexity, (March) Cornell Distinguished Lecture Series, (April), 1987.

27. Leiserson, C.E. "Communication-Efficient Parallel Graph Algorithms." Lecture given at Graph Theory Day (New York Academy of Sciences), Albany, New York, October 1986.

28. Lynch, N. "Correctness Conditions for Highly Available Replicated Databases." Lecture given at 6th Symposium. Principles of Distributed Computing, August 1986.

29. Lynch, N. "Hierarchical Proofs for Distributed Algorithms." Lecture given at Workshop on Communication and Distributed Computing, October 1986.

30. Lynch, N. "The Theory of Nested Transactions." Lecture given at University of California Irvine, U.C.L.A., Computer Corporation of America, Brown University, Washington University, 1987.

31. Maley, F.M. "Single-Layer Wire Routing." Lecture given at Princeton University, Department of Computer Science, Bell Communications Research, DEC Systems Research Center, Cornell University, Department of Computer Science, AT&T Bell Laboratories, Murray Hill, Carnegie Mellon University, Department of Computer Science, February/March 1987.

32. Meyer, A.R. "Computable Values Can Be Classical." Lecture given at Year of Programming Workshop on Types, University of Texas, Austin, June 1987.

33. Meyer, A.R. "Looping Combinators in Polymorphic Lambda Calculus." Lecture given at Mid-Atlantic Mathematical Logic Seminar, Philadelphia, PA, February 1987

34. Micali, S. "Detecting and resolving Dynamic Deadlocks." Lecture given at Mathematical Foundations of Computer Science Conference, Czechoslovakia (August), York University, Toronto, (October), 1986 Meeting on Complexity Theory, Germany (November), Carnegie-Mellon University, PA (December), 1986.

35. Micali, S. "Proofs, Knowledge and Computation." Lecture given at Mathematical Foundations of Computer Science Conference, Czechoslovakia (August), York University, Toronto, (October), 1986 Meeting on complexity Theory, Germany (November), Carnegie-Mellon University, PA (December), 1986.

36. Micali, S. "Recent Trends in Complexity Based Cryptography." Lecture given at Centro Ricerche per le Applicazioni Industrial, Naples, Italy, July 1986.

37. Micali, S. "The Completeness Theorem for Protocols with Honest Majority." Lecture given at University of Salerno, Italy (July), Yale University, CT (December), University of Rome, Italy (January, '87), 1986, 1987.

38. Plotkin, S. "Parallel Symmetry Breaking in Sparse Graphs." Lecture given at MIT Laboratory for Computer Science, May 1987.

39. Riecke, J.G. "The Semantics of Miranda's Algebraic Types." Lecture given at MIT Laboratory for Computer Science (January), Workshop on Mathematical Foundations of Programming Language Semantics, Tulane University (March), 1987.

40. Rivest, R.L. "A New Approach to Unsupervised Learning in Deterministic Environments." Lecture given at Princeton University Department of Computer Science, April 1987.

41. Rivest, R.L. "A New Approach to Unsupervised Learning in Deterministic Environments." Lecture given at MIT Laboratory for Computer Science May 1987.

42. Sherman, A.T. "Computer Education in the Public Schools: What Should It Be?" Lecture given at Bentley College, March 1987.

43. Sherman, A.T. "What is a Zero-Knowledge Proof?" Lecture given at Brandeis University, April 1987.

44. Shmoys, D.B. "Flipping Persuasively in Constant Expected Time." Lecture given at MIT Laboratory for Computer Science, Mathematical Sciences

Research Institute, Princeton University, FOCS, Tel Aviv University, Technion, 1986-1987.

45. Shmoys, D.B. "Analyzing Approximation Algorithms: Objectives, Impossibilities and possibilities. Lecture given at Tel Aviv University, January 1987.

46. Shmoys, D.B. "A Polynomial Approximation Scheme for Machine Scheduling on Uniform Processors: Using the Dual Approximation Approach." Lecture given at Cornell University, June 1987.

47. Shmoys, D.B. "Scheduling Unrelated Parallel Machines." Lecture given at Center for Mathematics and Computer Science, Amsterdam, January 1987.

48. Sieber, K. "Reasoning About Local Variables." Lecture given at MIT Laboratory for Computer Science November 1986.

49. Sloan, R. "The Notion of Security for Probabilistic Cryptosystems." Lecture given at Crypto '86, August 1986.

THEORY OF DISTRIBUTED SYSTEMS

Academic Staff

N. A. Lynch, Group Leader

Graduate Students

J. Aspnes
B. Coan
A. Fekete
K. Goldman
M. Tuttle
J. Welch

Support Staff

E. Pothier

Visiting Faculty

Y. Moses
L. Shrira
P. Vitanyi

1. INDIVIDUAL PROGRESS REPORTS

James Aspnes

During this period, James Aspnes carried out an MS thesis project entitled "Timestamp Ordering and Nested Transactions". Using the [16] model for database concurrency control and recovery, he described a general method for proving serial correctness of concurrency control algorithms which use timestamp ordering. He then used this method to prove serial correctness of the object history mechanism described in [23].

Brian Coan

During this period, Brian Coan has worked on the following projects:

He finished his Ph.D. thesis which gives several new results in the area of fault-tolerant distributed computing. His specific interest is consensus protocols, that is, protocols that enable correct processors to reach agreement in the presence of disruptive behavior by faulty processors. The results presented in the thesis are as follows: Two new efficient agreement protocols, one randomized and one deterministic; a method for efficiently transforming a protocol that reaches agreement on a single bit into a protocol that reaches agreement on values chosen from a larger set; a general method for compiling a protocol that tolerates relatively benign processor faults into one that tolerates more serious processor faults; and a strengthening of the known lower bound on the number of rounds of communication required by consensus protocols.

Using communication primitives and a message validation scheme developed by Bracha, Coan has constructed a compiler that increases the fault-tolerance of certain asynchronous protocols. Specifically, it transforms a "source protocol" that is resilient to crash faults into an "object protocol" that is resilient to Byzantine faults. His compiler can simplify the design of protocols for the Byzantine fault model because it breaks the design process into two steps. The first step is to design a protocol for the crash fault model. The second step, which is completely mechanical, is to compile the protocol into one for the Byzantine fault model. He uses the compiler to produce a new asynchronous approximate agreement protocol that operates in the Byzantine fault model. Specifically, he designs a new asynchronous approximate agreement protocol for the crash fault model and he observes that this protocol can be compiled into a protocol for the Byzantine fault-model. In the Byzantine fault model, the new protocol improves in several respects on the performance of the asynchronous approximate agreement protocol of Dolev et al.

The task of achieving consensus among the correct processors in a fault-tolerant distributed computer system has been recognized as a fundamental problem in distributed computing. Many consensus protocols are known. Some of these require an amount of communication that is exponential in the number of processors. Coan has developed a general simulation of any synchronous consensus protocol by a

communication-efficient protocol (i.e., a protocol with communication cost polynomial in the number of processors). An important corollary of the simulation technique is a new communication-efficient protocol for a particular consensus problem called the agreement problem. This new protocol uses about half the number of rounds required by the best previously-known communication-efficient agreement protocol. His new protocol approaches the known lower bound for rounds to within a small factor arbitrarily close to one. The only previously known protocols that achieve the lower bound for rounds use an amount of communication that is exponential in the system size.

In designing fault-tolerant distributed database systems, a frequent goal is making the system highly available despite component failure. With Brian Oki and Elliot Kolodner, Coan has examined software approaches to achieving high availability in the presence of partitions. In particular, they consider various replicated-data management protocols that maintain database consistency and attempt to increase database availability when networks partition. They conclude that no protocol does better than a bound which they have determined. Their conclusions hold under the assumption that the pattern of data accesses by transactions obeys a uniformity assumption. There may be some particular distribution for which specialized protocols can increase availability.

With Jennifer Lundelius Welch, he has studied the transaction commit problem under realistic timing assumptions. They identify an almost asynchronous model, which they claim is more realistic than some (synchronous) models that have been studied previously. In this model they give a randomized transaction commit protocol based on Ben-Or's randomized asynchronous Byzantine agreement protocol. The expected number of asynchronous rounds until their protocol terminates is a small constant, and the number of failstop faults tolerated is optimal.

Alan Fekete

During the past year, Alan worked on three subjects: approximate agreement, database concurrency control, and verification of network algorithms. He has been studying the question of approximate agreement, which is a variant of interactive consistency (Byzantine agreement) where the processors are required to reach values that are close together, rather than identical. This problem is interesting because it can be solved in an asynchronous system (whereas interactive consistency has no solution in such systems). Following his earlier work on this problem in synchronous systems, Alan showed that in an asynchronous system where processors fail by crashing or omitting to send messages, no solution can perform better than a simple round-by-round protocol.

Alan has been working also on developing and verifying algorithms for managing data in a Database Management System with nested transactions. In recent years several researchers, particularly those building general purpose distributed systems, have begun to explore the possibility of giving transactions (the basic unit of atomicity in a system)

additional structure, so that operations within a transaction can be performed concurrently (but serializably), or aborted independently. In this case we say the system provides *nested transactions*. An example of such a system is the Argus system, developed at MIT. Implementing such a system requires extending protocols for concurrency control and recovery that have been studied extensively when transactions are not nested. His work (in collaboration with Professors Nancy Lynch and William Weihl (MIT) and Dr. Michael Merritt (AT&T Bell Labs) has been concerned with an algorithm using locking. They proved the correctness of an algorithm using read and write locks, developed by Moss for Argus. They then found a way to extend Moss' algorithm to allow the use of more semantic information expressed in a table listing which operations of each data type do not interfere with each other. This algorithm is based on an algorithm for database systems without nesting, which was due to Weihl. Both Moss' algorithm, and the new "Conflict-Based Locking" were proved correct by showing that objects implemented using them satisfy a simple local condition called dynamic atomicity, and also proving that a nested transaction system is serially correct if every object is dynamic atomic.

The third topic of Alan's research has been the use of the new I/O automaton model of computation (due to Lynch, Merritt, and Tuttle) to describe and verify a protocol due to Awerbuch for network synchronization. This protocol allows programs that were written for a synchronous network to run in an asynchronous failure-free network, with as little overhead in time and communication as possible. This work (performed together with Professor Lynch and Dr. Liuba Shrira (MIT)) provides a proof whose organization follows closely the informal arguments made by Awerbuch. In particular the proof lays bare the precise modularity of the algorithm, which combines different techniques for synchronization within and between clusters of nodes in the network.

Ken Goldman

During this period, Ken Goldman completed his MS thesis, entitled "Data Replication in Nested Transaction Systems". Gifford's basic Quorum Consensus algorithm for data replication is generalized to accommodate nested transactions and transaction failures (aborts). A formal description of the generalized algorithm is presented using the new Lynch-Merritt input-output automaton model for nested transaction systems. This formal description is used to construct a complete (yet simple) proof of correctness that uses standard assertional techniques and is based on a natural correctness condition. Nondeterminism is used in the algorithm description to yield a correctness proof that is independent of any particular programming language or implementation. The presentation and proof treat issues of data replication entirely separately from issues of concurrency control and recovery.

