Abstracts of all technical reports published by the Department of Computer Science between 1985 and 1991 are listed below. Some of these reports are available for anonymous `ftp` or can be purchased from the Department. Details are given at the end of this report.

TR-001-85 Mutual Exclusion in Partitioned Distributed Systems
Daniel Barbara and Hector Garcia-Molina
A network partition can break a distributed computing system into groups of isolated nodes. When this occurs, a mutual exclusion mechanism may be required to ensure that isolated groups do not concurrently perform conflicting operations. We study and formalize these mechanisms in three basic scenarios: where there is a single conflicting type of action; where there are two conflicting types, but operations of the same type do not conflict; and where there are two conflicting types, but operations of one type do not conflict among themselves. For each scenario, we present applications that require mutual exclusion (e.g., name servers, termination protocols, concurrency control). In each case, we also present mutual exclusion mechanisms that are more general and that may provide higher reliability than the voting mechanisms that have been proposed as solutions to this problem. (44 pages, July 1985)

TR-002-85 Database Processing with Triple Modular Redundancy
Frank Pittelli and Hector Garcia-Molina
N-Modular Redundancy (NMR) protects against arbitrary types of hardware or software failures in a minority of system components, thereby yielding the highest degree of reliability. In this paper we study the application of NMR, specifically Triple Modular Redundancy (TMR), to general-purpose database processing. We discuss the structure and implementation tradeoffs of a TMR system that is "synchronized" at the transaction level. That is, complete transactions are distributed to all nodes, where they are processed independently, and only the majority output is accepted. We examine the inherent "cost" of such a TMR database system by presenting preliminary performance results from a version implemented on three SUN-2/120 workstations. (34 pages, July 1985)

TR-003-85 Rotation Distance
Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston
In this note we summarize our recent results on rotation distance, a distance measure on binary trees with computer science applications. Our main result is that the maximum rotation distance between any two n-node binary trees is at most $2n - 6$ for $n \geq 11$, and this bound is tight for infinitely many n. (10 pages, July 1985)

TR-004-85 A Locally Adaptive Data Compression Scheme
Jon Louis Bentley, Daniel E. Sleator, Robert E. Tarjan, and Victor K. Wei
We describe a data compression scheme that exploits locality of reference, such as occurs when words are used frequently over short intervals and then fall into long periods of disuse. The scheme is based on a simple heuristic for self-organizing sequential search and on variable-length encodings of intergers. We prove that it never performs much worse than Huffman coding and can perform substantially better; experiments on real files show that its performance is usually quite close to that of Huffman coding. Our scheme has many implementation advantages; it is simple, allows fast encoding and decoding, and requires only one pass over the data to be compressed (static Huffman coding takes two passes). (30 pages, July 1985)

TR-005-85 Planar Point Location Using Persistent Search Trees
Neil Sarnak and Robert E. Tarjan
The planar point location problem is that of preprocessing a polygonal subdivision of the plane so that, given a sequence of points, the polygon containing each point can be determined quickly. Several ways of solving this problem in $O(\log n)$ query-time and $O(n)$-space are known, but they are all rather complicated. We propose a simple $O(\log n)$-query time, $O(n)$-space solution, using persistent search trees. A persistent search tree differs from an ordinary search tree in that after an insertion or deletion, the old version of the tree can still be searched. We develop a persistent form of binary search tree that supports insertions and deletions in the present version and queries in any version, past or present. The time per query or update is $O(\log m)$, where m is the total number of updates, and the space needed is $O(1)$ per update. Our planar point location algorithm is an immediate application of this data structure. (25 pages, July 1985)

TR-006-85 Efficient Top-Down Updating of Red-Black Trees
Robert E. Tarjan
The red-black tree is an especially flexible and efficient form of binary search tree. In this note we show that an insertion or deletion in a red-black tree can be performed in one top- down pass, requiring $O(1)$ rotations and color changes in the amortized case. (15 pages, June 1985)

TR-007-85 Rectilinear Planar Layouts of Planar Graphs and Bipolar Orientations
Pierre Rosenstiehl and Robert E. Tarjan
We propose a linear-time algorithm for generating a planar layout of a planar graph. Each vertex is represented by a horizontal line segment and each edge by a vertical line segment. All endpoints of the segments have integer co-ordinates. The total space occupied by the layout is at most n by at most $2n - 4$. Our algorithm, a variant of one by Otten and van Wijk, generally produces a more compact layout than theirs and allows the dual of the graph to be laid out in an interlocking way. The algorithm is based on the concept of a bipolar orientation. We discuss relationships among the bipolar orientations of a planar graph. (19 pages, July 1985)

TR-008-85 The Pairing Heap: A New Form of Self-Adjusting Heap
Michael L. Fredman, Robert Sedgewick, Daniel D. Sleator and Robert E. Tarjan
Recently, Fredman and Tarjan invented a new, especially efficient form of heap (priority queue) called the Fibonacci heap. Although theoretically efficient, Fibonacci heaps are complicated to implement and not as fast in practice as other kinds of heaps. In this paper we describe a new form of heap, called the pairing heap, intended to be competitive with the Fibonacci heap in theory and easy to implement and fast in practice. We provide a partial complexity analysis of pairing heaps. Giving a complete analysis remains an open problem. (31 pages, July 1985)

TR-009-85 Using Semantic Knowledge for Transaction Processing
Ricardo A. Cordon (Thesis)
New methods of concurrency control that utilize the semantics of an application to improve performance have been proposed in recent years. In this thesis we study one of these mechanisms. After presenting the details of the mechanism, we compare it, via a simulation, to a conventional two phase locking strategy. The goal of such comparison is to determine the conditions under which the higher complexity and overhead of the application dependent mechanism pays off. Although the results presented are specific to the two selected mechanisms, we believe they provide insight into the operation of other application dependent mechanisms. Also, we present an application of our mechanism that will help to manage long lived transactions (LLT). This application shows that for the cases when a large percentage of the transactions in the system are compatible with the executing LLTs, the performance of the system improves greatly. (145 pages, October 1985)

TR-010-85 Policies for Dynamic Vote Reassignment
Daniel Barbara, Hector Garcia-Molina, and Annemarie Spauster
Voting is used commonly to enforce mutual exclusion in distributed systems. Each node is assigned a number of votes and only the group with a majority of votes is allowed to perform a restricted operation. This paper describes techniques for dynamically assigning votes upon node or link failure, in an attempt to make the system more resilient. The emphasis is on policies with which a node autonomously select its new vote value. Simulation results are presented comparing the autonomous methods to static vote assignments and to group consensus strategies. (22 pages, September 1985)

TR-011-85 A Suggested Architecture for the Massive Memory Machine
Daniel Lopresti
This report describes a computer architecture with two classes of primary storage; a tree-structured memory which is massive, but slow, and a standard, paged memory which is fast, but small. We then argue that only a small percentage of a program's memory references, those made to large (heap) data structures, are responsible for the thrashing a massive memory machine is designed to avoid. Hence, mapping the heap into the tree-structured memory and placing program segments which are well-behaved in the standard memory retains the benefits of the massive memory, while minimizing the impact of its slower access time. Finally, we propose several enhancements to the basic architecture that permit limited parallel processing. (7 pages, October 1985)

TR-012-85 Optimizing Shadow Recovery Algorithms

Jack Kent and Hector Garcia-Molina
Experiments conducted on a database testbed at Princeton indicate excessive page-table I/O is the major performance drawback of shadow recovery. In light of this, we propose a method for parametrizing shadow recovery that minimizes page-table I/O without sacrificing too much disk utilization. Using a simple model, we analyze and evaluate our mechanism, comparing it to two conventional ones. (46 pages, September 1985)

TR-013-85 Two Streamlined Depth-First Search Algorithms
Robert E. Tarjan
Many linear-time graph algorithms using depth-first search have been invented. We propose simplified versions of two such algorithms, for computing a bipolar orientation or st-numbering of an undirected graph and for finding all feedback vertices of a directed graph. (18 pages, July 1985)

TR-014-85 Geometric Complexity and Computer Graphics — Does Theory Apply in Practice?
David P. Dobkin
Theoretical work in geometric complexity is often justified by its relevance to key problems of computer graphics, most notable the problems of hidden line and hidden surface removal. We consider a geometric structure — the convex drum — both in the context of a theoretical algorithm for polyhedral intersection and in a practical context giving an algorithm for computing and decomposing unions of polygons. This is used as a model of situations where theoretical ideas can have relevance to actual implementations. (13 pages, July 1985)

TR-015-85 Embedding Computation in One-Dimensional Automata by Phase Coding Solitons
Kenneth Steiglitz, Irfran Kamal and Arthur Watson
We show that some kind of meaningful computation can be embedded in very simple, microscopically homogeneous, one-dimensional automata — filter automata with a parity next-state rule. A systematic procedure is given for generating moving, periodic structures ("particles"). These particles exhibit soliton-like properties; that is, they often pass through one another with phase shifts. We then discuss ways to encode information in the phase of these particles. Finally, the search for useful logical operations is reduced to a search for paths in certain graphs. As a demonstration of principle, we give the details of implementing a carry-ripple adder. (17 pages, November 1985)

TR-016-85 An Example of Structured Explanation Generation
M. A. Bienkowski
An explanation mechanism designed to explain the behavior of a goal-driven system which navigates its way around the streets of Princeton is described in this paper. The mechanism embodies a model in which explanation is viewed as a type of goal-directed, purposive discourse. In this model, discourse is produced via the achievement of goals at several levels, each level posing goals to be satisfied by lower levels, until a set of instructions which a sentence generator can follow to produce English is reached. These levels are: linearization, selection and local coherence operations. The model was implemented and tested for a special method of linearization called structured linearization. Structured linearization utilizes the order inherent in the knowledge being explained to provide the ordering necessary for test generation. An example of an actual run of the system is shown. The explanation mechanism is currently being enhanced with the addition of a dialogue management system; a brief sketch of this work is also presented. (31 pages, November 1985)

TR-017-85 Design of an Interleaved Massive Memory Machine
Arvin Park
This document presents a high level description of an architecture for a massive memory machine (a computer with several gigabytes of main memory). The architecture provides a logical extension of the standard three level (cache, main memory, disk) memory hierarchy to a four level scheme (massive register set, cache, main memory, disk). The design combines: the compiler optimized data motion of registers; the temporal locality advantages of caching; and the high throughput benefits of interleaving into a design which suggests a very natural physical implementation using present day packaging technologies. (20 pages, November 1985)

TR-018-86 A Load Balancing Implementation for a Local Area Network of Workstations
Rafael Alonso, Phillip Goldman and Peter Potrebic
This paper presents the details of a prototype load balancing package and tabulates the results of

preliminary performance measurements. Our implementation uses the networking facilities provided in 4.2 BSD UNIX, and is currently running on a local area network of SUN workstations. Although our work is still in progress, initial benchmarks show that our approach is extremely promising. (14 pages, January 1986)

TR-019-86 Superposed Parallel Buses: A Systolic Area-Time Optimal VLSI Interconnection
Bruce W. Arden and Toshio Nakatani
This paper presents a new class of interconnection schemes, based on superposed parallel buses and called systolic grid interconnection. This scheme is optimal both in area and time in terms of VLSI complexity for multi-point networks with prescheduled permutation. This scheme is faster than previously reported interconnection schemes using linear area. This scheme is especially suitable for utilizing fast but area-limited technology to interconnect a large number of single chip processors using slower technology. Although the primary proposed application is to solve large sparse systems of linear equations, such as those arising from finite element analysis, this interconnection scheme is widely applicable because of its fast permutation capability. The construction of the grid and its VLSI optimality are described as well as some applications. (21 pages, February 1986)

TR-020-86 The Total DNA Homology Experiment
Richard J. Lipton, Daniel Lopresti, J. Douglas Welsh
This report describes briefly an experiment which may help answer fundamental questions in molecular biology. In essence, we plan to compare all known DNA sequences with each other, filtering out close matches for further analysis. The size of the computation, in terms of total operations, makes it one of the largest attempted for any purpose; the experiment only becomes feasible when massive parallelism is employed. To this end, we have already designed, fabricated, and tested a systolic array for DNA sequence matching. Preliminary benchmarks indicate that it is hundreds of times faster than current minicomputers. Using a small number of these chips, we will be able to complete the experiment in one year's time. (7 pages, January 1986)

TR-021-86 The Rotation Graph of Binary Trees is Hamiltonian
Joan M. Lucas
In this paper we study the rotation transformation on binary trees and consider the properties of binary trees under this operation. The rotation is the universal primitive used to rebalance binary search trees. It is a simple, local restructuring technique that alters the depths of some of the nodes in the binary tree. New binary search tree algorithms have recently been introducted by Sleator and Tarjan. It has been conjectured that these algorithms are as efficient as any algorithm that dynamically restructures the tree using rotations. We hope that by studying rotations in binary trees, which in turn will lead to a proof of this"dynamic optimality conjecture." We define a graph $RG(n)$, whose vertex set consists of all binary trees containing n nodes, and which has an edge between two trees if they differ by only one rotation. We shall introduce a new characterization of the structure of $RG(n)$ and use it to demonstrate the existence of a Hamiltonian cycle in the graph. The proof is constructive and can be used to enumerate all binary trees with n nodes in $O(C_n)$ time, where each tree in the list differs from its predecessor by only one rotation. (62 pages, January 1986)

TR-022-86 Using Residue Arithmetic to Simplify VLSI Processor Arrays for Dynamic Programming
Richard J. Lipton and Daniel Lopresti
We introduce a technique for transforming dynamic programming algorithms into ones which are equivalent, but require asymptotically less space and time under the logarithmic cost criterion. When mapped into silicon, these transformed algorithms result in VLSI processor arrays which can be significantly smaller and faster. The condition necessary for such savings is characterized. We illustrate the discussion with a general case of serial dynamic programming and nonserial pattern matching problem. Finally, we show that our technique does not apply to every instance of dynamic programming. (20 pages, January 1986)

TR-023-86 A Weinberger Array Generator
William W. Lin, Susan S. Yeh and Andrea S. LaPaugh
The Weinberger Array Generator (WAG) is a tool for implementing random logic. Boolean equations are input, and a layout description of gates and wires (the circuit) realizing the equations is output. In the above aspects, WAG is similar to a PLA generator. The main difference is that the Weinberger

array structure allows many levels of logic, with complex gates such as NAND-of-ORs; whereas a PLA structure allows only two levels of logic, with no gates more complex than NORs. We describe our implementation of WAG, presenting issues concerning the optimization of logic, placement of gates, track assignment, and layout. Along with this, we discuss the trade-off between space requirements and timing delays that must be considered in choosing between a PLA and a Weinberger Array structure. Finally, we discuss possible improvements and uses for WAG. (39 pages, January 1986)

TR-024-86 Permutations on Superposed Parallel Buses
Bruce W. Arden and Toshio Nakatani
This paper presents two systolic algorithms, a self-pipelined and a multi-pipelined algorithm, for both static and dynamic permutations. These algorithms are derived from the simulation of an Ω-network on interconnection schemes based on superposed parallel buses. These algorithms are area-efficient and almost optimal for permutations using the AT^2 measure. They are faster than previously reported interconnection schemes using almost linear area. They are especially suitable for highspeed but area-limited technologies, due to simple control algorithms and regular, area-efficient interconnections. Descriptions of these two algorithms and a summary of their VLSI complexities for both static and dynamic cases are included. (27 pages, January 1986)

TR-025-86 Intersection of Convex Objects in Two and Three Dimensions
Bernard Chazelle and David P. Dobkin
One of the basic geometric operations involves determining whether a pair of convex objects intersect. This problem is well understood in a model of computation where the objects are given as input and their intersection is returned as output. For many applications, however, we may assume that the objects already exist within the computer and that the only output desired is a single piece of data giving a common point if the objects intersect, or reporting no intersection if they are disjoint. For this problem, none of the previous lower bounds are valid and we propose algorithms requiring sublinear time for their solution in two and three dimensions. (40 pages, January 1986)

TR-026-86 Comparing Long Strings on a Short Systolic Array
Richard J. Lipton and Daniel Lopresti
In this paper we demonstrate two techniques for comparing strings of arbitrary length on a systolic array we have proposed and implemented. The first is an application of algorithm partitioning, the second an implementation of a string matching heuristic for which we prove performance bounds. We analyze the time and number of processors required in each case and determine the processor utilizations. (15 pages, February 1986)

TR-027-86 Practical Variations of Shellsort
Janet Incerpi and Robert Sedgewick
Shellsort is based on a sequence of integer increments h_i and works by performing insertion sort on subfiles consisting of every h_ith element. We consider variations of Shellsort that limit the work performed per pass. By allowing only linear work per pass, insurance must be given that the file is sorted after $O(\log N)$ passes. We describe one such method: it uses $\log N$ passes, has potential as a practical sorting algorithm, and could possibly lead to a simple sorting network. (9 pages, February 1986)

TR-028-86 Reliable Scheduling in a TMR Database System
Frank M. Pittelli and Hector Garcia-Molina
A Triple Modular Redundant (TMR) system achieves high reliability by replicating data and all processing at three independent nodes. When TMR is used for database processing all non-faulty computers must execute the same sequence of transactions, and this is ensured by a collection of processes known as schedulers. In this paper we study the implementation of efficient schedulers through analysis of various enhancements such as null transactions and message batching. The schedulers have been implemented in an experimental TMR system and the evaluation results are presented here. (45 pages, April 1986)

TR-029-86 Distributed Computing Research at Princeton — 1985
R. Abbot, R. Alonso, D. Barbara, R. Cordon, H. Garcia-Molina, P. Goldman, M. Karol, J. Kent, B. Kogan, F. Pittelli, P. Potrebic, S. Schwartz, P. Simpson, A. Spauster, M. Sotessl and S. Verdu
This note briefly summarizes the distributed computing research we performed in the year 1985. In general terms, our emphasis was on studying and implementing mechanisms for efficient and reliable computing and data management. Our work can be roughly divided into nine categories: mutual

exclusion, dynamic vote reassignment, the implementation of a highly reliable database system, the implementation of a high availability database system, a survey of reliable distributed data management, the optimization of shadow recovery mechanisms, load balancing, catching and prefetching in information systems, and protocols for multiple-access packet-broadcast channels. Due to space limitations, we concentrate on describing our own work and we do not survey the work of other researchers in the field. For survey information and references, we refer readers to some of our reports. (15 pages, February 1986)

TR-030-86 PATHALIAS or The Care and Feeding of Relative Addresses
Peter Honeyman and Steven M. Bellovin
Pathalias computes electronic mail routes in environments that mix explicit and implicit routing, as well as syntax styles. We describe the history of pathalias, its algorithms and data structures, and our design decisions and compromises. Pathalias is guided by a simple philosophy: get the mail through, reliably and efficiently. We discuss the principles of routing in heterogeneous environments necessary to make this philosophy a reality. (13 pages, March 1986)

TR-031-86 Optimization of One-Bit Full Adders Embedded in Regular Structures
Kazuo Iwano and Kenneth Steiglitz
We study the problem of optimizing the transistor sizes in the one-bit nMOS full adder either isolated or embedded in a regular array. A local optimization method that we call the critical-path optimization method is developed. In this method two parameters at a time are changed along the critical path until a locally optimal choice of transistor sizes is found. The critical-path optimization method uses the Berkeley VLSI tools and the hierarchical layout language ALLENDE developed at Princeton. First, we optimize the isolated one-bit full adder implemented in three ways: as a PLA, Data Selector, and with Random Logic. The details of the critical path optimization method and Power-Time tradeoff curves are illustrated here. Second, we optimize the one-bit full adder embedded in a simple array multiplier. The entire $3 \times 3, 4 \times 4, 8 \times 8$ and 10×10 multipliers are optimized and their local optima are compared. Because the optimization of the entire circuit becomes less practical when the circuit becomes larger, we develop a method that makes use of circuit regularity. We prove that some small array of one-bit full adders, called the canonical configuration, has the same local optima as the $n \times n$ multiplier for large n, with the criterion of minimizing the delay time T. Hence we can greatly reduce the computation load by optimizing this canonical configuration instead of optimizing the entire circuit. Experimental results confirm the effectiveness of this approach. (35 pages, March 1986)

TR-032-86 A New Approach to Fast Control of Permutation Networks
Bruce Arden and Abdou Youssef
Controlling Benes-Clos networks in full generality has proven to be very slow. Many approaches have been taken to speed up the control for certain classes of permutations. In this report, a new approach is developed for 3-stage networks with $n \times n$ switches as building blocks (denoted by $B(n, 2)$). The new approach allows the network to be self-routed for many interesting classes of problems, or more precisely, the permutations they need. The families of permutations that can be self-routed by the new scheme are characterized. Several problems such as FFT, bitonic sorting, simulation of standard fixed interconnection networks on $B(n, 2)$, and others are shown to produce families of permutations that yield to the new scheme. A new control algorithm of $O(\log^2 N)$ complexity is derived for $B(n, 2)$ where $N = n^2$ is the number of terminals, and possible implementation of the scheme is discussed. (42 pages, April 1986)

TR-033-86 The Optimal Uniform Schedules of Arbitrary Static Permutations on Superposed Parallel Buses
Bruce Arden and Toshio Nakatani
The paper is concerned with the optimal schedule for the permutation of n^2 packetes on an $n \times n$ square grid of superposed, parallel time multiplexed buses. It is shown that the Uniform schedule (that is, all two-step transfers consistently row first or alternately column first) is optimal. Moreover, it requires $n+1$ bus transfers, or cycle times. Although n-cycle, non-uniform schedules exist for specific permutations, it is shown that $n + 1$ cycle, uniform schedule is optimal. (12 pages, April 1986)

TR-034-86 Crash Recovery Mechanisms for Main Storage Database Systems
Kenneth Salem and Hector Garcia-Molina
In a main storage database system the primary copy of all data resides permanently in primary (semiconductor) memory. A major problem for such systems is crash recovery, i.e., ensuring that transactions

are atomic and durable in spite of main memory's volatility. In this paper we study several possible crash recovery mechanisms and analyze their impact on performance. (28 pages, May 1986)