Nancy A. Lynch

The main project Nancy Lynch worked on this year was the development of a theory for nested transactions, a joint project with Dr. Michael Merritt of AT&T Bell Laboratories. (The concept of nested transactions is central to new languages and

systems for distributed computing.) The theory allows careful description of the semantics of nested transactions, as well as description and correctness proofs for algorithms which implement them. The work began with a paper [16] last summer, in which they presented a semantic model and stated the basic correctness conditions to be satisfied by nested transaction systems. In this paper, they presented and proved correctness of an exclusive locking algorithm for nested transactions.

With colleagues at MIT and elsewhere, they continued by treating an interesting variety of algorithms. With Dr. Maurice Herlihy of CMU and Prof. Bill Weihl, they presented proofs for two algorithms for management of "orphan" transactions [13]. With Alan Fekete and Weihl, they extended the work in [16] to read-write locking [7] and are now working on a further extension to a new algorithm for general commutativity-based locking. Lynch and Goldman presented a proof for a replicated data-management algorithm [12]. Lynch has been helping Jim Aspnes to carry out a similar treatment of timestamp-based algorithms [1]. The model has provided a vocabulary for describing an amazing variety of interesting algorithms, and has permitted proofs which follow the natural modularity which system designers use in talking about the algorithms informally.

At bottom, all of this work is based on the "I/O automaton" model of concurrent computation, a new model which emphasizes the distinction between input and output events. This model was developed by Lynch and Mark Tuttle this year (and last) [17]. In addition to using it to model concurrency control algorithms, Lynch and Tuttle used it to carry out an interesting hierarchical proof of a distributed network resource-allocation algorithm. Lynch and her students are continuing to develop this model and to apply it to many other problems in concurrency. For instance, she helped Bard Bloom to describe his new algorithm for implementing two-writer atomic registers from one-writer atomic registers, by using I/O automata as a framework for presenting all the definitions, algorithms and proofs [2]. She is helping Jennifer Welch to use this model to present algorithms for mutual exclusion and Dining and Drinking Philosophers. She has worked with Fekete and Dr. Liuba Shrira to present a modular decomposition of a network synchronizer algorithm of Baruch Awerbuch [9].

She has also been working with Dr. Leslie Lamport on a new technique for carrying out lattice-structured correctness proofs for distributed network algorithms. They have been using it for reasoning about algorithms for minimum spanning tree computation and for bank audit.

She plans to continue the work on nested transaction theory, continue developing the I/O automaton model, and continue trying to apply this model to study a variety of application areas.

Yoram Moses

Yoram Moses spent part of the period of July 1 - December 31, 1986 at MIT and was affiliated with the TDS Group in the other part. During that period his main efforts

were in writing the paper "Programming Simultaneous Actions Using Common Knowledge" with Mark Tuttle. This paper presents a thorough analysis of a large class of problems involving fault-tolerant protocols for coordination among processors in a distributed system. The novel aspect of this work is that it uses a knowledge-based analysis to uncover the precise structure of the problems involved. As a consequence of this analysis, this work constructs for the first time protocols for these problems that are optimal in a much stronger sense than was previously known possible: they behave optimally in the face of each and every particular faulty behavior of the system. The class of problems considered includes simultaneous Byzantine agreement, the distributed firing squad problem and other problems. Yoram presented this paper at a theory seminar at MIT in August, and Mark presented it at the Symposium on the Foundations of Computer Science in Toronto in October. The paper appeared in the proceedings to the conference, and was invited for a special issue of the journal *Algorithmica*. Yoram also presented this paper at the University of Toronto in November and at the Mathematical Center in Amsterdam in December. The predecessor of this paper, a paper called "Knowledge and common knowledge in a Byzantine environment: The case of crash failures" that Yoram wrote with Cynthia Dwork, appeared as MIT/LCS/TM-300 in July, and was accepted for publication in the *Journal of Information and Communication*.

Liuba Shrira

This year Dr. Shrira's main interests were in programming methodology and specifications in distributed computing.

With A. Fekete and N. Lynch, Dr. Shrira worked on modular specifications of network protocols [FLS]. The work analyzed a network synchronizing algorithm by B Awerbuch designed to be used as subcomponent in derivation of other protocols. Modular specification and correctness proof were given to the algorithm which enable them to be reused in specifications and proofs of the derived protocols . The work uses the I/O automaton model [LT] and provides further evidence to the versatility of the model.

Dr. Shrira completed an earlier work on layered composition of distributed protocols. In this work R. Gerth and Dr. Shrira suggest a sound and relatively complete method for correctness proof of layered composition. Contrary to the previous works, this method is modular [11].

With R. Ladin and B. Liskov, Dr. Shrira worked on a new efficient fault tolerant data replication schema. The schema improves availability of the system by exploiting the semantic knowledge of the application to relax the up to date consistency constraint [15].

Dr. Shrira participated in the LCS Mercury Project and proposed an expressive linguistic mechanism for integrating Mercury streams into programming languages. The idea is to map an asynchronous remote procedure invocation on a stream into a special

data type. With B. Liskov, Dr. Shrira developed a uniform local and remote streamed concurrency integration schema for Argus language [19].

Mark Tuttle

Mark completed his Master's Thesis in the second semester of this year, joint work with Lynch entitled "Hierarchical Correctness Proofs for Distributed Algorithms." One of the major obstacles to progress in the field of distributed computing is that many of the important algorithms seem to be too complex for rigorous understanding. It is often the case that the designers of an algorithm have an intuitive understanding of how the algorithm behaves, but that this intuition is lost in the detailed proof of the algorithm's correctness. In the approach studied by this thesis, the algorithm is first modeled at a very high level of abstraction, and the correctness of this algorithm proven using the high-level, intuitive arguments of the algorithm's designers. The algorithm is then modeled at successively lower levels of abstraction, and each level shown to simulate the algorithm at the preceding level of abstraction. Since in the proof of simulation any property that has already been proven for preceding levels, the high-level intuition used by the algorithm's designers actually becomes a part of a rigorous proof of the detailed algorithm's correctness. One important contribution of this thesis is the introduction of a new model of computation in asynchronous networks, the *input-output automaton*, a general model of computation retaining the clean, automata-theoretic semantics abandoned by other models such as CSP.

Mark has also completed work with Yoram Moses entitled "Programming Simultaneous Actions Using Common Knowledge." This work applies the theory of knowledge in distributed systems to the design of efficient fault-tolerant protocols. This work defines a large class of problems requiring coordinated, simultaneous action in synchronous systems, and gives a method of transforming specifications of such problems into protocols that are *optimal in all runs*: these protocols are guaranteed to perform the simultaneous actions as soon as any other protocol could possibly perform them, given the input to the system and pattern of faulty processor behavior. (In contrast, most protocols do not adapt their behavior on the basis of faulty processor behavior and hence always perform only as well as they do in their worst-case runs.) This transformation is performed in two steps. The first step extracts directly from the problem specification a high-level protocol programmed using explicit tests for common knowledge. The second step consists of a careful analysis of when facts become common knowledge, thereby providing a method of efficiently implementing these protocols in many variants of the omissions failure model. In the generalized omissions model, however, this analysis shows that testing for common knowledge is NP-hard. Given the close correspondence between common knowledge and simultaneous actions, it is possible to show that no optimal protocol for any such problem can be computationally efficient in this model. The analysis in this paper exposes many subtle differences between the variants of the omissions failure model, including the precise point at which this gap in complexity occurs.

Paul Vitanyi

Lower Time Bounds for Simulating One Type of Storage by Another

Using Kolmogorov complexity, Dr. Paul Vitanyi derived optimal or nearly optimal lower bounds on the time to simulate more X by one Y, where X and Y can be tapes, stacks or queues. The derived lower bounds are typically within a log n factor of the upper bound, or match the upper bound.

Interaction between Algorithms and Model of Computation

Traditionally, computational complexity theory deals with sequential computations. In the computational models the underlying physics is hardly accounted for. In the area of parallel/distributed computing physical space, time and physics are much more important. He obtained new lower bounds on the average interconnect length in d-dimensional space embeddings of symmetric circuits (like the binary n-cube), which hold for multiprocessor models in any technology.

Distributed Control

The number of messages for matching pairs of mobile processes in a multiprocessor network is a measure for the cost of setting up temporary communication between such processes. We establish lower bounds on the average number of point-to-point transmissions between any pair of nodes in this context. Applications of the results include lower bounds on the number of messages for distributed nameserver, distributed mutual exclusion and generalizations to distributed s-matching (that is, matching a group of s processes) and distributed s-mutual exclusion (that is, s-1 processes may enter a critical section simultaneously, but s processes may not) for any $s>=2$.

Jennifer Welch

Jennifer Lundelius Welch has been investigating the modular description of distributed algorithms. She studied two existing resource allocation algorithms, and by using modularity obtained new algorithms that are more efficient. Modularity is also being used to rigorously prove correct, using a lattice-style decomposition, a network minimum spanning tree algorithm, whose correctness had not previously been formally given.

Resource Allocation Algorithms

Many resource allocation algorithms presented in the literature [21] [3] are described as using a solution to another problem as a "subroutine," yet the description employs a particular solution, with the details of its implementation inextricably embedded in the algorithm. The use of the word "subroutine" suggests attempting to describe such algorithms modularly, so that any solution at all to the sub-problem can be used. Welch and Nancy Lynch have used the I/O automaton model of Lynch and Tuttle [17]

to describe two resource allocation algorithms, one for mutual exclusion and one for drinking philosophers. In each case, the description uses a solution to another problem as a true subroutine; nothing needs to be known about the implementation, only that its "interface" is correct. The advantages of this approach are that the algorithms are easier to describe and prove correct. As an added bonus, one can sometimes obtain more efficient (in some complexity measure) solutions by using a more efficient subroutine. In fact, a faster drinking philosophers algorithm was obtained with this method.

Liveness conditions have in the past not been treated formally in many papers. Using the I/O automaton model, Welch is able to state precise correctness conditions, including liveness properties, for the resource allocation problems. The liveness properties of the subroutines are inherited by the mutual exclusion and drinking philosophers algorithms.

Mutual Exclusion

The mutual exclusion problem [5] is an abstraction of many problems that occur in operating systems and other computer applications. Each process in an asynchronous system has a section of its code singled out as its *critical region*. The mutual exclusion problem consists of guaranteeing that if some process is executing in its critical region, then no other process is in its critical region. In an operating system context, the critical region could represent access to a shared resource, such as a printer. In order to rule out a useless solution to the problem, in which no process ever enters its critical region, a solution must satisfy the *no-deadlock* property: as long as some process wants to enter its critical region, some process (but not necessarily the original one) does so. Sometimes the stronger property of *no-lockout* [14] is desired: any process that wants to enter its critical region eventually does so.

Welch describes the mutual exclusion tournament of Peterson and Fischer [21] in a completely modular way, so that any 2-process solution at all can be used as a subroutine. All that is required of the subroutine is that it satisfy the definition of a mutual exclusion algorithm. In addition, desirable properties of the subroutine carry over to the entire tournament solution --- if the subroutine has no-deadlock, then so does the tournament, and the same for no-lockout. (Unfortunately, the nature of the tournament precludes guaranteeing only a bounded number of bypasses.)