TR-035-86 The Design of Load Balancing Strategies for Distributed Systems
Rafael Alonso
In this paper we consider the problem of designing and selecting load balancing mechanisms for distributed systems based on local area networks. In particular, we will focus on the type of information needed to make balancing decisions and on the desirable properties of the decision algorithms. We also describe our current approach to this problem, which essentially consists of carrying out a series of experiments on a prototype load balancing implementation. Finally, we present the insights we have derived from our experimental results. (6 pages, May 1986)

TR-036-86 Massive Memory Means Massive Performance
Arvin Park
While most of the computing world has been exploring the uses of parallelism to increase computational speed, we at Princeton have been examining methods of using large amounts of semiconductor memory to achieve the same end. We have demonstrated that time-space tradeoffs can be exploited across a wide range of applications to speed up computational tasks. On some tasks, our 128 megabyte VAX will outperform even the fastest supercomputers. These time-space tradeoffs coupled with rapidly falling semiconductor memory prices make Massive Memory systems economically inevitable. This paper documents performance results from our 128 megabyte testbed machine, and goes on to investigate promising application domains for Massive Memory computing systems. (16 pages, May 1986)

TR-037-86 Protocols for Dynamic Vote Reassignment
Daniel Barbara, Hector Garcia-Molina and Annmarie Spauster
Voting is used commonly to enforce mutual exclusion in distributed systems. Each node is assigned a number of votes and only the group with a majority of votes is allowed to perform a restricted operation. This paper describes techniques for dynamically reassigning votes upon node or link failure, in an attempt to make the system more resilient. Protocols are given which allow nodes to select new vote values autonomously while still maintaining mutual exclusion requirements. The lemmas and theorems to validate the protocols are presented, along with proof of correctness. A simple example shows how to apply the method to a database object-locking scheme; the protocols, however, are versatile and can be used for any application requiring mutual exclusion. Also included is a brief discussion of simulation results. (11 pages, May 1986)

TR-038-86 Three Partition Refinement Algorithms
Robert Paige and Robert E. Tarjan
We present improved partition refinement algorithms for three problems: lexicographic sorting, relational coarsest partition, and double lexical ordering. (23 pages, January 1986)

TR-039-86 Linear Time Algorithms for Visibility and Shortest Path Problems Inside Simple Polygons
Leo Guibas, John Hershberger, Daniel Leven, Micha Sharir and Robert E. Tarjan
We present linear time algorithms for solving the following problems involving a simple planar polygon P: (i) Computing the collection of all shortest paths inside P from a given source vertex s to all the other vertices of P; (ii) computing the subpolygon of P consisting of points that are visible from a segment within P; (iii) preprocessing P so that for any query ray r emerging from some fixed edge e of P, we can find in logarithmic time the first intersection of r with the boundary of P; (iv) preprocessing P so that for any query point x in P, we can find in logarithmic time the portion of the edge e that is visible from x; (v) preprocessing P so that for any query point x inside P and direction u, we can find in logarithmic time the first point on the boundary of P hit by the ray at direction u from x; (vi) calculating a hierarchical decomposition of P into smaller polygons by recursive polygon cutting; and (vii) calculating the (clockwise and counterclockwise)"convex ropes" (in the terminology of [PS]) from a fixed vertex s of P lying on its convex hull, to all other vertics of P. All these algorithms are based on a recent linear time algorithm of Tarjan and Van Wyk for triangulating a simple polygon, but use additional techniques to make all subsequent phases of these algorithms also linear. (49 pages, May 1986)

TR-040-86 K-Way Bitonic Sort
Bruce W. Arden and Toshio Nakatani
The paper presents k-way bitonic sort, which is the generalization of Batcher's bitonic sort. K-way

bitonic sort is based on the k-way decomposition scheme instead of two-way decomposition. We prove that Batcher's bitonic sequence decomposition theorem still holds with multi-way decomposition. This leads to the applications of the sorting network with bitonic sorters of arbitrary or mixed sizes. (14 pages, May 1986)

TR-041-86 Bus Partitionability and Parallel Permutations
Bruce W. Arden and Toshio Nakatani
This memo provides an answer to the question whether bus-partitionability can improve the performance of parallel permutations. With permutations, no further reduction in time, due to increased parallelism, is possible through the use of partitionable buses. (8 pages, May 1986)

TR-042-86 Array Access Bounds for Block Storage Memory Systems
Arvin Park, K. Balasubramanian and Richard J. Lipton
This paper explores array storage and access strategies on block storage devices (interleaved memories, RAM disks, and disk drives). A tradeoff is exhibited for row access speed and column access speed, and an optimal upper bound for their product is established. Practical array access strategies are discussed as well as extensions and further research. (14 pages, June 1986)

TR-043-86 Achieving High Availability in Distributed Databases
Hector Garcia-Molina and Boris Kogan
A new approach is presented for managing distributed database systems in the face of communication failures and network partitions. It offers high availability and, at the same time, guarantees some meaningful correctness properties. The approach is based on the idea of fragments and agents. It does not require prompt and correct detection of partitions and other failures. (28 pages, June 1986)

TR-044-86 Soliton Phase Shifts in a Dissipative Lattice
Nayeem Islam and Kenneth Steiglitz
We report measurements of soliton phase shifts resulting from head-on collisions in the LC lattice of Hirota and Suzuki, the electrical analog of the Toda Lattice. The effect of dissipation along this lattice is to decrease the amplitude and increase the width of the solitons as they travel down the lattice. To within experimental error, however, we found that the velocity and phase shifts remain constant, so that a soliton species is uniquely determined by its velocity. This shows that the positional phase of such solitons can be used to encode information in a very simple way, and that the lattice can be used to do computation, of which parity checking is a simple example. (16 pages, June 1986)

TR-045-86 Garbage Collection Can Be Faster Than Stack Allocation
Andrew W. Appel
Optimizing LISP compilers try to deduce which closures (function-call frames) may be stack-allocated, because heap-allocated closures must be garbage-collected at (presumably) great expense. In fact, with enough memory, garbage-collection becomes cheaper than decrementing a stack pointer. Special hardware, intricate garbage-collection algorithms, and fancy compiler analysis become unnecessary. (4 pages, June 1986)

TR-046-86 Optimal Parallel Sorting on a Linear Processor Array
Arvin Park and K. Balasubramanian
The problem of sorting n elements on k linearly connected processors is examined. We reduce the communication complexity (measured in units of single data transfers between adjacent processors) by a factor of two from the previous best bound while maintaining the same (asymptotically optimal) number of comparisons. The result holds even if only unidirectional communication between processors is allowed (a unidirectional ring architecture). This result is significant because communication time dominates computation time in most realistic parallel sorting problems. (9 pages, July 1986)

TR-047-86 Reliable Distributed Database Management
Hector Garcia-Molina and Robert K. Abbott
A reliable distributed database must manage dispersed and replicated data, making it available to users in spite of hardware failures. In this paper we study the algorithms and techniques that can achieve this reliability. Three scenarios are considered. In the first, data are distributed but not replicated; in the second, data are replicated; and in the third, both data and processing are fully replicated in an n-modular redundancy strategy. Network partitions are not considered in any scenario. (58 pages, August 1986)

TR-048-86 Improved Sorting Algorithms for Parallel Computers
Arvin Park and K. Balasubramanian
We make observations that improve processor utilization and decrease communication overhead for several parallel sorting algorithms. These lead to constant factor improvements on the best previous parallel sorting bounds for both mesh-connected and linearly connected parallel architectures (Previous bounds were within a constant factor of optimal.) These improved bounds are achieved using fewer processors with greater processor utilization. (16 pages, August 1986)

TR-049-86 One-Processor Scheduling of Tasks with Preferred Starting Times
Michael R. Garey, Robert E. Tarjan and Gordon T. Wilfong
We consider a new class of one-processor scheduling problems having the following form: Tasks $T_1, T_2 ..., T_N$ are given, with each T_i having a specified length l_i and a preferred starting time p_i. The tasks are to be scheduled nonpreemptively (i.e., a task cannot be split) on a single processor as close to their preferred starting times as possible. We examine two different cost measures for such schedules, the sum of the individual discrepancies from the preferred starting times and the maximum such discrepancy. For the first of these, we show that the problem of finding minimum cost schedules is NP-complete; however, we give an efficient algorithm that finds minimum cost schedules whenever the tasks either all have the same length or are required to be executed in a given fixed sequence. For the second cost measure, we give an efficient algorithm that finds minimum cost schedules in general, with no constraints on the ordering or lengths of the tasks. (26 pages, August 1986)

TR-050-86 A New Approach to the Maximum Flow Problem
Andrew V. Goldberg and Robert E. Tarjan
All previously known efficient maximum flow algorithms work by finding augmenting paths, either one path at a time (as in the original Ford and Fulkerson algorithm) or all shortest length augmenting paths at once (using the layered network approach of Dinic). We introduce an alternative method based on the preflow concept of Karzanov. A preflow is like a flow except that the total amount flowing into a vertex is allowed to exceed the total amount flowing out. The method maintains a preflow in the original network and pushes local flow excess toward the sink along what are estimated to be shortest paths. The algorithm and its analysis are simple and intuitive, yet the algorithm runs as fast as any other known method on dense graphs, achieving an $O(n^3)$ time bound on an n-vertex graph. By incorporating the dynamic tree data structure of Sleator and Tarjan, we obtain a version of the algorithm running in $O(nm \log(n^2/m))$ time on an n-vertex, m-edge graph. This is as fast as any known method for any graph density and faster on graphs of moderate density. The algorithm also admits efficient distributed and parallel implementations. We obtain a parallel implementation running in $O(n^2 \log n)$ time and using only $O(m)$ space. This time bound matches that of the Shiloach-Vishkin algorithm, which requires $O(n^2)$ space. (28 pages, July 1986)

TR-051-86 Decomposition and Intersection of Simple Splinegons
David P. Dobkin, Diane L. Souvaine and Christopher J. Van Wyk
A splinegon is a polygon whose edges have been replaced by "well-behaved" curves. We show how to decompose a simple splinegon into a union of monotone pieces and into a union of differences of unions of convex pieces. We also show how to use a fast triangulation algorithm to test whether two given simple splinegons intersect. We conclude with examples of splinegons that make the extension of algorithms from polygons to splinegons difficult. (17 pages, August 1986)

TR-052-86 An $O(n \log\log n)$-Time Algorithm for Triangulating Simple Polygons
Robert E. Tarjan and Christopher J. Van Wyk
Given a simple n-vertex polygon, the triangulation problem is to partition the interior of the polygon into $n-2$ triangles by adding $n-3$ nonintersecting diagonals. We propose an $O(n \log\log n)$-time algorithm for this problem, improving on the previously best bound of $O(n \log n)$ and showing that triangulation is not as hard as sorting. Improved algorithms for several other computational geometry problems follow from our result. (60 pages, July 1986)

TR-053-86 A Semiring On Convex Polygons and Zero-Sum Cycle Problems
Kazuo Iwano and Kenneth Steiglitz
We show that two natural operations on the set of convex polygons form a closed semiring; the two operations are vector summation and convex hull of the union. We then investigate various properties of

these operations: for example, the operation of vector summation takes $O(m \log m)$ time where m is the number of edges involved in the operation, while the decomposition of a given convex polygon into two convex polygons (in a sense, the inverse of vector summation) is NP-complete. Kleene's algorithm applied to this closed semiring solves the problem of determining whether a directed graph with two-dimensional labels has a zero-sum cycle or not. We show that this algorithm runs in polynomial time in the special cases of graphs with one-dimensional labels, BTTSP (Backedged Two-Terminal Series-Parallel) graphs, and graphs with bounded labels. We also investigate the undirected zero-sum cycle problem and the zero-sum *simple* cycle problem. (46 pages, September 1986)

TR-054-86 Robust Contour Tracing
David P. Dobkin, Silvio V. F. Levy, William P. Thurston and Allan R. Wilks
We present a robust method for tracing a curve that is represented as the contour of a function in Euclidean space of any dimension. The method proceeds locally by following the intersections of the contour with the facets of a triangulation of space. The algorithm is robust in the presence of high curvature of the contour, and gives reasonable results when the curve is self-intersecting. It accumulates essentially no round-off error, and has a well-defined integer test for detecting a loop. In developing the algorithm we explore the nature of a particular class of triangulations of Euclidean space, namely, those generated by reflections. (38 pages, September 1986, Revised November 1987)

TR-055-86 Lower Bounds on the Complexity of Multidimensional Searching
Bernard Chazelle
We establish new lower bounds on the complexity of several searching problems. We show that the time for solving the partial sum problem on n points in d dimensions is at least proportional to $(\log n / \log \frac{2m}{n})^{d-1}$ in both the worst and average cases; m denotes the amount of storage used. This bound is provably tight for $m = \Omega(n \log^c n)$ and any $c > d - 1$. We also prove a lower bound of $\Omega(n(\log n / \log \log n)^d)$ on the time required for executing n inserts and queries. Other results include a lower bound on the complexity of orthogonal range searching in d dimensions (in report-mode). We show that on a pointer machine a query time of $O(s + \text{polylog}(n))$ time can only be achieved at the expense of $\Omega(n(\log n / \log \log n)^{d-1})$ space, which is optimal; n and s denote respectively the input and output sizes. (36 pages, October 1986)

TR-056-86 Increasing Availability under Mutual Exclusion Constraints with Dynamic Vote Reassignment
Daniel Barbara, Hector Garcia-Molina and Annemarie Spauster
Voting is used commonly to enforce mutual exclusion in distributed systems. Each node is assigned a number of votes and only the group with a majority of votes is allowed to perform a restricted operation. This paper describes techniques for dynamically reassigning votes upon node or link failure, in an attempt to make the system more resilient to future failures. We focus on autonomous methods for achieving this, i.e., methods that allow the nodes to make independent choices about changing their votes and picking new vote values, rather than group consensus techniques that require tight coordination among the remaining nodes. Protocols are given which allow nodes to install new vote values while still maintaining mutual exclusion requirements. The lemmas and theorems to validate the protocols are presented. A simple example shows how to apply the method to a database object-locking scheme; the protocols, however, are versatile and can be used for any application requiring mutual exclusion. In addition, policies are presented that allow nodes to autonomously select their new vote values. Simulation results are presented comparing the autonomous methods to static vote assignments and to group consensus strategies. These results demonstrate that under high failure rates, dynamic vote reassignment shows great improvement over a static assignment of votes in terms of availability. In addition, many autonomous methods for determining a new vote assignment yield almost as much availability as a group consensus method and at the same time are faster and more flexible. (43 pages, November 1986)

TR-057-86 Providing Fault Tolerance In Parallel Secondary Storage Systems
Arvin Park and K. Balasubramanian
Reliability is a critical concern for designers of parallel data storage systems. These systems consist of large numbers of storage devices which can provide high rates of data transfer. However, the consequential dependence on large numbers of devices can make such systems more prone to failure than non-parallel systems which depend on only a single storage device. This problem can be remedied by employing a modest amount of redundant storage to provide fault tolerance. In fact, providing fault tolerance

in parallel data storage systems requires less redundancy (and is therefore more cost effective) than providing fault tolerance for non-parallel systems. This paper describes and analyses an effective method of providing fault tolerance in parallel data storage systems. (10 pages, November 1986)

TR-058-86 Finding the Optimal Variable Ordering for Binary Decision Diagrams
Steven J. Friedman and Kenneth J. Supowit
The ordered binary decision diagram is a canonical representation for Boolean functions, presented by Bryant as a compact representation for a broad class of interesting functions derived from circuits. However, the size of the diagram is very sensitive to the choice of ordering on the variables; hence for some applications, such as Differential Cascode Voltage Switch (DCVS) trees, it becomes extremely important to find the ordering leading to the most compact representation. We present an algorithm for this problem with time complexity $O(n^2 3^n)$, an improvement over the previous best, which required $O(n!2^n)$. (15 pages, November 1986)

TR-059-86 Recognizing Circle Graphs in Polynomial Time
Csaba P. Gabor, Wen-Lian Hsu and Kenneth J. Supowit
Our main result is an $O(|V|\times|E|)$ time algorithm for deciding whether a given graph is a circle graph, that is, the intersection graph of a set of chords on a circle. Our algorithm utilizes two new graph-theoretic results, regarding necessary induced subgraphs of graphs having neither articulation points nor similar pairs of vertices. Furthermore, as a substep of our algorithm, we show how to find in $O(|V| \times |E|)$ time a decomposition of a graph into prime graphs, thereby improving on a result of Cunningham. (73 pages, July 1986)

TR-060-86 Optimal Permutations on Superposed Parallel Buses
Bruce W. Arden and Toshio Nakatani
The paper is concerned with the optimal schedule for the permutation of n^2 data items on an $n \times n$ square grid formed by the orthogonal superposition of $2n$ time multiplexed buses. The upper bound is proved by showing the existence of the $n + 1$ cycle, uniform schedule (that is, all two-step transfers consistently row first or alternately column first) for an arbitrary permutation. The lower bound is proved by showing the non-existence of n-cycle, non-uniform schedules for some non-degenerate permutations. We also show a simple way to perform an artitrary permutation dynamically with n buffers at every bus intersection. We further show that, with specific example of permutations, the potential increase in parallelism by dynamically partitioning the $2n$ buses does not lead to a further reduction in time. (18 pages, November 1986)

TR-061-86 More Parallelism into the Monte Carlo Solution of Partial Differential Equations
Bruce W. Arden and Addou S. Youssef
The Monte Carlo Method has been studied and used to solve elliptic and parabolic partial differential equatiions. It has several numerical and computational advantages over other methods. The main computational advantage is the great amount of inherent parallelism it manifests. However, an often costly part of the method has remained sequential. It is the random-walk computation (RWC). In this report we parallelize (RWC) using fan-in and fan-out methods. The parallel algorithm takes $O(\log n)$ time while the sequential one takes $O(n)$ where n is the average random-walk length. (16 pages, November 1986)

TR-062-86 Linear Space Data Structures for Two Types of Range Search
Bernard Chazelle and Herbert Edelsbrunner
This paper investigates the existence of linear space data structures for range searching. We examine the homothetic range search problem, where a set S of n points in the plane is to be preprocessed so that for any triangle T with sides parallel to three fixed directions the points of S that lie in T can be computed efficiently. We also look at domination earching in three dimensions. In this problem, S is a set of n points in E^3 and the question is to retrieve all points of S that are dominated by some query point. We describe linear space data structures for both problems. The query time is optimal in the first case and nearly optimal in the second. (21 pages, November 1986)

TR-063-86 Bitonic Sorting on Superposed Parallel Buses
Bruce W. Arden and Toshio Nakatani
The paper develops time complexity measures, including constants, for bitonic sorting on non-partitionable and partitionable buses, in both linear and two dimensional cases. The P processing elements, which share the time multiplexed buses, have a single input port and single output port which are connected

to the buses. The bus bandwidth, B, represented by the number of bus cycles per processor cycle, is a parameter of the model. This parameter can alternately be viewed as the number of parallel buses having one bus cycle per processor cycle. The routing over the buses is based on an optimal data movement for bitonic merge on a linear bus. For the two dimensional case, the execution time is close to optimal in terms of VLSI complexity. (45 pages, November 1986)

TR-064-86 Maintaining Availability of Replicated Data in a Dynamic Failure Environment
Daniel Barbara, Hector Garcia-Molina and Boris Kogan
An approach is presented for maintaining high availability in a replicated database system with a failure prone communications network. The status of the network is assumed to change dynamically making the detection of partitions infeasible. The approach is based on restricting the data items transactions can access and on special requirements placed on update propagation. (19 pages, December 1986)

TR-065-86 Data Caching in an Information Retrieval System
Hector Garcia-Molina, Rafael Alonso, Daniel Barbara and Soraya Abad
Currently existing computer communication networks give users access to an ever growing number of information retrieval systems. Some of those services are provided by commercial enterprises (examples are Dow Jones and The Source), while others are research efforts (such as the Boston Community Information System). In many cases these systems are accessed from personal or medium size computers which usually have available sizable amounts of local storage. To improve the response time of user queries, it becomes desirable to cache data at the user's site. However, to reduce the overhead of maintaining multiple copies, it may be appropriate to allow copies to diverge in a controlled fashion. This makes it possible to propagate updates to the copies efficiently, e.g., when the system is lightly loaded, when communication tariffs are lower, or by batching together updates. It also makes it possible to access the copies even when the communication lines or the central site are down. In this paper we present the notion of quasi-copies which embodies the ideas sketched above. We also define the types of deviations that seem useful, and discuss the available implementation strategies. (35 pages, December 1986)

TR-066-86 Planarity Testing of Doubly Periodic Infinite Graphs
Kazuo Iwano and Kenneth Steiglitz
This paper describes an efficient way to test the VAP-free (Vertex Accumulation Point free) planarity of one- and two-dimensional dynamic graphs. Dynamic graphs are infinite graphs consisting of an infinite number of basic cells connected regularly according to labels in a finite graph called a static graph. Dynamic graphs arise in the design of highly regular VLSI circuits, such as systolic arrays and digital signal processing chips. We show that VAP-free planarity testing of dynamic graphs can be done efficiently by making use of their regularity. First, we will establish necessary conditions for VAP-free planarity of dynamic graphs. Then we show the existence of a finite graph which is planar if and only if the original dynamic graph is VAP-free planar. From this it follows that VAP-free planarity testing of one- and two-dimensional dynamic graphs is asymptotically no more difficult than planarity testing of finite graphs, and thus can be done in linear time. (26 pages, December 1986)

TR-067-86 Models and Measurements of File System Performance
Arvin Park and Richard J. Lipton
File system buffering strategies can produce unexpected system behavior. For instance, writing one byte to a file can take more than twice as long as writing 4096 bytes to the same file. To examine this curious phenomenon and other aspects of file system performance, we develop models for file system behavior which factor in contributions of processor speed and buffer cache organization. These models agree very nicely with a set of performance measurements conducted on a VAX-11/750 running the UNIX 4.3 BSD operating system. We use these same models to predict the performance of recently proposed parallel mass storage architectures. We demonstrate that these architectures can potentially provide orders of magnitude more file system bandwidth than conventional non-parallel systems now provide. (13 pages, December 1986)