Another pleasing aspect of her approach is its efficient time performance --- the time a process spends waiting to enter its critical region is O(n log n), assuming no process is bypassed more than once in the 2-process subroutines. The asynchronous time measure used is essentially to assume a constant upper bound but no lower bound on the time between process steps.

Drinking Philosophers

(joint work with Nancy Lynch)

The mutual exclusion algorithm may be generalized in many ways to other resource allocation paradigms. One that has been much studied is the dining philosophers problem [6] [20] [22], in which each process (philosopher) in the system periodically requests a fixed set of resources (forks). In order to perform useful work (eat), a process must have exclusive access simultaneously to all the resources in its set. Chandy and Misra [3] generalized this to a more dynamic problem, the drinking philosophers problem, in which for each process, there is a maximum set of resources that it can request, and each time a process wishes to do some work, it may request an arbitrary subset of its maximum set.

In [3] a dining philosophers algorithm is given and proved correct, and then a drinking philosophers algorithm is given, which uses as a subroutine the aforementioned dining philosophers algorithm. Using the I/O automaton model, Lynch and Welch give a modular description of this drinking philosophers algorithm that uses any dining philosophers algorithm as a subroutine. As in the work on mutual exclusion described above, precise correctness conditions, including liveness properties such as no-deadlock and no-lockout, are given, and it is shown that the liveness properties of the dining philosophers algorithm carry over to the drinking philosophers algorithm. The maximum waiting time for a drinking philosopher to enter its critical region is the maximum time for a dining philosopher to enter its critical region. Thus, by replacing the dining philosophers algorithm of [3], which has waiting time O(n), with a dining philosophers algorithm such as that of Lynch [20], which has waiting time O(1), a more efficient drinking philosophers algorithm is obtained.

Minimum Spanning Tree Algorithm

(joint work with Leslie Lamport and Nancy Lynch)

The existence of a spanning tree in a distributed system, where processors correspond to nodes of a graph and communication links to edges, greatly simplifies many necessary tasks, such as routing and flow control. If the sum of the weights of the edges in the spanning tree is a minimum, where the weight assigned to each edge is application-dependent, then greater efficiency can result. In [10], Gallager, Humblet and Spira present a minimum spanning tree algorithm that is complex and delicate, but has no proof of correctness. The goal of the work described here is to provide a rigorous proof of correctness that corresponds to one's intuition concerning the workings of the algorithm.

The approach Lamport, Lynch and Welch are taking to proving correctness of this algorithm is to describe it at different levels of abstraction. The novel aspect is that these different descriptions do not form the usual chain hierarchy, but instead are organized into a lattice. Thus the original algorithm, or implementation, is described by several more abstract algorithms, each expanding on a different aspect of the implementation, and incomparable to each other. The completed work will present a framework of definitions and theorems to show the relationships between algorithms in

the lattice and to justify why these relationships imply the correctness of the implementation.

References

1. Aspnes, J. "Timestamp Ordering and Nested Transactions." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

2. Awerbuch, B., L. Kirousis, and E. Kranakis. "On Proving Register Atomicity." CWI Technical Report and submitted to a journal.

3. Chandy, K.M. and J. Misra. "The Drinking Philosophers Problem." *ACM Transactions on Programming Languages and Systems*, 6, 4, (1984), 632-646.

4. Coan, B. "Achieving Consensus in Fault-Tolerant Distributed Computer Systems: Protocols, Lower Bounds and Simulations." Ph. D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

5. Dijkstra, E.W. "Solution of a Problem in Concurrent Programming Control." *Communications of the ACM*, 8, (1965), 569.

6. Dijkstra, E.W. "Hierarchical Ordering of Sequential Processes." *Acta Informatica*, 1, (1971), 115-138.

7. Fekete, A. and N. Lynch, M. Merritt and W. Weihl. "Nested Transactions and Read/Write Locking." *Proceedings of the 6th ACM Symposium on Principles of Database Systems*, San Diego, CA, March 1987, 97-111. Expanded as MIT/LCS/TM-324, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

8. Fekete, A. and N. Lynch, M. Merritt and W. Weihl. "Nested Transactions and Dynamic Atomicity." In preparation.

9. Fekete, A., N. Lynch and L. Shrira. "A Modular Proof of Correctness for a Network Synchronizer." *Proceedings of the 2nd International Workshop on Distributed Algorithms*, Amsterdam, July 1987, to appear.

10. Gallager, R.G., P.A. Humblet and P.M. Spira. "A Distributed Algorithm for Minimum-Weight Spanning Trees." *ACM Transactions on Programming Languages and Systems*, 5, 1, (1983), 66-77.

11. Gerth, R. and L. Shrira "Communication Closed Layers: an Optimized Approach." *Proceedings of 6th Conference on Foundations of Software Technology & Theoretical Computer Science*, Delhi, India, December 1986.

12. Goldman, K.J. and N.A. Lynch. "Quorum Consensus in Nested Transaction Systems." *Proceedings 6th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing,* Vancouver, British Columbia, Canada, August 1987, to appear.

13. Herlihy, M., A. Lynch, M. Merritt and W. Weihl. "On the Correctness of Orphan Elimination Algorithms." *Proceedings of 17th International Symposium on Fault-Tolerant Computing,* Pittsburgh, PA, July 1987, to appear. Also, MIT/LCS/TM-329, MIT Laboratory for Computer Science, Cambridge, MA, Cambridge, MA, May 1987.

14. Knuth, D.E. "Additional Comments on a Problem in Concurrent Programming Control." *Communications of the ACM*, 9, 321, (1966).

15. Ladin, R., B. Liskov and L. Shrira. "A Technique for Constructing Highly-Available Services." Accepted in *Algorithmica.*

16. Lynch, N. and M. Merritt. "Introduction to the Theory of Nested Transactions." *Proceedings of International Conference on Database Theory,* Rome, Italy, September 1986, 278-305. Also MIT/LCS/TR-367, MIT Laboratory for Computer Science, Cambridge, MA, July 1986, to appear.

17. Lynch, N.A. and M.R. Tuttle. "Hierarchical Correctness Proofs for Distributed Algorithms." *Proceedings of 6th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing,* Vancouver, British Columbia, Canada, August 1987, to appear. Also MIT/LCS/TR-387, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

18. Moses, Y. and M.R. Tuttle. "Programming Simultaneous Actions using Common Knowledge." *Proceedings of the 27th Annual Symposium on Foundations of Computer Science,* Toronto, Ontario, Canada, October 1986, 208-221. Also to appear in *Algorithmica.*

19. Liskov, B. and L. Shrira. "Promises and Forks." Common System Design Note, MIT Laboratory of Computer Science, Cambridge, MA, June 1987.

20. Lynch, N. "Upper Bounds for Static Resource Allocation: A Distributed System." *JCSS*, 23, 2, (1981), 254-278.

21. Peterson, G.L. and M.J. Fischer. "Economical Solutions for the Critical Section Problem in a Distributed System." *Proceedings of the ACM Symposium on Theory of Computing*, (1977), 91-97.

22. Rabin, M. and D. Lehmann. "On the Advantages of Free Choice: A Symmetric and Fully Distributed Solution to the Dining Philosophers Problem." *Proceedings 8th ACM Symposium on Principles of Programming Languages*, 1981, 133-138.

23. Reed, D. "Naming and Synchronization In A Decentralized Computer System." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, September 1978. Also MIT/LCS/TR-205.

24. Welch, J.L. "Simulating Synchronous Processors." to appear in *Information and Computation.*

Publications

1. Bloom, B. "Constructing Two-Writer Atomic Registers." *Proceedings of 6th Symposium on Principles of Distributed Computing,* Vancouver, British Columbia, Canada, August 1987, to appear.

2. Coan, B. "A Communication-Efficient Canonical Form for Fault-Tolerant Distributed Protocols." *Proceedings of the 5th ACM Symposium on Principles of Distributed Computing* Calgary, Alberta, Canada, August 1986, 63-72.

3. Coan, B., B. Oki and E. Kolodner. "Limitations on Database Availability when Networks Partition." *Proceedings of the 5th ACM Symposium on Principles of Distributed Computing.* Calgary, Alberta, Canada, August 1986, 187-194.

4. Dwork, D. and Moses, Y. "Knowledge and Common Knowledge in a Byzantine Environment: Crash Failures." MIT/LCS/TM-300, Laboratory for Computer Science, Cambridge, MA, July 1986, to appear in the *Journal Information and Communication.*

5. Fekete, A. "Asynchronous Approximate Agreement." *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing*, to appear.

6. Fekete, A. "Asymptotically Optimal Algorithms for Approximate Agreement." *Proceedings of the 5th ACM Symposium on Principles of Distributed Computing,* Calgary, Alberta, Canada, August 1986, 73-87. Expanded and submitted to *Distributed Computing.*

7. Fekete, A. and N. Lynch, M. Merritt and W. Weihl. "Nested Transactions and Read/Write Locking." *Proceedings of the 6th ACM Symposium on*

Principles of Database Systems, San Diego, CA, March 1987, 97-111. Expanded as MIT/LCS/TM-324, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

8. Fekete, A. and N. Lynch, M. Merritt and W. Weihl. "Nested Transactions and Dynamic Atomicity." In preparation.

9. Fekete, A., N. Lynch and L. Shrira. "A Modular Proof of Correctness for a Network Synchronizer." *Proceedings of the 2nd International Workshop on Distributed Algorithms*, Amsterdam, July 1987, to appear.

10. Gerth, R. and L. Shrira "Communication Closed Layers: an Optimized Approach." *Proceedings of 6th Conference on Foundations of Software Technology & Theoretical Computer Science*, Delhi, India, December 1986.

11. Goldman, K.J. and N.A. Lynch. "Quorum Consensus in Nested Transaction Systems." *Proceedings 6th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, Vancouver, British Columbia, Canada, August 1987, to appear.

12. Herlihy, M., A. Lynch, M. Merritt and W. Weihl. "On the Correctness of Orphan Elimination Algorithms." *Proceedings of 17th International Symposium on Fault-Tolerant Computing*, Pittsburgh, PA, July 1987, to appear. Also, MIT/LCS/TM-329, MIT Laboratory for Computer Science, Cambridge, MA, Cambridge, MA, May 1987.

13. Kirousis, L., E. Kranakis and P.M.B. Vitanyi. "Atomic Multireader Register." *2nd International Workshop on Distributed Computing*, Amsterdam, July 1987.

14. Kranakis, E. and P.M.B. Vitanyi. "Distributed Control in Computer Networks and Cross-Sections of Multidimensional Bodies." MIT/LCS/TM-304, MIT Laboratory for Computer Science, Cambridge, MA, March 1986.

15. Li, M. and P.M.B. Vitanyi. "Tape versus Queue and Stacks: The Lower Bounds." September, 1986 version, Revision of February, 1986 version, Revision of September, 1985 version of this paper. (Submitted to *Information and Control*).