TR-068-86 Optimal Compaction of Multiple Two-Component Channels under River Routing
F. L. Heng and Andrea S. LaPaugh
We develop an $O(kn^3)$ time algorithm to establish the tradeoff graph of minimum total separation (i.e. width) versus spread (i.e. length) given k parallel river routing channels, each bounded by a single

component at the top and bottom, and a total of n nets. This solves the two dimensional compaction problem for a special case of slicing structured layout: a single hierarchical level structure with single layer interconnections between adjacent components. It serves as a first cut toward solving the two dimensional layout compaction problem in a slicing structure with one layer interconnection. (10 pages, December 1986)

TR-069-86 Designing Algorithms
Robert E. Tarjan
The quest for efficiency in computational methods yields not only fast algorithms, but also insights into the problems being solved. Such insights can produce problem-solving methods that combine speed with elegance, simplicity, and generality. This theme is illustrated with two examples: an algorithm for testing planarity of graphs, and a self-adjusting form of binary search tree. (29 pages, December 1986)

TR-070-87 SAGAS
Hector Garcia-Molina and Kenneth Salem
Long lived transactions (LLTs) hold on to database resources for relatively long periods of time, significantly delaying the termination of shorter and more common transactions. To alleviate these problems we propose the notion of a saga. A LLT is a saga if it can be written as a sequence of transactions that can be interleaved with other transactions. The database management system guarantees that either all the transactions in a saga are successfully completed or compensating transactions are run to amend a partial execution. Both the concept of saga and its implementation are relatively simple, but they have the potential to improve performance significantly. We analyze the various implementation issues related to sagas, including how they can be run on an existing system that does not directly support them. We also discuss techniques for database and LLT design that make it feasible to break up LLTs into sagas. (18 pages, January 1987)

TR-071-87 An Improved Upper Bound for Sorting on Non-partitionable Superposed Parallel Buses
Bruce W. Arden and Toshio Nakatani
The paper presents $O(n \log n)$ and $O(n \log \log n)$ sorting methods on $n \times n$ *non-partionable* superposed parallel buses. For merging and sorting on n processors connected by a linear bus, an optimal method based on enumeration is developed. It takes $3n/2$ and $2n$ steps, which are $O(\log n)$ and $O(\log^2 n)$ improvements from bitonic merge and sort respectively. For two dimensional sorting on superposed parallel buses, three different methods are considered: bitonic, shear, and reverse sort. Improvements in time complexity of $O(\log n)$ using the first two methods and of $O(\log^2 n / \log \log n)$ using the third method are obtained. Also time complexity measures based on the bus bandwidth are presented. A final discussion on VLSI optimality for the three sorting schemes is included. (23 pages, January 1987)

TR-072-87 Optimal Selection on Non-partitionable Superposed Parallel Buses
Bruce W. Arden and Toshio Nakatani
This paper presents an optimal selection algorithm on $n \times n$ *non-partitionable* superposed parallel buses. For arbitrary k $(1 \leq k \leq n^2)$ we can find the k-th smallest item in $O(n)$ time on $n \times n$ processors, where each processor has one data item, and is connected to row and column buses. This performance is shown to be optimal within a constant factor. This algorithm is $non-adaptive$ in the sense that the algorithm does not depend on the data items to select. The algorithm can be adapted to a mesh-connected computer with the same asymptotic time performance. (21 pages, January 1987)

TR-073-87 Distributed Computing Research at Princeton - 1986
Robert Abbott, Rafael Alonso, Luis Cova, Hector Garcia-Molina, Boris Kogan, Kriton Kyrimis, Frank Pittelli, Patricia Simpson, Annemarie Spauster, Kenneth Salem
In this note we briefly summarize the distributed computing research we performed in the year 1986. In general terms, our emphasis was on studying and implementing mechanisms for *efficient and reliable* computing and data management. Our work can be roughly divided into seven categories: the implementation of a highly reliable database system, the implementation of a high availability database system, dynamic vote reassignment, a mechanism for dealing with long-lived transactions, real time database systems, caching in information systems, and load balancing. Due to space limitations, we concentrate on describing our own work and we do not survey the work of other researchers in the field. For survey information and references, we refer readers to some of our reports. (13 pages, January 1987)

TR-074-87 IOStone: A Synthetic File System Performance Benchmark

Arvin Park and Richard J. Lipton

We have developed a portable benchmark program which measures file system performance on *typical* system loads. Our program accomplishes this by generating a string of file system requests which is representative of measured system loads. Instead of isolating a particular aspect of file system performance such as disk access speed, or channel bandwidth, our program measures performance of the entire file system which includes components of disk performance, CPU performance on file system tasks, and buffer cache performance. This single metric can act as a valuable comprehensive measure of file system performance. Measurements that we have made indicate that the balance between CPU performance and file system performance varies greatly across different computer systems. If an optimal balance between these two capabilities exists few system actually achieve this. (13 pages, January 1987)

TR-075-87 Performance Through Memory

Hector Garcia-Molina, Arvin Park and Lawrence R. Rogers

Two of the most important parameters of a computer are its processor speed and physical memory size. We study the relationship between these two parameters by experimentally evaluating the intrinsic memory and processor requirements of various applications. We also explore how hardware prices are changing the cost effectiveness of these two resources. Our results indicate that several important applications are "memory-bound," i.e., can benefit more from increased memory than from a faster processor. (28 pages, January 1987)

TR-076-87 Recovery in a Triple Modular Redundant Database System

Frank Pittelli and Hector Garcia-Molina

In a Triple Modular Redundant (TMR) database system the database is fully replicated at three computers. All transactions are executed at all nodes in the same relative order. The system can tolerate the arbitrary failure of a single computer since the correct data can be obtained from the two operating copies. After a failure, it is important to repair the computer so that the system can tolerate additional future failures. Repair in this case involves getting a correct and up to date copy of the database, without halting the two operational nodes. In this paper we analyze this database recovery problem. We describe a solution that has been implemented on an experimental TMR system running on SUNs/120 workstations. we also present performance results that illustrate the cost of recovery. (21 pages, January 1987)

TR-077-87 An Algorithm for Segment-Dragging and its Implementation

Bernard Chazelle

Given a collection of points in the plane, pick an arbitrary horizontal segment and move it vertically until it hits one of the points (if at all). This form of segment-dragging is a common operation in computer graphics and motion-planning. It can also serve as a building block for multidimensional data structures. This note describes a new approach to segment-dragging which yields a simple and efficient solution. The data structure requires $O(n)$ storage and $O(n \log n)$ preprocessing time, and each query can be answered in $O(\log n)$ time, where n is the number of points in the collection. The method is best understood as the end result of a sequence of transformations applied to a simple but inefficient starting solution. (29 pages, January 1987)

TR-078-87 Some Problems on Doubly Periodic Infinite Graphs

Kazuo Iwano

We show that finding weak components, finding an Eulerian path, and testing 2-colorability of two-dimensional doubly periodic graphs can be done in polynomial time with respect to the size of the static graph. (35 pages, February 1987)

TR-079-87 Re-opening Closures

Andrew W. Appel

There are two different commonly-used evaluation methods for functional languages: normal-order graph reduction, and call-by-value execution of closure code. The former is usually more expensive per operation, but has the capability of partially evaluating functions before they are applied. The latter method usually leads to faster execution – and is thus used in most compilers — but can't "optimize" functions before they are called. The different advantages of the two methods are particularly visible in the evaluation of higher-order functions. After a higher-order function is applied to one argument, the graph-reducer can begin evaluation, while the closure-code evaluator must wait until all arguments

are present. On the other hand, because the closure-code evaluator executes the native code of the computer, it usually outperforms the graph-reducer. The two evaluation algorithms can be combined to take advantage of the best behaviors of both. Fragments from programs that are already executing can be extracted, reduced, and re-compiled. This is done with an operator, reduce, that is semantically transparent: — reduce(f) does not change the behaviour of the program fragment f, but can make f much more efficient. This applies not just to functions f in the original program, but to functions constructed at runtime. (8 pages, February 1987)

TR-080-87 Concise Specifications of Locally Optimal Code Generators
Andrew W. Appel
Dynamic programming allows locally optimal instruction selection for expression trees. More importantly, the algorithm allows concise and elegant specification of code generators. Aho, Ganapathi, and Tjiang have built the Twig code-generator-generator, which produces dynamic-programming code-generators from grammar-like specifications. Encoding a complex architecture as a grammar for dynamic-programming code-generator-generator shows the expressive power of the technique. Each instruction, addressing mode, register and class can be expressed individually in the grammar. Twig specifications for the VAX and MC68020 are described, and the corresponding code generators select very good (and under the right assumptions, optimal) instruction sequences. Limitations and possible improvements to the specification language are discussed. (42 pages, February 1987)

TR-081-87 Solving Minimum-Cost Flow Problems by Successive Approximation
Andrew V. Goldberg and Robert E. Tarjan
We introduce a framework for solving minimum-cost flow problems. Our approach measures the quality of a solution by the amount that the complementary slackness conditions are violated. We show how to extend techniques developed for the maximum flow problem to improve the quality of a solution. This framework allows us to achieve $O(\min(n^3, n^{5/3}m^{2/3}, nm\log n)\log(nC))$ running time. (12 pages, February 1987)

TR-082-87 The Complexity of Cutting Complexes
Bernard Chazelle, Herbert Edelsbrunner and Leonidas J. Guibas
This paper investigates the combinatorial and computational aspects of certain extremal geometric problems in two and three dimensions. Specifically, we examine the problem of intersecting a convex subdivision with a line in order to maximize the number of intersections. A similar problem is to maximize the number of intersected facets in a cross-section of a three-dimensional convex polytope. Related problems concern maximum chains in certain families of posets defined over the regions of a convex subdivision. In most cases we are able to prove sharp bounds on the asymptotic behavior of the corresponding extremal functions. We also describe polynomial algorithms for all the problems discussed. (51 pages, March 1987)

TR-083-87 Performance of VLSI Engines for Lattice Computations
Steven D. Kugelmass, Richard Squier and Kenneth Steiglitz
We address the problem of designing and building efficient custom VLSI-based processors to do computations on large multi-dimensional lattices. The design tradeoffs for two architectures which provide practical engines for lattice updates are derived and analyzed. We find that I/O constitutes the principal bottle-neck of processors designed for lattice computations, and we derive upper bounds on throughput for lattice updates based on Hong and Kung's graph-pebbling argument that models I/O. In particular we show that $R = 0(BS^{\frac{1}{d}})$ where R is the site update rate, B is the main memory bandwidth, S is the processor storage, and d is the dimension of the lattice. (17 pages, March 1987)

TR-084-87 Computational Geometry — Then and Now
David P. Dobkin
In this paper we explore the development of solutions to five key problems in the computational geometry of the plane. The problems we consider were chosen on the basis of their longevity, their significance in computational geometry and the existence of reasonable solutions to each. In each case, the problem has been actively considered for a decade by both practitioners and theoreticians. The set of problems gives an accurate overview of the problems and methods of computational geometry (in the plane). In most cases, the solutions are practical and we will describe implementation issues. Where appropriate, we discuss extensions to higher dimensions. (22 pages, March 1987)