16. Li, M., L. Longpre and P.M.B. Vitanyi. "The Power of the Queue." *Structure in Complexity Theory Conference*, Lecture Notes in Computer Science 223, Springer-Verlag, Berlin, 1986, 219-233. Also MIT/LCS/TM-303, MIT Laboratory for Computer Science, Cambridge, MA, April 1986.

17. Lynch, N. and M. Merritt. "Introduction to the Theory of Nested Transactions." *Proceedings of International Conference on Database Theory.* Rome, Italy, September 1986, 278-305. Also MIT/LCS/TR-367, MIT Laboratory for Computer Science, Cambridge, MA, July 1986, to appear.

18. Lynch, N.A. and M.R. Tuttle. "Hierarchical Correctness Proofs for Distributed Algorithms." *Proceedings of 6th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing,* Vancouver, British Columbia, Canada, August 1987, to appear. Also MIT/LCS/TR-387, MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

19. Moses, Y. and M.R. Tuttle. "Programming Simultaneous Actions using Common Knowledge." *Proceedings of the 27th Annual Symposium on Foundations of Computer Science,* Toronto, Ontario, Canada, October 1986, 208-221. Also to appear in *Algorithmica.*

20. Mullender, S.J. and P.M.B. Vitanyi. "Distributed Match-Making, (Invited for *Algorithmica*, Special Issue on Distributed Computing)." To appear.

21. Vitanyi, P.M.B. "Non-Sequential Computation and Laws of Nature (Invited Lecture)." *Proceedings Aegean Workshop on Computing, VLSI Algorithms and Architectures (2nd International Workshop on Parallel Processing and VLSI).* Lecture Notes In Computer Science 227, Springer-Verlag, 1986, 108-120. Also MIT/LCS/TM-306, MIT Laboratory for Computer Science, Cambridge, MA, May 1986

22. Vitanyi, P.M.B. "Archirithmics or Algotecture?" In: *Mathematics and Computer Science II,* M. Hazewinkel, et al. (eds.), 139-161, North-Holland, Amsterdam, 1986.

23. Vitanyi, P.M.B. "Locality, Communication and Interconnect Length in Multicomputers." CWI Technical Report and submitted to a journal.

24. Vitanyi, P.M.B. and B. Awerbuch. "Atomic Shared Register Access by Asynchronous Hardware." *27th Annual IEEE Symposium on Theory of Computing*, Toronto, 1986. Also MIT/LCS/TM-322, MIT Laboratory for Computer Science, Cambridge, MA.

25. Welch, J.L. "Simulating Synchronous Processors." to appear in *Information and Computation.*

Theses Completed

1. Aspnes, J. "Timestamp Ordering and Nested Transactions." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987.

2. Coan, B. "Achieving Consensus in Fault-Tolerant Distributed Computer Systems: Protocols, Lower Bounds and Simulations." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

3. DiPesa Jr., A.P. "Real-Time Extensions To A Relational Database." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

4. Troxel, G. "Detection of and Recovery from Deadlock in a System using Remote Procedure Calls." S.B. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, June 1987.

5. Goldman, K.J. "Data Replication in Nested Transaction Systems." S.M. thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 1987. Also MIT/LCS/TR-390, MIT Laboratory for Computer Science, Cambridge, MA, May 1987.

6. Tuttle, M. "Correctness Proofs for Distributed Algorithms." S.M. thesis, MIT Department of Electrical Engineering and and Computer Science, Cambridge, MA, April 1987. Also MIT/LCS/TR-387 MIT Laboratory for Computer Science, Cambridge, MA, April 1987.

Theses in Progress

1. Fekete, A. "Topics In Distributed Algorithms." Ph.D. dissertation, Harvard University, Department of Mathematics, Cambridge, MA.

2. Tuttle, M. "Knowledge in Distributed Systems." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected June 1991.

3. Welch, J. "Topics in Distributed Computing: The Impact of Partial Synchrony and Modular Decomposition of Algorithms." Ph.D. dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, expected September 1987.

Talks

1. Coan, B. "A Communication-Efficient Canonical Form for Fault-Tolerant Distributed Protocols." 5th ACM Symposium on Principles of Distributed Computing, Calgary, Alberta, Canada, August 1986.

2. Coan, B. "A Compiler that Increases the Fault-Tolerance of Asynchronous Protocols." Cornell University, Ithaca, NY, March 1987.

3. Fekete, A. "Asymptotically Optimal Algorithms For Approximate Agreement." 5th ACM Symposium on Principles of Distributed Computing, Calgary, Alberta, Canada, August 1986.

4. Fekete, A."Nested Transactions and Read-Write Locking."
   Macquarie University, Sydney, Australia, January 1987;
   6th ACM Symposium on Principles of Database Systems,
   San Diego, CA, March 1987.

5. Lynch, N. "Correctness Conditions for Highly Available Replicated Databases." 5th ACM Symposium on Distributed Computing, Calgary, Alberta, Canada, August 1986.

6. Lynch, N. "Hierarchical Proofs for Distributed Algorithms." Workshop on Communication and Distributed Computing, October 1986.

7. Lynch, N. "The Theory of Nested Transactions." 1987.
   University of California, Irvine;
   University of California, Los Angeles;
   Computer Corporation of America;
   Brown University;
   Washington University.

8. Moses, Y. "Programming Simultaneous Actions using Common Knowledge."
   MIT Laboratory for Computer Science, Theory Seminar,
   August 1986;
   University of Toronto, November 1986;
   Mathematical Center, Amsterdam, The Netherlands,
   December 1986.

9. Shrira, L. "On the Complexity of Global Computation in the Presence of Faults."

Computer Science Department, Eindhoven University of Technology;
Computer Science Department, University of Utrecht, PODC, Calgary.

10. Shrira, L. "High Level Fault Tolerant Communication Primitives."
CWI, Amsterdam;
DEC SRC, Palo Alto;
Computer Science Department, CMU, Pittsburgh, PA.

11. Tuttle, M. "Programming Simultaneous Actions using Common Knowledge." 27th Annual Symposium on Foundations of Computer Science, Toronto, Ontario, Canada, October 1986.

12. Vitanyi, P. "Non-Sequential Computation and Laws of Nature."
University of California at San Diego, Department of EECS, Computer Science Seminar, May 27, 1986;
Proceedings Aegean Workshop on Computing, VLSI Algorithms and Architectures (2nd International Workshop on Parallel Processing and VLSI), Loutraki-Corinth, Greece, July 1986.

13. Vitanyi, P. "Archirithmics or Algotecture?" (Invited Lecture), Symposium on Fundamental Contribution to Mathematics and Computer Science in The Netherlands since 1945, Amsterdam, October 6, 1986.

14. Vitanyi, P. "Distributed Match-Making for Mobile Processes in Computer Networks."
Catholic University of Nijmegen (The Netherlands), Computer Science Department, Computer Science Seminar, April 22, 1986;
Laboratoire d'Informatique, Ecole Normale Superieure, Paris, France, December 2, 1986.

15. Vitanyi, P. "Atomic Shared Register Access by Asynchronous Hardware."

Computer Science Seminar, AT&T Bell Labs, Holmdel, NJ, September 4, 1986;
Computer Science Seminar, MIT Laboratory for Computer Science, Cambridge, MA, September 10, 1986;
Computer Science Colloquium, Cornell University, September 11, 1986;
27th Annual IEEE Symposium on Theory of Computing, Toronto, October 28, 1986;
Computer Science Seminar, University of Rochester, October 30, 1986;
CWI General Colloquium, Centrum voor Wiskunde en Informatica, Amsterdam, November 24, 1986.

16. Welch, J. "Transaction Commit in a Realistic Fault Model."
5th ACM Symposium on Distributed Computing, Calgary, Alberta, Canada, August;
Harvard University, April 1987.

17. Welch, J. "The Power of Time in Distributed Systems." MITRE Corporation, September 1986. (overview of some results about clock synchronization and model simulations)

PUBLICATIONS

Technical Memoranda

TM-10[4] Jackson, J.N.
Interactive Design Coordination for the Building Industry, June 1970, AD 708-400

TM-11 Ward, P.W.
Description and Flow Chart of the PDP-7/9 Communications Package, July 1970; AD 711-379

TM-12 Graham, R.M.
File Management and Related Topics June 12, 1970, September 1970; AD 712-068

TM-13 Graham, R.M.
Use of High Level Languages for Systems Programming, September 1970; AD 711-965

TM-14 Vogt, C.M.
Suspension of Processes in a Multi-processing Computer System, September 1970; AD 713-989

TM-15 Zilles, S.N.
An Expansion of the Data Structuring Capabilities of PAL, October 1970; AD 720-761

TM-16 Bruere-Dawson, G.
Pseudo-Random Sequences, October 1970; AD 713-852

TM-17 Goodman, L.I.
Complexity Measures for Programming Languages, September 1971, AD 729-011

TM-18 Reprinted as TR-85

TM-19 Fenichel, R.R.
A New List-Tracing Algorithm, October 1970; AD 714-522

TM-20 Jones, T.L.
A Computer Model of Simple Forms of Learning, January 1971; AD 720-337

[4]TMs 1-9 were never issued.

PUBLICATIONS

TM-21 Goldstein, R.C.
The Substantive Use of Computers For Intellectual Activities, April 1971, AD 721-618

TM-22 Wells, D.M.
Transmission Of Information Between A Man-Machine Decision System And Its Environment, April 1971, AD 722-837

TM-23 Strnad, A.J.
The Relational Approach to the Management of Data Bases, April 1971; AD 721-619

TM-24 Goldstein, R.C. and Strnad, A.J.
The MacAIMS Data Management System, April 1971, AD 721-620

TM-25 Goldstein, R.C.
Helping People Think, April 1971, AD 721-998

TM-26 Iazeolla, G.G.
Modeling and Decomposition of Information Systems for Performance Evaluation, June 1971, AD 733-965

TM-27 Bagchi, A.
Economy of Descriptions and Minimal Indices, January 1972, AD 736-960

TM-28 Wong, R.
Construction Heuristics for Geometry and a Vector Algebra Representation of Geometry, June 1972, AD 743-487

TM-29 Hossley, R. and Rackoff, C.
The Emptiness Problem for Automata on Infinite Trees, Spring 1972; AD 747-250

TM-30 McCray, W.A.
SIM360: A S/360 Simulator, October 1972; AD 749-365

TM-31 Bonneau, R.J.
A Class of Finite Computation Structures Supporting the Fast Fourier Transform, March 1973; AD 757-787

TM-32 Moll, R.
An Operator Embedding Theorem for Complexity Classes of Recursive Functions, May 1973; AD 759-999

TM-33 Ferrante, J. and Rackoff, C.
A Decision Procedure for the First Order Theory of Real Addition with Order, May 1973; AD 760-000

TM-34 Bonneau, R.J.
Polynomial Exponentiation: The Fast Fourier Transform Revisited, June 1973, PB 221-742

TM-35 Bonneau, R.J.
An Interactive Implementation of the Todd-Coxeter Algorithm, December 1973; AD 770-565

TM-36 Geiger, S.P.
A User's Guide to the Macro Control Language, December 1973; AD 771-435

TM-37 Schonhage, A.
Real-Time Simulation of Multidimensional Turing Machines by Storage Modification Machines, December 1973; PB 226-103/AS