TR-085-87 The Application of Workstation Caching to Information Systems
Patricia Simpson and Rafael Alonso
Information retrieval (IR) systems provide individual remote access to centrally managed data. The current proliferation of personal computer systems, as well as advances in storage and communication technology, have created new possibilities for designing information systems which are easily accessible, economical, and responsive to user needs. This paper outlines methods of integrating personal computers (PCs) into large information systems, with emphasis on effective use of the storage and processing capabilities of these computers. In particular we discuss means for caching retrieved data at PC-equipped user sites, noting that caching in this environment poses unique problems. An event-driven simulation program is described which models information system operation. This simulator is being used to examine caching strategies. Our studies show that the limiting factors in caching effectiveness are the speed of the communication channel and the proportion of session time during which that channel is idle. When these values are low, unrestricted caching can seriously degrade performance. However, when they are sufficiently high, caching becomes a viable means of distributing data to users. (24 pages, March 1987)

TR-086-87 Monotone Bipartite Graph Properties are Evasive
Andrew Chi-Chih Yao
A Boolean function P from $\{0,1\}^t$ into $\{0,1\}$ is said to be evasive, if every decision tree algorithm for evaluating P must examine at least t arguments in the worst case. It was known that any nontrivial monotone bipartite graph property on vertex set $V \times W$ must be evasive, when $|V| \cdot |W|$ is a power of a prime number. In this paper, we prove that every nontrivial monotone bipartite graph property is evasive. (7 pages, April 1987)

TR-087-87 Altruistic Locking: A Strategy for Coping With Long Lived Transactions
Kenneth Salem, Hector Garcia-Molina and Rafael Alonso
Long lived transactions (LLTs) hold on to database resources for relatively long periods of time, significantly delaying the completion of shorter and more common transactions. To alleviate these problems we propose an extension to two-phase locking, called altruistic locking, whereby LLTs can release their locks early. Transactions that access this released but uncommitted data run in the wake of the LLT and must follow special locking and commit rules. Additional performance improvements can be obtained if the LLT predeclares its access set (this is only an option). Altruistic locking guarantees serializability and allows transactions to access the database in any order. In addition to presenting the protocol, we discuss how LLTs can be automatically analyzed in order to extract the access pattern information used by altruistic locking. (23 pages, April 1987)

TR-088-87 The Processor Identity Problem
Richard J. Lipton and Arvin Park
In this paper we pose the problem of establishing unique identities for a group of N identical processors which possess a common shared memory to which asynchronous read and write operations can be performed. We introduce a set of protocols which successfully assign unique integers from the set $\{0, 1, 2, \ldots, N-1\}$ to each processor. By applying these simple protocols to a shared memory multiprocessor system under development at Princeton, we have eliminated the need for hardwired addresses, or customized software for individual processing nodes. Individual processing nodes can now be installed or replaced without tedious configuration work. We have thereby greatly improved system modularity. (24 pages, April 1987)

TR-089-87 Primitives for the Manipulation of Three-Dimensional Subdivisions
David P. Dobkin and Michael J. Laszlo
A major impediment to the implementation of algorithms that manipulate 3-dimensional cell complexes and subdivisions is the non-existence of a suitable data structure. What is needed is a data structure which is powerful enough to model such objects while being simple enough to allow their manipulation in well defined ways. We focus attention here on the development of such data structure. Our structure is analogous (though one dimension higher) to Baumgart's winged-edge and Guibas and Stolfi's quad-edge data structures which are widely accepted for modelling 2-manifolds. Just as these structures can be used to represent both planar polygonal cell complexes in R^3 and surfaces of 4-polyhedra, our results can be viewed as similar to the work done by Guibas and Stolfi in deriving the quad-edge structure, one dimension higher. What we attempt to achieve in this paper is a blend between a derivation of the

data structure and a small set of primitive operators for its manipulation, the development of macro operations from these primitives, and the use of these macros in the first two applications mentioned above. The results of this paper are implementable and an effort to build them is currently underway. (25 pages, April 1987)

TR-090-87 Some Thoughts on Probabilistic Databases
Hector Garcia-Molina and Daryl Porter
It is often desirable to represent entities in a database whose properties cannot be deterministically classified. We develop a new data model that includes probabilities or confidences associated with the values of the attributes. Thus we can think of the attributes as random variables with probability distributions dependent on the entity the tuple purportedly describes. We study two sets of issues, one dealing with the proper model for probabilistic data and the other dealing with the choice of operators and language necessary to manipulate such data. (37 pages, April 1987)

TR-091-87 Update Propagation in Bakunin Data Networks
Boris Kogan and Hector Garcia-Molina
In a Bakunin network data are replicated at all nodes in order to achieve very high data availability. Nodes operate autonomously, executing transactions even when they are cut off from the rest of the system. This means that transactions may read stale data. In spite of this, serializability can be guaranteed by placing restrictions on the types of transactions that a node can execute. These restrictions take the form of an acyclic Read-Access Graph. In addition, a special update propagation protocol is used to ensure that all nodes see data updates in the same order. In this paper we present several such protocols. The protocols take advantage of the particular structure of the read-access graph to expedite propagation. We also define the notion of virtual serializability. It is a weaker form of serializability that allows speedier propagation. (14 pages, May 1987)

TR-092-87 A Process Migration Implementation for a Unix System
Rafael Alonso and Kriton Kyrimis
In this paper we describe the implementation of a process migration mechanism under version 3.0 of the Sun UNIX operating system. Processes that do not communicate with other processes and that do not take actions that depend on knowledge of the execution environment (such as the process id), can be moved from one machine to another while running, in a transparent way. This is achieved by signaling a process to stop, saving all the kernel and memory information that is necessary to restart the process and then, by using this information, restarting the process on the new machine. This new functionality requires minor kernel modifications as well as the creation of a new signal and a new system call. (18 pages, May 1987)

TR-093-87 Simulating Digital Circuits with One Bit Per Wire
Andrew W. Appel
Conventional digital circuit simulators represent circuits using linked data structures, using one or more pointers per connection. To simulate a circuit of N nodes requires space proportional to $N \log N$ bits. Many circuits have a hierarchical or repetitive nature, so their specifications can be significantly smaller than the circuits themselves. This paper shows that such circuits can be simulated in space equal to one bit of memory per wire of the circuit, plus space proportional to the (smaller) size of the specification; that is, the space required is only $O(N)$ bits. The algorithm has been implemented; measurements of its efficiency are given. (13 pages, May 1987)

TR-094-87 Computational Geometry in a Curved World
Diane L. Souvaine (Thesis)
Computational geometry as a field deals with the algorithmic aspects of all geometric problems. But the majority of the results obtained heretofore have been focused on objects defined with straight lines and flat faces, in part because a computational geometry of curved objects seemed significantly more complex. The major result of this dissertation is to show that curved objects can indeed be processed efficiently. We extend the results of straight-edged computational geometry into the curved world by defining a pair of new geometric objects, the *splinegon* and the *splinehedron*, as curved generalizations of the polygon and polyhedron. We identify three distinct techniques for extending polygon algorithms to splinegons: the carrier polygon approach, the bounding polygon approach, and the direct approach. By these methods, large groups of algorithms for polygons can be extended as a class to encompass these new objects.

In general, if the original polygon algorithm has time complexity $O(f(n))$, the comparable splinegon algorithm has time complexity at worst $O(Kf(n))$ where K represents a constant number of calls to a series of primitive procedures on individual curved edges. These techniques apply also to splinehedra. In addition to presenting the general methods, we state and prove a series of specific theorems. Problem areas include convex hull computation, diameter computation, intersection detection and computation, kernel computation, monotonicity testing, and monotone decomposition, among others. (126 pages, October 1986)

TR-095-97 Some Techniques for Geometric Searching with Implicit Set Representations
Bernard Chazelle
There are many efficient ways of searching a set when all its elements can be represented in memory. Often, however, the domain of the search is too large to have each element stored separately and some implicit representation must be used. Whether it is still possible to search efficiently in these conditions is the underlying theme of this paper. We look at several occurrences of this problem in computational geometry and we propose various lines of attack. In the course of doing so, we improve the solutions of several specific problems; for example, computing order statistics, performing polygonal range searching, testing algebraic predicates, etc. (24 pages, June 1987)

TR-096-87 Computing on a Free Tree Via Complexity-Preserving Mappings
Bernard Chazelle
The relationship between linear lists and free trees is studied. We examine a number of well-known data structures for computing functions on linear lists and show that they can be cononically transformed into data structures for computing the same functions defined over free trees. This is used to establish new upper bounds on the complexity of several query-answering problems. (32 pages, June 1987)

TR-097-87 A Standard ML Compiler
Andrew W. Appel and David B. MacQueen
Standard ML is a major revision of earlier dialects of the functional language ML. We describe the first compiler written for Standard ML in Standard ML. The compiler incorporates a number of novel features and techniques, and is probably the largest system written to date in Standard ML. Great attention was paid to modularity in the construction of the compiler, leading to a successful large-scale test of the modular capabilities of Standard ML. The front end is useful for purposes other than compilation, and the back end is easily retargetable (we have code generators for the VAX and MC68020). The module facilities of Standard ML were taken into account early in the design of the compiler, and they particularly influenced the environment management component of the front end. For example, the symbol table structure is designed for fast access to opened structures. The front end of the compiler is a single phase that integrates parsing, environment management, and type checking. The middle end uses a sophisticated decision tree scheme to produce efficient pattern matching code for functions and case expressions. The abstract syntax produced by the front end is translated into a simple lambda-calculus-based intermediate representation that lends itself to easy case analysis and optimization in the code generator. Special care was taken in designing the runtime data structures for fast allocation and garbage collection. We describe the overall organization of the compiler and present some of the data representations and algorithms used in its various phases. We conclude with some lessons learned about the ML language itself and about compilers for modern functional languages. (23 pages, June 1987)

TR-098-87 Lower Bounds for Shellsort
Mark Allen Weiss (Thesis)
Shellsort is a simple classic algorithm that runs competitively on both mid-sized and nearly sorted files. It uses an increment sequence, the choice of which can drastically affect the algorithm's running time. Since we would like an optimal sorting algorithm and a trivial lower bound for Shellsort is $N^* \times$ (# of increments), we require that the size of the increment sequence is $O(\log N)$, where N is the size of the file to be sorted. These increment sequences also tend to perform the best in practice, because of this lower bound. For some time, the complexity of Shellsort was thought to be well understood, but Sedgewick was able to provide an increment sequence that lowered the upper bound, and then Incerpi and Sedgewick extended the result to give an even better upper bound. In both papers, the authors left as open the problem of whether their bounds were tight. We prove that Sedgewick's bound is tight by analyzing the time required to sort a particularly bad permutation. Extending this proof technique seems to lead to a lower bound that matches the upper bound of Incerpi and Sedgewick for not only their

increment sequence, but also for a wide class of other increment sequences. Additionally, we show that the permutations which make Shellsort run slowly are counterexamples to a conjecture that shaker-sort, a network sorting algorithm proposed by Incerpi and Sedgewick, runs in $O(N \log N)$ time, again for a large class of increment sequences, including all those that have thus far been proposed. (73 pages, June 1987)

TR-099-87 Concurrency Controls for Global Procedures in Federated Database Systems
Rafael Alonso, Hector Garcia-Molina and Kenneth Salem
A federated database system is a collection of autonomous databases cooperating for mutual benefit. Global procedures can access several databases, but controlling concurrent database accesses by them is problematic. In particular, each database has its own (possibly different) concurrency control mechanism, and it must remain independent from the global controls. Furthermore, it may be difficult for the global controls to know exactly what granules (e.g., records or pages) are touched by each database access. In this paper we discuss the concurrency control problem for federated database systems, and suggest two mechanisms that may satisfy the requirements. (13 pages, June 1987)

TR-100-87 Some Thoughts on Data Sharing Among Autonomous Cooperating Database Systems
Rafael Alonso and Hector Garcia-Molina
As our society becomes increasingly information dependent, there will be ever greater pressures for information sharing among separate entities. Organizations will find that, in order to cooperate efficiently, they will have to share data with their partners. The rapid spread of networking technology will speed this process along, since it will become more convenient, faster, and less expensive than ever to communicate with others. However, although the benefits of cooperation are self-evident, it seems clear also that many organizations will not be willing to surrender autonomy over their data as the price of cooperation. In this paper we describe our current thoughts about the issues involved in sharing information effectively among a very large number of independent database systems. A major point of our work is to preserve the autonomy of the participants while still allowing efficient sharing. This work has as a starting point the concepts behind federated databases, but we are expanding these ideas to apply to systems with much larger numbers of participants than usually considered, supporting very high transaction rates, and allowing for a greater degree of heterogeneity than in previous research. (9 pages, June 1987)

TR-101-87 Quasi-Copies: Efficient Data Sharing for Information Retrieval Systems
Rafael Alonso, Daniel Barbara, Hector Garcia-Molina and Soraya Abad
Currently, a variety of information retreival systems are available to potential users. These services are provided by commercial enterprises (such as Dow Jones and The Source), while others are research efforts (the Boston Community Information System). While in many cases these systems are accessed from personal computers, typically no advantage is taken of the computing resources of those machines (such as local processing and storage). In this paper we explore the possibility of using the user's local storage capabilities to cache data at the user's site. This would improve the response time of user queries albeit at the cost of incurring the overhead required in maintaining multiple copies. In order to reduce this overhead it may be appropriate to allow copies to diverge in a controlled fashion. This would not only make caching less costly, but would also make it possible to propagate updates to the copies more efficiently, e.g., when the system is lightly loaded, when communication tariffs are lower, or by batching together updates. Just as importantly, it also makes it possible to access the copies even when the communication lines or the central site are down. Thus, we introduce the notion of quasi-copies which embodies the ideas sketched above. We also define the types of deviations that seem useful, and discuss the available implementation strategies. (18 pages, June 1987)

TR-102-87 A Prototype for Research on Heterogeneous Database Systems
Rafael Alonso
When two organizations decide to share the information in their databases, it is not uncommon for them to find that the data stored in their respective computer systems are kept in incompatible database management systems. To allow users to query the combined database in a straightforward manner, a mechanism is needed that will mask the details of each particular system and present a homogeneous interface to the collection of database systems. A heterogeneous database system enables users to transparently access the data contained in a multiplicity of differing databases. In this paper we describe some of the issues that must be addressed in the implementation of heterogeneous database systems, and in particular we focus on resolving the conflicts that arise in data integration. We also describe the

design of a prototype that is currently being built to conduct research on this problem, one of whose salient features is the use of an embedded expert system to aid in data integration. (7 pages, June 1987)

TR-103-87 Amortized Analysis of Algorithms for Set Union with Backtracking
Jeffery Westbrook and Robert E. Tarjan
Mannila and Ukkonen have studied a variant of the classical disjoint set union (equivalence) problem in which an extra operation, called deunion, can undo the most recently performed union operation not yet undone. They proposed a way to modify standard set union algorithms to handle deunion operations. We analyze several algorithms based on their approach. The most efficient such algorithms have amortized running time of $O(\log n / \log\log n)$ per operation, where n is the total number of elements in all the sets. These algorithms use $O(n \log n)$ space, but the space usage can be reduced to $O(n)$ by a simple change. We prove that any separable pointer-based algorithm for the problem requires $\Omega(\log n / \log\log n)$ time per operation, thus showing that our upper bound is tight. (20 pages, May 1987)

TR-104-87 Algorithms for Two Bottleneck Optimization Problems
Harold N. Gabow and Robert E. Tarjan
A bottleneck optimization problem on a graph with edge costs is the problem of finding a subgraph of a certain kind that minimizes the maximum edge cost in the subgraph. The bottleneck objective contrasts with the more common objective of minimizing the sum of edge costs. We propose fast algorithms for two bottleneck optimization problems. For the problem of finding a bottleneck spanning tree in a directed graph of n vertices and m edges, we propose an $O(\min\{n \log n + m\,,\, m \log^* n\})$-time algorithm. For the bottleneck maximum cardinality matching problem, we propose an $O((n \log n)^{1/2} m)$-time algorithm. (9 pages, May 1987)

TR-105-87 A Fast Parametric Maximum Flow Algorithm
Giorgio Gallo, Michael D. Grigoriadis and Robert E. Tarjan
The classical maximum flow problem sometimes occurs in settings in which the arc capacities are not fixed but are functions of a single parameter, and the goal is to find the value of the parameter such that the corresponding maximum flow or minimum cut satisfies some side condition. Finding the desired parameter value requires solving a sequence of related maximum flow problems. We show that the recent maximum flow algorithm of Goldberg and Tarjan can be extended to solve an important class of such parametric maximum flow problems, at the cost of only a constant factor in its worst case time bound. Faster algorithms for a variety of combinatorial optimization problems follow from our result. (40 pages, July 1987)

TR-106-87 Finding Minimum-Cost Circulations by Successive Approximation
Andrew V. Goldberg and Robert E. Tarjan
We develop a new approach to solving minimum-cost circulation problems. Our approach combines methods for solving the maximum flow problem with successive approximation techniques based on cost scaling. We measure the accuracy of a solution by the amount that the complementary slackness conditions are violated. We propose a simple minimum-cost circulation algorithm, one version of which runs in $O(n^3 \log(nC))$ time on an n-vertex network with integer arc costs of absolute value at most C. By incorporating sophisticated data structures into the algorithm, we obtain a time bound of $O(nm \log(n^2/m) \log(nC))$ on a network with m arcs. A slightly different use of our approach shows that a minimum-cost circulation can be computed by solving a sequence of $O(n \log(nC))$ blocking flow problems. A corollary of this result is an $O(n^2(\log n)\log(nC))$-time, n-processor parallel minimum-cost circulation algorithm. Our approach also yields strongly polynomial minimum-cost circulation algorithms. Our results provide evidence that the minimum-cost circulation problem is not much harder than the maximum flow problem. We believe that a suitable implementation of our method will perform extremely well in practice. (53 pages, July 1987)

TR-107-87 Finding Minimum-Cost Circulations by Canceling Negative Cycles
Andrew V. Goldberg and Robert E. Tarjan
A classical algorithm for finding a minimum-cost circulation consists of repeatedly finding a residual cycle of negative cost and canceling it by pushing enough flow around the cycle to saturate an arc. We show that a judicious choice of cycles for canceling leads to a polynomial bound on the number of iterations in this algorithm. This gives a very simple strongly polynomial algorithm that uses no scaling. A variant of the algorithm that uses dynamic trees runs in $O(nm(\log n)\min\{\log(nC), m \log n\})$ time on a network

of n vertices, m arcs, and arc costs of maximum absolute value C. This bound is comparable to those of the fastest previously known algorithms. (16 pages, July 1987)

TR-108-87 A Linear-Time Algorithm for Finding a Minimum Spanning Pseudoforest
Harold N. Gabow and Robert E. Tarjan
A pseudoforest is a graph each of whose connected components is a tree or a tree plus an edge; a spanning pseudoforest of a graph contains the greatest number of edges possible. This paper shows that a minimum cost spanning pseudoforest of a graph with n vertices and m edges can be found in $O(m+n)$ time. This implies that a minimum spanning tree can be found in $O(m)$ time for graphs with girth at least $\log(i)n$ for some constant i. (6 pages, July 1987)

TR-109-87 Relaxed Heaps: An Alternative to Fibonacci Heaps
James R. Driscoll, Harold N. Gabow, Ruth Shrairman and Robert E. Tarjan
The relaxed heap is a priority queue data structure that achieves the same amortized time bounds as the Fibonacci heap – a sequence of m decrease-key and n delete-min operations takes time $O(m + n \log n)$. A variant of relaxed heaps achieves similar bounds in the worst case — $O(1)$ time for decrease-key and $O(\log n)$ for delete-min. A relaxed heap is a type of binomial queue that allows heap order to be violated. (16 pages, July 1987)

TR-110-87 Distributed Reachability Analysis for Protocol Verification Environments
Sudhir Aggarwal, Rafael Alonso and Costas Courcoubetis
a topic of importance in the area of distributed algorithms is the efficient implementation of formal verification techniques. Many such techniques are based on coupled finite state machine models, and reachability analysis is central to their implementation. SPANNER is an environment developed at AT&T Bell Laboratories, and is based on the selection/resolution model (S/R) of coupled finite state machines. It can be used for the formal specification and verification of computer communication protocols. In SPANNER, protocols are specified as coupled finite state machines, and analyzed by proving properties of the joint behavior of these machines. In this last step, reachability analysis is used in order to generate the "product" machine from its components, and constitutes the most time consuming part of the verification process. In this paper we investigate aspects of distributing reachability over a local area network of workstations, in order to reduce the time needed to complete the calculation. A key property which we exploit in our proposed design is that the two basic operations performed during reachability, the new state generation, and the state tabulation, can be performed asynchronously, and to some degree independently. Furthermore, each of these operations can be decomposed into concurrent subtasks. We provide a description of the distributed reachability algorithm we are currently in the process of implementing in SPANNER, and an investigation of the scheduling problems we face. (27 pages, August 1987)

TR-111-87 Faster Scaling Algorithms for Network Problems
Harold N. Gabow and Robert E. Tarjan
This paper presents algorithms for the assignment problem, the transportation problem and the minimum cost flow problem of operations research. The algorithms find a minimum cost solution, but run in time close to the best-known bounds for the corresponding problems without costs. For example, the assignment problem (equivalently, minimum cost matching on a bipartite graph) can be solved in $O(\sqrt{n}m \log(nN))$ time, where n, m and N denote the number of vertices, number of edges and largest magnitude of a cost; costs are assumed to be integral. The algorithms work by scaling. As in the work of Goldberg and Tarjan, in each scaled problem an approximate optimum solution is found, rather than an exact optimum. (31 pages, August 1987)