TM-38 Meyer, A.R.
Weak Monadic Second Order Theory of Successor Is Not Elementary-Recursive, December 1973; PB 226-514/AS

TM-39 Meyer, A.R.
Discrete Computation: Theory and Open Problems, January 1974; PB 226-836/AS

TM-40 Paterson, M.S., Fischer, M.J. and Meyer, A.R.
An Improved Overlap Argument for On-Line Multiplication, January 1974, AD 773-137

TM-41 Fischer, M.J. and Paterson, M.S.
String-Matching and Other Products, January 1974; AD 773-138

TM-42 Rackoff, C.
On the Complexity of the Theories of Weak Direct Products, January 1974, PB 228-459/AS

TM-43 Fischer, M.J. and Rabin, M.O.
Super-Exponential Complexity of Presburger Arithmetic, February 1974, AD 775-004

TM-44 Pless, V.
Symmetry Codes and their Invariant Subcodes, May 1974; AD 780-243

TM-45 Fischer, M.J. and Stockmeyer, L.J.
Fast On-Line Integer Multiplication, May 1974; AD 779-889

PUBLICATIONS

TM-46 Kedem, Z.M.
Combining Dimensionality and Rate of Growth Arguments for Establishing Lower Bounds on the Number of Multiplications, June 1974, PB 232-969/AS

TM-47 Pless, V.
Mathematical Foundations of Flip-Flops, June 1974, AD 780-901

TM-48 Kedem, Z.M.
The Reduction Method for Establishing Lower Bounds on the Number of Additions, June 1974, PB 233-538/AS

TM-49 Pless, V.
Complete Classification of (24,12) and (22,11) Self-Dual Codes, June 1974, AD 781-335

TM-50 Benedict, G.G.
An Enciphering Module for Multics, S.B. Thesis, EE Department, July 1974, AD 782-658

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 236-815/AS

TM-52 Lind, J.C.
Computing in Logarithmic Space, September 1974, PB 236-167/AS

TM-53 Bengelloun, S.A.
MDC-Programmer: A Muddle-to-Datalanguage Translator for Information Retrieval, S.B. Thesis, EE Department, October 1974, AD 786-754

TM-54 Meyer, A.R.
The Inherent Computation Complexity of Theories of Ordered Sets: A Brief Survey, October 1974, PB 237-200/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 237-206/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 237-033/AS

TM-59 Hack, M.
Decision Problems for Petri Nets and Vector Addition Systems, March 1975; PB 231-916/AS

TM-60 Weiss, R.B.
CAMAC: Group Manipulation System, March 1975, PB 240-495/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, AD A010-217

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.
Formal Properties of Well-Formed Data Flow Schemas, S.B., S.M. & E.E. Thesis, EE Department, June 1975

TM-67 Cardoza, E.E.
Computational Complexity of the Word 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

PUBLICATIONS

TM-70 Ruth, G.R.
Automatic Design of Data Processing Systems, February 1976; AD A023-451

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; AD A026-912

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; AD A030-418

TM-75 Yao, A.C. and Rivest, R.I.
K+1 Heads are Better than K, September 1976; AD A030-008

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; AD A030-402

TM-78 Benjamin, A.J.
Improving Information Storage Reliability Using a Data Network, S.M. Thesis, EE & CS Department, October 1976; AD A033-394

TM-79 Brown, G.P.
A System to Process Dialogue: A Progress Report, October 1976; AD A033-276

TM-80 Even, S.
The Max Flow Algorithm of Dinic and Karzanov: An Exposition, December 1976

TM-81 Gifford, D.K.
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.
A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, April 1977; AD A039-036

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; AD A040-698

TM-86 Szolovits, P., Hawkinson, L.B. and Martin, W.A.
An Overview of OWL, A Language for Knowledge Representation, June 1977; AD A041-372

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. Jr.
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; AD A045-526

TM-91 Shamir, A.
Factoring Numbers in O(log n) Arithmetic Steps, November 1977; AD A047-709

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

PUBLICATIONS

TM-94 Berez, J.M.
A Dynamic Debugging System for MDL, S.B. Thesis, EE & CS Department, January 1978; AD A050-191

TM-95 Harel, D.
Characterizing Second Order Logic with First Order Quantifiers, February 1978

TM-96 Harel, D., Amir P. 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; AD A052-538

TM-101 Martin, W.A.
Descriptions and the Specialization of Concepts, March 1978; AD A052-773

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; AD A057-152

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; AD A061-222

TM-115 Bryant, R.E. and Dennis, J.B.
Concurrent Programming, October 1978; AD A061-180

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

PUBLICATIONS

TM-118	Borkin, S.A. Data Model Equivalence, December 1978; AD A062-753
TM-119	Shamir, A. and Zippel, R.E. On the Security of the Merkle-Hellman Cryptographic Scheme, December 1978; AD A063-104
TM-120	Brock, J.D. Operational Semantics of a Data Flow Language, S.M. Thesis, EE & CS Department, December 1978; AD A062-997
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; AD A063-886
TM-125	Shamir, A., Rivest, R.L. and Adelman, L.M. Mental Poker, February 1979; AD A066-331
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; AD A067-972

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; AD A068-967

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; AD A069-397

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 (Revised May 1980), September 1979; AD A074-625

TM-140 Szolovits, P.
Artificial Intelligence and Clinical Problem Solving, September 1979

TM-141 Hammer, M. and McLeod, D.
On Database Management System Architecture, October 1979; AD A076-417

PUBLICATIONS

TM-142 Lipski, W., Jr.
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.
A $T = O(2^{n/2})$, $S = O(2^{n/4})$ Algorithm for Certain NP-Complete Problems, January 1980; AD A080-385

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" Branch-and-Bound Exploration Schema, January 1980; AD A079-912

TM-150 Meyer, A.R.
Ten Thousand and One Logics of Programming, February 1980

TM-151 Toffoli, T.
Reversible Computing, February 1980; AD A082-021

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 Atai, 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; AD A083-491

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; AD A084-456

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

PUBLICATIONS

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; AD A087-357
TM-169	Burke, G. and Moon, D. LOOP Iteration Macro, (revised January 1981) July 1980; AD A087-372
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; AD A090-130
TM-173	Hawkinson, L.B. XLMS: A Linguistic Memory System, September 1980; AD A090-033
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; AD A091-015
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 Database Concurrency Control, October 1980

TM-181 Lipski, W. Jr. 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; AD A092-625

TM-186 Szolovits, P. and Martin, W.A.
BRAND X Manual, November 1980; AD A093-041

TM-187 Fischer, M.J., Meyer, A.R. and Paterson, M.S.
Ω(n log n) Lower Bounds on Length of Boolean Formulas, November 1980

TM-188 Mayr, E.
An Effective Representation of the Reachability Set of Persistent Petri Nets, January 1981

TM-189 Mayr, E.
Persistence of Vector Replacement Systems is Decidable, January 1981

PUBLICATIONS

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. and Reif, J.
The Propositional Dynamic Logic of Deterministic Well-Structured Programs, March 1981

TM-199 Mayr, E. and Meyer, A.R.
The Complexity of the Word Problems for Communative 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.
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. and Pinter, R.
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.
The Traveling Salesman Problem with Many Visits to Few Cities, November 1981

TM-209 Johnson, D. and Papadimitriou, C.
Computational Complexity and the Traveling Salesman Problem, December 1981

TM-210 Greif, I.
Software for the 'Roiels' 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

PUBLICATIONS

TM-214	Meyer, A. and Mitchell, J. Terminations for Recursive Programs: Completeness and Axiomatic Definability, March 1982
TM-215	Leiserson, C. and Saxe J. Optimizing Synchronous Systems, March 1982
TM-216	Church, K. and Patil, R. Coping with Syntactic Ambiguity or How to Put the Block in the Box on the Table, April 1982.
TM-217	Wright, 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-Times Elements for NMOS VLSI Systems, May 1982
TM-221	Leighton, F., Lepley, M. and Miller, G. 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's 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. and Zachos, S.
Two Remarks on the Power of Counting, August 1982

TM-229 Cosmadakis, 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.
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.
Implementing Internet Remote Login on a Personal Computer, S.B. Thesis, Electrical Engineering Department, December 1982

TM-234 Rivest, R. and Sherman, A.
Randomized Encryption Techniques, January 1983.

TM-235 Mitchell, J.
The Implication 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

PUBLICATIONS

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; AD A134239
TM-242	Pingali, K. and Arvind. Efficient Demand-driven Evaluation (I), September 1983; AD A133477
TM-243	Pingali, K. and Arvind. Efficient Demand-driven Evaluation (II), September 1983; AD A133879
TM-244	Goldreich, O., Goldwasser, S. and Micali, S. How to Construct Random Functions, November 1983
TM-245	Meyer, A. 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., 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; AD A140638

TM-254 Feldmeier, D.C.
Empirical Analysis of a Token Ring Network, April 1984; AD A138905

TM-255 Bhatt, S. and Leiserson, C.
How to Assemble Tree Machines, April 1984; AD 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., April 1984

TM-257 Dwork, C., Kanellakis, P. and Mitchell, J.
On the Sequential Nature of Unification, April 1984; AD A139939

TM-258 Halpern, M., Meyer A. and Trakhtenbrot, B.
The Semantics of Local Storage, or What Makes the Free-list Free?, April 1984

TM-259 Lynch, N. and Fredrickson, G.
The Impact of Synchronous Communication on the Problem of Electing a Leader in a Ring, April 1984

TM-260 Chor, B. and Goldreich, O.
RSA/Rabin Least Significant Bits are $1/2 + 1/\text{poly}(\log N)$ Secure, May 1984

TM-261 Zaks, S.
Optimal Distributed Algorithms for Sorting and Ranking, May 1984

PUBLICATIONS

TM-262 Leighton, T. and Rosenberg, A.
Three-dimensional Circuit Layouts, June 1984; AD A143430

TM-263 Sirbu, M.S. and Sutherland, J.B.
Naming and Directory Issues in Message Transfer Systems, July 1984; AD A154727

TM-264 Sarin, S.K. and Greif, I.
Software for Interactive On-Line Conferences, July 1984; AD A154742

TM-265 Lundelius, J. and Lynch, N.
A New Fault-Tolerant Algorithm for Clock Synchronization, July 1984; AD A154771

TM-266 Chor, B. and Coan, B.A.
A Simple and Efficient Randomized Byzantine Agreement Algorithm, August 1984; AD A153240

TM-267 Schooler, R. and Stamos, J.W.
Proposal for a Small Scheme Implementation, October 1984; AD A148707

TM-268 Awerbuch, B.
Complexity of Network Synchronization, January 1985

TM-269 Fisher, M., Lynch, N.A., Burns, J. and Borodin, A.
The Colored Ticket Algorithm, August 1983; AD A148696

TM-270 Dwork, C., Lynch, C. and Stockmeyer, L.
Consensus in the Presence of Partial Synchrony (Preliminary Version), July 1984; AD 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; AD A154726

TM-273 Berman, F., Leighton, T., Shor, P.W. and Shor, L.
Generalized Planar Matching, April 1985; AD 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; AD A154809

TM-276 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; AD A156541

TM-277 Frederickson, G.N. and Lynch, N.A.
A General Lower Bound for Electing a Leader in a Ring, March 1985; AD 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; AD A157553

TM-279 Fischer, M.J., Lynch, N.A. and Merritt, M.
Easy Impossibility Proofs for Distributed Consensus Problems, June 1985; AD 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; AD A159496

TM-285 Kilian, J.J
Two Undecidability Results in Probabilistic Automata Theory, S.B. Thesis, EE & CS Department, June 1985

PUBLICATIONS

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	Fisher, 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; AD 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 Recency-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; AD 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; AD A172224

TM-305 Sacks, E.
Representing Change, May 1986

TM-306 Vitanyi, P.M.B.
Nonsequential Computation and Laws of Nature, May 1986; AD 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; AD A172144

PUBLICATIONS

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; AD 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, E.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 (^ + 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; AD A178402

TM-322 Cormen, T.H.
Efficient Multichip Partial Concentrator Switches, February 1987; AD 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; AD A191981

TM-325 Awerbuch, B., Goldreich, O., and Vainish, R.
On the Message Complexity of Broadcast: A Basic Lower Bound, 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; AD A182175

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

Technical Reports

TR-1[5] Bobrow, D.G.
Natural Language Input for a Computer Problem Solving System, Ph.D. Dissertation, Math. Department, September 1964; AD 604-730

TR-2 Raphael, B.
SIR: A Computer Program for Semantic Information Retrieval, Ph.D. Dissertation, Math. Department, June 1964; AD 608-499

TR-3 Corbato, F.J.
System Requirements for Multiple-Access, Time-Shared Computers, May 1964; AD 608-501

TR-4 Ross, D.T. and Feldman, C.G.
Verbal and Graphical Language for the AED System: A Progress Report, May 1964; AD 604-678

TR-6 Biggs, J.M. and Logcher, R.D.
STRESS: A Problem-Oriented Language for Structural Engineering, May 1964; AD 604-679

TR-7 Weizenbaum, J.
OPL-1: An Open Ended Programming System within CTSS, April 1964; AD 604-680

TR-8 Greenberger, M.
The OPS-1 Manual, May 1964; AD 604-681

TR-11 Dennis, J.B.
Program Structure in a Multi-Access Computer, May 1964; AD 608-500

TR-12 Fano, R.M.
The MAC System: A Progress Report, October 1964; AD 609-296

TR-13 Greenberger, M.
A New Methodology for Computer Simulation, October 1964; AD 609-288

TR-14 Roos, D.
Use of CTSS in a Teaching Environment, November 1964; AD 661-807

[5]TRs 5, 9, 10, 15 were never issued

TR-16 Saltzer, J.H.
CTSS Technical Notes, March 1965; AD 612-702

TR-17 Samuel, A.L.
Time-Sharing on a Multiconsole Computer, March 1965; AD 462-158

TR-18 Scherr, A.L.
An Analysis of Time-Shared Computer Systems, Ph.D. Dissertation, EE Department, June 1965; AD 470-715

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 474-018

TR-20 Wantman, M.E.
CALCULAID: An On-Line System for Algebraic Computation and Analysis, S.M. Thesis, Sloan School, September 1965; AD 474-019

TR-21 Denning, P.J.
Queueing Models for File Memory Operation, S.M. Thesis, EE Department, October 1965; AD 624-943

TR-22 Greenberger, M.
The Priority Problem, November 1965; AD 625-728

TR-23 Dennis, J.B. and Van Horn, E.C.
Programming Semantics for Multi-programmed Computations, December 1965; AD 627-537

TR-24 Kaplow, R., Strong, S. and Brackett, J.
MAP: A System for On-Line Mathematical Analysis, January 1966; AD 476-443

TR-25 Stratton, W.D.
Investigation of an Analog Technique to Decrease Pen-Tracking Time in Computer Displays, S.M. Thesis, EE Department, March 1966; AD 631-396

TR-26 Cheek, T.B.
Design of a Low-Cost Character Generator for Remote Computer Displays, S.M. Thesis, EE Department, March 1966; AD 631-269

TR-27 Edwards, D.J.
OCAS - On-Line Cryptanalytic Aid System, S.M. Thesis, EE Department, May 1966; AD 633-678

PUBLICATIONS

TR-28 Smith, A.A.
Input/Output in Time-Shared, Segmented, Multiprocessor Systems, S.M. Thesis, EE Department, June 1966; AD 637-215

TR-29 Ivie, E.L.
Search Procedures Based on Measures of Relatedness between Documents, Ph.D. Dissertation, EE Department, June 1966; AD 636-275

TR-30 Saltzer, J.H. Traffic Control in a Multiplexed Computer System, Sc.D. Thesis, EE Department, July 1966; AD 635-966

TR-31 Smith, D.L.
Models and Data Structures for Digital Logic Simulation, S.M. Thesis, EE Department, August 1966; AD 637-192

TR-32 Teitelman, W.
PILOT: A Step Toward Man-Computer Symbiosis, Ph.D. Dissertation, Math. Department, September 1966; AD 638-446

TR-33 Norton, L.M. ADEPT - A Heuristic Program for Proving Theorems of Group Theory, Ph.D. Dissertation, Math. Department, October 1966; AD 645-660

TR-34 Van Horn, E.C., Jr.
Computer Design for Asynchronously Reproducible Multiprocessing, Ph.D. Dissertation, EE Department, November 1966; AD 650-407

TR-35 Fenichel, R.R.
An On-Line System for Algebraic Manipulation, Ph.D. Dissertation, Appl. Math. (Harvard), December 1966; AD 657-282

TR-36 Martin, W.A.
Symbolic Mathematical Laboratory, Ph.D. Dissertation, EE Department, January 1967; AD 657-283

TR-37 Guzman-Arenas, A.
Some Aspects of Pattern Recognition by Computer, S.M. Thesis, EE Department, February 1967; AD 656-041

TR-38 Rosenberg, R.C., Kennedy, D.W. and Humphrey, R.A.
A Low-Cost Output Terminal For Time-Shared Computers, March 1967; AD 662-027

TR-39 Forte, A.
Syntax-Based Analytic Reading of Musical Scores, April 1967; AD 661-806

TR-40 Miller, J.R.
On-Line Analysis for Social Scientists, May 1967; AD 668-009

TR-41 Coons, S.A.
Surfaces for Computer-Aided Design of Space Forms, June 1967; AD 663-504

TR-42 Liu, C.L., Chang, G.D. and Marks, R.E.
Design and Implementation of a Table-Driven Compiler System, July 1967; AD 668-960

TR-43 Wilde, D.U.
Program Analysis by Digital Computer, Ph.D. Dissertation, EE Department, August 1967; AD 662-224

TR-44 Gorry, G.A.
A System for Computer-Aided Diagnosis, Ph.D. Dissertation, Sloan School, September 1967; AD 662-665

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 663-502

TR-46 Alsop, J.W.
A Canonic Translator, S.B. Thesis, EE Department, November 1967; AD 663-503

TR-47 Moses, J.
Symbolic Integration, Ph.D. Dissertation, Math. Department, December 1967; AD 662-666

TR-48 Jones, M.M.
Incremental Simulation on a Time-Shared Computer, Ph.D. Dissertation, Sloan School, January 1968; AD 662-225

TR-49 Luconi, F.L.
Asynchronous Computational Structures, Ph.D. Dissertation, EE Department, February 1968; AD 667-602

TR-50 Denning, P.J.
Resource Allocation in Multiprocess Computer Systems, Ph.D. Dissertation, EE Department, May 1968; AD 675-554

TR-51 Charniak, E.
CARPS, A Program which Solves Calculus Word Problems, S.M. Thesis, EE Department, July 1968; AD 673-670

PUBLICATIONS

TR-52 Deitel, H.M.
Absentee Computations in a Multiple-Access Computer System, S.M. Thesis, EE Department, August 1968; AD 684-738

TR-53 Slutz, D.R.
The Flow Graph Schemata Model of Parallel Computation, Ph.D. Dissertation, EE Department, September 1968; AD 683-393

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 689-468

TR-55 Rappaport, R.L.
Implementing Multi-Process Primitives in a Multiplexed Computer System, S.M. Thesis, EE Department, November 1968; AD 689-469

TR-56 Thornhill, D.E., Stotz, R.H., Ross, D.T. and Ward, J.E.
An Integrated Hardware-Software System for Computer Graphics in Time-Sharing, December 1968; AD 685-202

TR-57 Morris, J.H.
Lambda-Calculus Models of Programming Languages, Ph.D. Dissertation, Sloan School, December 1968; AD 683-394

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 686-988

TR-59 Guzman; A.
Computer Recognition of Three- Dimensional Objects in a Visual Scene, Ph.D. Dissertation, EE Department, December 1968; AD 692-200

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 689-305

TR-61 Baecker, R.M.
Interactive Computer-Mediated Animation, Ph.D. Dissertation, EE Department, June 1969; AD 690-887

TR-62 Tillman, C.C., Jr.
EPS: An Interactive System for Solving Elliptic Boundary-Value Problems with Facilities for Data Manipulation and General-Purpose Computation, June 1969; AD 692-462

TR-63 Brackett, J.W., Hammer, M. and Thornhill, D.E.
Case Study in Interactive Graphics Programming: A Circuit Drawing and Editing Program for Use with a Storage-Tube Display Terminal, October 1969; AD 699-930

TR-64 Rodriguez, J.E.
A Graph Model for Parallel Computations, Sc.D. Thesis, EE Department, September 1969; AD 697-759

TR-65 DeRemer, F.L.
Practical Translators for LR(k) Languages, Ph.D. Dissertation, EE Department, October 1969; AD 699-501

TR-66 Beyer, W.T.
Recognition of Topological Invariants by Iterative Arrays, Ph.D. Dissertation, Math. Department, October 1969; AD 699-502

TR-67 Vanderbilt, D.H.
Controlled Information Sharing in a Computer Utility, Ph.D. Dissertation, EE Department, October 1969; AD 699-503

TR-68 Selwyn, L.L.
Economies of Scale in Computer Use: Initial Tests and Implications for The Computer Utility, Ph.D. Dissertation, Sloan School, June 1970; AD 710-011

TR-69 Gertz, J.L.
Hierarchical Associative Memories for Parallel Computation, Ph.D. Dissertation, EE Department, June 1970; AD 711-091

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 711-060

TR-71 Fiasconaro, J.G.
A Computer-Controlled Graphical Display Processor, S.M. Thesis, EE Department, June 1970; AD 710-479

TR-72 Patil, S.S.
Coordination of Asynchronous Events, Sc.D. Thesis, EE Department, June 1970; AD 711-763

TR-73
Griffith, A.K.
Computer Recognition of Prismatic Solids, Ph.D. Dissertation, Math. Department, August 1970; AD 712-069

TR-74 Edelberg, M.
Integral Convex Polyhedra and an Approach to Integralization, Ph.D. Dissertation, EE Department, August 1970; AD 712-070

TR-75 Hebalkar, P.G.
Deadlock-Free Sharing of Resources in Asynchronous Systems, Sc.D. Thesis, EE Department, September 1970; AD 713-139

TR-76 Winston, P.H.
Learning Structural Descriptions from Examples, Ph.D. Dissertation, EE Department, September 1970; AD 713-988

TR-77 Haggerty, J.P.
Complexity Measures for Language Recognition by Canonic Systems, S.M. Thesis, EE Department, October 1970; AD 715-134

TR-78 Madnick, S.E.
Design Strategies for File Systems, S.M. Thesis, EE Department & Sloan School, October 1970; AD 714-269

TR-79 Horn, B.K.
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 717-336

TR-80 Clark, D.D., Graham, R.M., Saltzer, J.H. and Schroeder, M.D
The Classroom Information and Computing Service, January 1971; AD 717-857

TR-81 Banks, E.R.
Information Processing and Transmission in Cellular Automata, Ph.D. Dissertation, ME Department, January 1971; AD 717-951

TR-82 Krakauer, L.J.
Computer Analysis of Visual Properties of Curved Objects, Ph.D. Dissertation, EE Department, May 1971; AD 723-647

TR-83 Lewin, D.E.
In-Process Manufacturing Quality Control, Ph.D. Dissertation, Sloan School, January 1971; AD 720-098

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 721-399

TR-85 Miller, P.L.
Automatic Creation of a Code Generator from a Machine Description, E.E. Thesis, EE Department, May 1971; AD 724-730

TR-86 Schell, R.R.
Dynamic Reconfiguration in a Modular Computer System, Ph.D. Dissertation, EE Department, June 1971; AD 725-859

TR-87 Thomas, R.H.
A Model for Process Representation and Synthesis, Ph.D. Dissertation, EE Department, June 1971; AD 726-049

TR-88 Welch, T.A.
Bounds on Information Retrieval Efficiency in Static File Structures, Ph.D. Dissertation, EE Department, June 1971; AD 725-429

TR-89 Owens, R.C., Jr.
Primary Access Control in Large-Scale Time-Shared Decision Systems, S.M. Thesis, Sloan School, July 1971; AD 728-036

TR-90 Lester, B.P.
Cost Analysis of Debugging Systems, S.B. & S.M. Thesis. EE Department, September 1971; AD 730-521

TR-91 Smoliar, S.W.
A Parallel Processing Model of Musical Structures, Ph.D. Dissertation, Math. Department, September 1971; AD 731-690

TR-92 Wang, P.S.
Evaluation of Definite Integrals by Symbolic Manipulation, Ph.D. Dissertation, Math. Department, October 1971; AD 732-005

TR-93 Greif, I.G.
Induction in Proofs about Programs, S.M. Thesis, EE Department, February 1972; AD 737-701

TR-94 Hack, M.H.T.
Analysis of Production Schemata by Petri Nets, S.M. Thesis, EE Department, February 1972; AD 740-320

PUBLICATIONS

TR-95 Fateman, R.J.
Essays in Algebraic Simplification (A revision of a Harvard Ph.D. Dissertation), April 1972; AD 740-132

TR-96 Manning, F.
Autonomous, Synchronous Counters Constructed Only of J-K Flip-Flops, S.M. Thesis, EE Department, May 1972; AD 744-030

TR-97 Vilfan, B.
The Complexity of Finite Functions, Ph.D. Dissertation, EE Department, March 1972; AD 739-678

TR-98 Stockmeyer, L.J.
Bounds on Polynomial Evaluation Algorithms, S.M. Thesis, EE Department, April 1972; AD 740-328

TR-99 Lynch, N.A.
Relativization of the Theory of Computational Complexity, Ph.D. Dissertation, Math. Department, June 1972; AD 744-032

TR-100 Mandl, R.
Further Results on Hierarchies of Canonic Systems, S.M. Thesis, EE Department, June 1972; AD 744-206

TR-101 Dennis, J.B.
On the Design and Specification of a Common Base Language, June 1972; AD 744-207

TR-102 Hossley, R.F.
Finite Tree Automata and Ω-Automata, S.M. Thesis, EE Department, September 1972; AD 749-367

TR-103 Sekino, A.
Performance Evaluation of Multiprogrammed Time-Shared Computer Systems, Ph.D. Dissertation, EE Department, September 1972; AD 749-949

TR-104 Schroeder, M.D.
Cooperation of Mutually Suspicious Subsystems in a Computer Utility, Ph.D. Dissertation, EE Department, September 1972; AD 750-173

TR-105 Smith, B.J.
An Analysis of Sorting Networks, Sc.D. Thesis, EE Department, October 1972; AD 751-614

TR-106 Rackoff, C.W.
The Emptiness and Complementation Problems for Automata on Infinite Trees, S.M. Thesis, EE Department, January 1973; AD 756-248

TR-107 Madnick, S.E.
Storage Hierarchy Systems, Ph.D. Dissertation, EE Department, April 1973; AD 760-001

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 767-730

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 771-430

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 774-141

PUBLICATIONS

TR-117 Clark, D.D.
An Input/Output Architecture for Virtual Memory Computer Systems, Ph.D. Dissertation, EE Department, January 1974; AD 774-738

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 775-545

TR-120 Ramchandani, C.
Analysis of Asynchronous Concurrent Systems by Timed Petri Nets, Ph.D. Dissertation, EE Department, February 1974; AD 775-618

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 775-546

TR-123 Saltzer, et al.
Introduction to Multics, February 1974; AD 918-562

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 778-688

TR-126 Altman, V.E.
A Language Implementation System, S.B. & S.M. Thesis, Sloan School, May 1974; AD 780-672

TR-127 Greenberg, B.S.
An Experimental Analysis of Program Reference Patterns in the Multics Virtual Memory, S.M. Thesis, EE Department, May 1974; AD 780-407

TR-128 Frankston, R.M.
The Computer Utility as a Marketplace for Computer Services, S.M. & EE Thesis, EE Department, May 1974; AD 780-436

TR-129 Weissberg, R.W.
Using Interactive Graphics in Simulating the Hospital Emergency Room, S.M. Thesis, EE Department, May 1974; AD 780-437

TR-130 Ruth, G.R.
Analysis of Algorithm Implementations, Ph.D. Dissertation, EE Department, May 1974; AD 780-408

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 781-305

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 787-795

TR-136 Ward, S.A.
Functional Domains of Applicative Languages, Ph.D. Dissertation, EE Department, September 1974; AD 787-796

TR-137 Seiferas, J.I.
Nondeterministic Time and Space Complexity Classes, Ph.D. Dissertation, Math. Department, September 1974.
PB 236-777/AS

TR-138 Yun, D.Y.Y.
The Hensel Lemma in Algebraic Manipulation, Ph.D. Dissertation, Math. Department, November 1974; AD A002-737

TR-139 Ferrante, J.
Some Upper and Lower Bounds on Decision Procedures in Logic, Ph.D. Dissertation, Math. Department, November 1974.
PB 238-121/AS

TR-140 Redell, D.D.
Naming and Protection in Extendable Operating Systems, Ph.D. Dissertation, EE Department, November 1974; AD A001-721

TR-141 Richards, M., Evans, A. and Mabee, R.
The BCPL Reference Manual, December 1974; AD A003-599

TR-142 Brown, G.P.
Some Problems in German to English Machine Translation, S.M. & E.E. Thesis, EE Department, December 1974; AD A003-002

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 Department, February 1975.

TR-145 Henderson, D.A.
The Binding Model: A Semantic Base for Modular Programming Systems, Ph.D. Dissertation, EE Department, February 1975; AD A006-961

TR-146 Malhotra, A.
Design Criteria for a Knowledge-Based English Language System for Management: An Experimental Analysis, Ph.D. Dissertation, EE Department, February 1975.

TR-147 Van De Vanter, M.L.
A Formalization and Correctness Proof of the CGOL Language System, S.M. Thesis, EE Department, March 1975.

TR-148 Johnson, J.
Program Restructuring for Virtual Memory Systems, Ph.D. Dissertation, EE Department, March 1975; AD A009-218

TR-149 Snyder, A.
A Portable Compiler for the Language C, S.B. & S.M. Thesis, EE Department, May 1975; AD A010-218

TR-150 Rumbaugh, J.E.
A Parallel Asynchronous Computer Architecture for Data Flow Programs, Ph.D. Dissertation, EE Department, May 1975; AD A010-918

TR-151 Manning, F.B.
Automatic Test, Configuration, and Repair of Cellular Arrays, Ph.D. Dissertation, EE Department, June 1975; AD A012-822

TR-152 Qualitz, J.E.
Equivalence Problems for Monadic Schemas, Ph.D. Dissertation, EE Department, June 1975; AD A012-823

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 A016-302

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 Computer 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 A018-997

TR-158 Grossman, R.W.
Some Data-base Applications of Constraint Expressions, S.M. Thesis, EE & CS Department, February 1976; AD A024-149

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 A026-688

PUBLICATIONS

TR-161 Hack, M.
Decidability Questions, Ph.D. Dissertation, EE & CS Department, June 1976.

TR-162 Kent, S.T.
Encryption-Based Protection Protocols for Interactive User-Computer Communication, S.M. Thesis, EE & CS Department, June 1976; AD A026-911

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.J.
High Level Expression of Semantic Integrity Specifications in a Relational Data Base System, S.M. Thesis, EE & CS Department, September 1976; AD A034-184

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 A034-185

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.R.
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 A035-397

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 A040-601

TR-179 Karger, P.A.
Non-Discretionary Access Control for Decentralized Computing Systems, S.M. Thesis, EE & CS Department, May 1977; AD A040-804

TR-180 Luniewski, A.W.
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 A042-370

PUBLICATIONS

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 A043-901

TR-185 Goldman, B.
Deadlock Detection in Computer Networks, S.B. & S.M. Thesis, EE & CS Department, September 1977; AD A047-025

TR-186 Ackerman, W.B.
A Structure Memory for Data Flow Computers, S.M. Thesis, EE & CS Department, September 1977; AD A047-026

TR-187 Long, W.J.
A Program Writer, Ph.D. Dissertation, EE & CS Department, November 1977; AD A047-595

TR-188 Bryant, R.E.
Simulation of Packet Communication Architecture Computer Systems, S.M. Thesis, EE & CS Department, November 1977; AD A048-290

TR-189 Ellis, D.J.
Formal Specifications for Packet Communication Systems, Ph.D. Dissertation, EE & CS Department, November 1977; AD A048-980

TR-190 Moss, J.E.B.
Abstract Data Types in Stack Based Languages, S.M. Thesis, EE & CS Department, February 1978; AD A052-332

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 A051-149

TR-192 Niamir, B.
Attribute Partitioning in a Self-Adaptive Relational Database System, S.M. Thesis, EE & CS Department, January 1978; AD A053-292

TR-193 Schaffert, J.C.
A Formal Definition of CLU, S.M. Thesis, EE & CS Department, January 1978

TR-194 Hewitt, C. and Baker, H., Jr.
Actors and Continuous Functionals, February 1978; AD A052-266

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.H.
Final Report of the Multics Kernel Design Project, March 1978

TR-197 Baker, H., Jr.
Actor Systems for Real-Time Computation, Ph.D. Dissertation, EE & CS Department, March 1978: AD A053-328

TR-198 Halstead, R.H., Jr.
Multiple-Processor Implementation of Message-Passing Systems, S.M. Thesis, EE & CS Department, April 1978; AD A054-009

TR-199 Terman, C.J.
The Specification of Code Generation Algorithms, S.M. Thesis, EE & CS Department, April 1978; AD A054-301

TR-200 Harel, D.
Logics of Programs: Axiomatics and Descriptive Power, Ph.D. Dissertation, EE & CS Department, May 1978

TR-201 Scheifler, R.W.
A Denotational Semantics of CLU, S.M. Thesis, EE & CS Department, June 1978

TR-202 Principato, R.N., Jr.
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 A058-232

TR-204 Teixeira, T.J.
Real-Time Control Structures for Block Diagram Schemata, S.M. Thesis, EE & CS Department, August 1978; AD A061-122

PUBLICATIONS

TR-205 Reed, D.P.
Naming and Synchronization in a Decentralized Computer System, Ph.D. Dissertation, EE & CS Department, October 1978; AD A061-407

TR-206 Borkin, S.A.
Equivalence Properties of Semantic Data Models for Database Systems, Ph.D. Dissertation, EE & CS Department, January 1979; AD A066-386

TR-207 Montgomery, W.A.
Robust Concurrency Control for a Distributed Information System, Ph.D. Dissertation, EE & CS Department, January 1979; AD A066-996

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 A068-111

TR-210 Papadimitriou, C.H.
Serializability of Concurrent Database Updates, March 1979

TR-211 Bloom, T.
Synchronization Mechanisms for Modular Programming Languages, S.M. Thesis, EE & CS Department, April 1979; AD A069-819

TR-212 Rabin, M.O.
Digitalized Signatures and Public-Key Functions as Intractable as Factorization, March 1979

TR-213 Rabin, M.O.
Probabilistic Algorithms in Finite Fields, March 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 A068-112

TR-215 Svobodova, L., Liskov, B. and Clark, D.
Distributed Computer Systems: Structure and Semantics, April 1979; AD A070-286

TR-216 Myers, J.M.
Analysis of the SIMPLE Code for Dataflow Computation, June 1979

TR-217 Brown, D.J.
Storage and Access Costs for Implementations of Variable - Length Lists, Ph.D. Dissertation, EE & CS Department, June 1979

TR-218 Ackerman, W.B. and Dennis, J.B.
VAL--A Value-Oriented Algorithmic Language: Preliminary Reference Manual, June 1979; AD A072-394

TR-219 Sollins, K.R.
Copying Complex Structures in a Distributed System, S.M. Thesis, EE & CS Department, July 1979; AD A072-441

TR-220 Kosinski, P.R.
Denotational Semantics of Determinate and Non-Determinate 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., Jr.
Reference Tree Networks: Virtual Machine and Implementation, Ph.D. Dissertation, EE & CS Department, September 1979; AD A076-570

TR-223 Brown, G.P.
Toward a Computational Theory of Indirect Speech Acts, October 1979; AD A077-065

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., Bloom, T., Moss, E., Schaffert, C., Scheifler, R. and Snyder, A.
CLU Reference Manual, October 1979; AD A077-018

TR-226 Reuveni, A.
The Event Based Language and Its Multiple Processor Implementations, Ph.D. Dissertation, EE & CS Department, January 1980; AD A081-950

PUBLICATIONS

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 Serializers, Ph.D. Dissertation, EE & CS Department, March 1980; AD A082-885

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.W.
The Architecture of an Object Based Personal Computer, Ph.D. Dissertation, EE & CS Department, March 1980; AD A083-433

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 A084-411

TR-234 Herlihy, M.P. Transmitting Abstract Values in Messages, S.M. Thesis, EE & CS Department, May 1980; AD A086-984

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 A085-877

TR-238 Bloniarz, P.A.
The Complexity of Monotone Boolean Functions and an Algorithm for Finding Shortest Paths in a Graph, Ph.D. Dissertation, EE & CS Department, June 1980

TR-239 Baker, C.M.
Artwork Analysis Tools for VLSI Circuits, S.M. & EE Thesis, EE & CS Department, June 1980; AD A087-040

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., Jr.
Representation and Analysis of Real-Time Control Structures, S.M. Thesis, EE & CS Department, August 1980; AD A089-828

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 A089-836

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

PUBLICATIONS

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 A096-374
TR-253	Ilson, R. An Integrated Approach to Formatted Document Production, S.M. Thesis, EE & CS Department, February 1981
TR-254	Ruth, G., Alter, S. and Martin, W. A Very High Level Language for Business Data Processing, March 1981
TR-255	Kent, S.T. 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	Cisari, C. Application of Data Flow Architecture to Computer Music Synthesis, S.B./S.M. Thesis, EE & CS Department, February 1981
TR-258	Singh, N. 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; AD 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

PUBLICATIONS

TR-271 Rutherford, C.J., Davies, B., Barnett, A.I., 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 1981

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 1982

TR-292 Dornbrook, M. and Blank, M.
The MDL Programming Language Primer, EE & CS Department, June 1980

PUBLICATIONS

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.K.
Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment, Ph.D. Dissertation, EE & CS Department, June 1983; AD A139996

TR-298 Krugler, K.
Video Games and Computer Aided Instruction, EE & CS Department, June 1983

TR-299 Wing, J.
A Two Tiered Approach to Specifying Programs, June 1983; AD A133949

TR-300 Cooper, G.
An Argument for Soft Layering of Protocols, May 1983; AD 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; AD A136094

TR-303 Bloom T.
Dynamic Module Replacement in a Distributed Programming System, Ph.D. Dissertation, EE & CS Department, September 1983; AD A140619

TR-304 Terman, C.J.
Simulation Tools for Digital LSI Design, Ph.D. Dissertation, EE & CS Department, September 1983; AD A136116

TR-305 Bhatt, S.N. and Leighton, F.T.
A Framework for Solving VLSI Graph Layout Problems, Ph.D. Dissertation, EE & CS Department. October 1983; AD 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; AD A136117

TR-308 Oki, B.M.
Reliable Object Storage to Support Atomic Actions, M.S. Thesis, EE & CS Department, November 1983; AD 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, M.S. Thesis, EE & CS Department, December 1983; AD A148990

TR-311 Burke, G.G., Carrette, G.J. and Eliot, C.R.
NIL Reference Manual, M.S. Thesis, EE & CS Department, December 1983

TR-312 Lancaster, J.
Naming in a Programming Support Environment, M.S. Thesis, EE & CS Department, April 1984; AD A142018

TR-313 Koile, K.
The Design and Implementation of an Online Directory Assistance System, M.S. Thesis, EE & CS Department, April 1984; AD A140821

TR-314 Weihl, W.
Specification and Implementation of Atomic Data Types, Ph.D. Dissertation, EE & CS Department, April 1984; AD A141823

PUBLICATIONS

TR-315 Coan, B. and Turpin, R.
Extending Binary Byzantine Agreement to Multivalued Byzantine Agreement, April 1984; AD 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; AD 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 VIMVAL, 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; AD 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; AD 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; AD A150562

TR-327 Chiu, S.Y.
Debugging Distributed Computations in a Nested Atomic Action System, Ph.D. Dissertation, EE & CS Department, December 1984; AD 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; AD A154723

TR-331 Sollins, K.R.
Distributed Name Management, Ph.D. Dissertation, EE & CS Department, February 1985; AD A154785

TR-332 Culler, D.E.
Resource Management for the Tagged Token Dataflow Architecture, S.M. Thesis, EE & CS Department, January 1985; AD A154773

TR-333 Arnold, J.M.
Parallel Simulation of Digital LSI Circuits, S.M. Thesis, EE & CS Department, April 1984; AD 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

PUBLICATIONS

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, S.M. Thesis, EE & CS Department, May 1985; AD 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; AD 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; AD 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 ; AD 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; AD 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; AD 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

PUBLICATIONS

TR-359 Feldmeier, D.C.
A CATV-Based High-Speed Packet-Switching Network Design, S.M. Thesis, EE & CS Department, April 1986; AD 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; AD 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; AD A171428

TR-368 Scheifler, R.W. and Gettys, J.
The X Window System, October 1986; AD 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; AD A 176525

TR-374 Goldberg, A.V.
Efficient Graph Algorithms for Sequential and Parallel Computers, Ph.D. Dissertation, EE & CS Department, February 1987; AD A178403

TR-375 Osborne, R.B.
Modeling the Performance of the Concert Multiprocessor, S.M. Thesis, EE & CS Department, May 1987; AD A183619

TR-376 Day, M.S.
Replication and Reconfiguration in a Distributed Mail Repository, S.M. Thesis, EE & CS Department, May 1987; AD A186967

TR-377 Ng, P.
Long Atomic Computations, Ph.D. Dissertation, EE & CS Department, October 1986; AD 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; AD A175776

TR-381 Sherman, A.T.
Cryptology and VLSI, Ph.D. Dissertation, EE & CS Department, October 1986; AD A175853

PUBLICATIONS

TR-382 Chien, A.A.
Congestion Control in Routing Networks, S.M. Thesis, EE & CS Department, October 1986; AD 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.K.
Remote Pipes and Procedures for Efficient Distributed Communications, October 1986

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; AD 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: AD A186896

Progress Reports

- Project MAC Progress Report I, to July 1964; AD 465-088
- Project MAC Progress Report II, July 1964-July 1965; AD 629-494
- Project MAC Progress Report III, July 1965-July 1966; AD 648-346
- Project Mac Progress Report IV, July 1966-July 1967; AD 681-342
- Project MAC Progress Report V, July 1967-July 1968; AD 687-770
- Project MAC Progress Report VI, July 1968-July 1969; AD 705-434
- Project MAC Progress Report VII, July 1969-July 1970; AD 732-767
- Project MAC Progress Report VIII, July 1970-July 1971; AD 735-148
- Project MAC Progress Report IX, July 1971-July 1972; AD 756-689
- Project MAC Progress Report X, July 1972-July 1973; AD 771-428
- Project MAC Progress Report XI, July 1973-July 1974; AD A004-966
- Laboratory for Computer Science Progress Report XII, July 1974-July 1975; AD A024-527
- Laboratory for Computer Science Progress Report XIII, July 1975-July 1976; AD A061-246
- Laboratory for Computer Science Progress Report XIV, July 1976-July 1977; AD A061-932
- Laboratory for Computer Science Progress Report 15, July 1977-July 1978; AD A073-958
- Laboratory for Computer Science Progress Report 16, July 1978-July 1979; AD A088-355
- Laboratory for Computer Science Progress Report 17, July 1979-July 1980; AD A093-384
- Laboratory for Computer Science Progress Report 18, July 1980-June 1981; AD A127586

- Laboratory for Computer Science Progress Report 19, July 1981-June 1982; AD A143429
- Laboratory for Computer Science Progress Report 20, July 1982-June 1983; AD A145134
- Laboratory for Computer Science Progress Report 21, July 1983-June 1984; AD A154810
- Laboratory for Computer Science Progress Report 22, July 1984-June 1985
- Laboratory for Computer Science Progress Report 23, July 1985-June 1986

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.