TR-112-87 An Experimental Evaluation of Load Balancing Strategies
Rafael Alonso
In this paper we report on an experimental study of load balancing strategies for a network of workstations. We have implemented a load balancing mechanism on which a variety of strategies have been tested. Our current implementation runs on a local area network composed of a variety of Sun workstations. Some of the difficulties involved in developing a practical load balancing mechanism are described, as well as our suggested solutions to those problems. Among the issues addressed are how to avoid instabilities in the decision policy (especially in the case where the load balancing mechanism must deal with incomplete load information), and how to implement load sharing in an environment where the

individual processors are owned by separate users (clearly, on such networks, policies that evenly spread the load throughout the system are not appropriate). (9 pages, September 1987)

TR-113-87 Delaunay Graphs are Almost as Good as Complete Graphs
David P. Dobkin, Steven J. Friedman and Kenneth J. Supowit
Let S be any set of N points in the plane and let $DT(S)$ be the graph of the Delaunay triangulation of S. For all points a and b of S, let $d(a,b)$ be the Euclidean distance from a to b and let $DT(a,b)$ be the length of the shortest path in $DT(S)$ from a to b. We show that there is a constant $c(\leq \frac{1+\sqrt{5}}{2}\pi \approx 5.08)$ independent of S and N such that

$$\frac{DT(a,b)}{d(a,b)} < c\,.$$

(16 pages, June 1987)

TR-114-87 Covering Minima and Lattice Point Free Convex Bodies
Ravi Kannan and Laszlo Lovasz
The covering radius of a convex body K (with respect to a lattice L) is the least factor by which the body needs to be blown up so that its translates by lattice vectors cover the whole space. The covering radius and related quantities have been studied extensively in the Geometry of Numbers (mainly for convex bodies symmetric about the origin). In this paper, we define and study the "covering minima" of a general convex body. The covering radius will be one of these minima; the "lattice width" of the body will be the reciprocal of another. We derive various inequalities relating these minima. These imply bounds on the width of lattice point free convex bodies. We prove that every lattice-point-free body has a projection whose volume is not much larger than the determinant of the projected lattice. (35 pages, November 1987)

TR-115-87 A Density Theorem for Purely Iterative Zero Finding Methods
Joel Friedman
In this paper we prove that a wide class of purely iterative root finding methods work for all complex valued polynomials with a positive probability depending only on the method and the degree of the polynomial. More precisely, if we consider the set of polynomials with roots in the unit ball, then for fixed degree the area of convergent points in the bass of radius 2 is bounded below by some constant for any purely iterative method $z + 1 \leftarrow T_f(z_i)$ where $T_f(z)$ is a rational function of z and f and its derivatives for which (1) ∞ is repelling fixed point for all f of degree > 1 and (2) $T_f(z)$ depends only on z and f's roots and commutes with linear maps on the complex plane. (13 pages, November 1987)

TR-116-87 Random Polynomials and Approximate Zeros of Newton's Method
Joel Friedman
In this paper we study the area of approximate zeros for Newton's method, i.e. the set of points for which Newton's method converges doubly exponentially fast starting from the first iterate. We obtain a bound in terms of the separation of the roots. We then apply this various probability distributions on polynomials of fixed degree, obtaining estimates on the probability that the approximate zero region has small area. For polynomials of the form $f(z) = \sum_{i=0}^{d} a_i z^i$ with $a_d = 1$ and complex valued at a_i chosen independently and uniformly in the unit ball for $i < d$ we get that with high probability the area of approximate zeros in the ball of radius $3 > d^{-2-\epsilon}$ for any $\epsilon > 0$. (48 pages, October 1987)

TR-117-87 Exploiting Symmetries for Low-Cost Comparison of File Copies
Daniel Barbara and Hector Garcia-Molina
In this paper we examine a new technique for comparison of remotely located file copies. With this new technique up to two differing pages can be located and any number of multiple differing pages can be detected. The technique uses a communication overhead of $O(\log^2(N))$, where N is the number of pages in the file. It is based on a set of symmetries of an hypercube with dimension $\log(N)$. (14 pages, November 1987)

TR-118-87 Improved Time Bounds for the Maximum Flow Problem
Ravindra K. Ahuja, James B. Orlin and Robert E. Tarjan
Recently, Goldberg proposed a new approach to the maximum network flow problem. The approach yields a very simple algorithm running in $O(n^3)$ time on n-vertex networks. Incorporation of the dynamic tree data structure of Sleator and Tarjan yields a more complicated algorithm with a running time of $O(nm\log(n^2/m))$ on m-edge networks. Ahuja and Orlin developed a variant of Goldberg's algorithm

that uses scaling and runs in $O(nm + n^2 \log U)$ time on networks with integer edge capacities bounded by U. In this paper we obtain a modification of the Ahuja-Orlin algorithm with a running time of $O(nm + n^2 \frac{\log U}{\log\log U})$. We show that the use of dynamic trees in this algorithm reduces the time bound to $O(nm \log(\frac{n \log U}{m \log\log U} + 2))$. This result demonstrates that the combined use of scaling and dynamic trees results in speed not obtained by using either technique alone. (18 pages, September 1987)

TR-119-87 Crash Recovery for Memory-Resident Databases
Kenneth Salem and Hector Garcia-Molina
A main memory database system holds all data in volatile semiconductor memory. The crash recovery manager of such a system logs changes to disk and periodically checkpoints the database to non-volatile storage. The manager is different from that of a disk-based system because after a failure the main memory and not the disk must be restored to a consistent state. Furthermore, the performance of the recovery manager is more critical since it is the only component of the system that performs expensive I/O operations. In this paper we study the algorithms for the performing main memory crash recovery. Their performance is compared via a detailed model of the critical resources for this environment: CPU overhead and disk bandwidth. The performance results suggest a set of "rules of thumb" for selecting a crash recovery strategy. (52 pages, November 1987)

TR-120-87 Sharing Jobs Among Independently Owned Processors
Rafael Alonso and Luis L. Cova
In many distributed systems it is possible to share the processing capabilities among the nodes. To accomplish this goal, a number of load balancing algorithms have been proposed in the literature. The purpose of this type of algorithms is to redistribute the system workload with the objective of equalizing the workload at each node. In a network of independently owned processors (e.g., a network of workstations), load balancing schemes cannot consider the whole network as one unit and thus try to optimize the overall performance. Instead, they have to consider the needs of the resource owners. For this type of environment load sharing is a more appropriate goal. Load sharing has been accomplished in some systems in an "all or nothing" fashion, i.e., if a node is idle then it becomes a candidate for executing a remote workload, otherwise it is not. This style of sharing is too restrictive in an environment where most resources are underutilized. We present a scheme that replaces this "all or nothing" approach with a gradual one, i.e., where each machine in the network determines the amount of sharing it is willing to do. The scheme, called High-Low, makes sure that the service provided to local jobs of a lightly loaded node does not deteriorate by more than predefined amount. It simultaneously helps improve the service at heavily loaded nodes. We empirically show that load sharing in a network of workstations can be effective even with a simple-minded allocation of jobs. (20 pages, November 1987)

TR-121-87 On Selecting the Second Largest with Median Tests
Andrew Chi-Chih Yao
Let $V_k(n)$ be the minimas complexity of selecting the $k-th$ largest of n numbers $x_1, x_2, \ldots, x_n$ by pairwise comparisons $x_i : x_j$. It is well known that $V_2(n) = n - 2 + \lceil \lg n \rceil$. In this paper we study $V_2'(n)$, the minimax complexity of selecting the second largest, when tests of the form "Is x_i the median of $\{x_i, x_j, x_k\}$?" are also allowed. It is proved that $n - 3 + \lceil \lg n \rceil \leq V_2'(n) \leq n - 2 + \lceil \lg n \rceil$. Furthermore, both upper and lower bounds are achieved for infinitely many n. (10 pages, November 1987)

TR-122-87 Reflections in Curved Surfaces
E. S. Panduranga (Thesis)
Reflections add a new dimension to the realism of computer generated scenes. The only existing method for implementing reflections satisfactorily is ray-tracing. However, it has major drawbacks, the most obvious being its expensive CPU requirements. The objective of this dissertation is to gain an understanding of the nature of reflections, and to develop alternate methods for rendering them by exploiting their geometric coherence. We develop a disciplined science for characterizing reflections in curved surfaces. There is no hope for closed form solutions to rendering them since they are algebraically intractable. However, we describe three techniques to handle the situation: contour tracing, special purpose numerical methods, and spline approximation. We extend four existing algorithms to include reflections, namely, the painter's algorithm, the z-buffer algorithm, the scan plane algorithm, and what we call the curvilinear trapezoid method. Methods and scope for parallelizing these algorithms are also discussed. A comparative study of these algorithms is attempted to outline their suitability under various computing

environments. The computing environment is thus a significant factor for the choice of an algorithm. Our efforts are focused on scenes composed of spheres but more general scenes are possible. (130 pages, October 1987)

TR-123-87 Reliable Broadcast in Networks with Nonprogrammable Servers
Hector Garcia-Molina, Boris Kogan and Nancy Lynch
The problem of implementing reliable broadcast in ARPA-like computer networks is studied. The environment is characterized by the absence of any multicast facility on the communications subnetwork level. Thus, broadcast has to be implemented directly on hosts. A reliable broadcast protocol is presented and evaluated on several important performance criteria. (33 pages, November 1987)

TR-124-87 A Probabilistic Model for Clock Skew
Steven D. Kugelmass and Kenneth Steiglitz
A probabilistic model for the accumulation of clock skew in synchronous systems is presented. Using this model, we derive upper bounds for expected skew, and its variance, in tree distribution systems with N synchronously clocked processing elements. We apply these results to two specific models for clock distribution. In the first, which we call metric-free, the skew in a buffer stage is Gaussian with a variance independent of wire length. In this case the upper bound on skew grows as $\Theta(\log N)$ for a system with N processing elements. The second, metric, model, is intended to reflect VLSI constraints: the clock skew in a stage is Gaussian with a variance proportional to wire length, and the distribution tree is an H-tree embedded in the plane. In this case the upper bound on expected skew is $\Theta(\sqrt{N \log N})$ for a system with N processors. Thus the probabilistic model is more optimistic than the deterministic summation model of Fisher and Kung, which predicts a clock skew $\Theta(N)$ in this case, and is also consistent with their lower bound of $\Omega(\sqrt{N})$ for planar embeddings. We have estimates of the constants of proportionality, as well as the asymptotic behavior, and we have verified the accuracy of our estimates by simulation. (14 pages, November 1987)

TR-125-87 A Data Structure for Manipulating Three-Dimensional Subdivisions
Michael Jay Laszlo (Thesis)
A major impediment to the implementation of algorithms that manipulate 3-dimensional cell complexes and subdivisions is the non-existence of a suitable data structure. What is needed is a data structure which is powerful enough to model such objects while being simple enough to allow their manipulation in well defined ways. We focus attention here on the development of such data structure. Our structure is analogous (though one dimension higher) to Baumgart's winged-edge and Guibas and Stolfi's quad-edge data structures which are widely accepted for modelling 2-manifolds. Just as these structures can be used to represent both planar polygonal cell complexes in R^3 and surfaces of 4-polyhedra, our results can be viewed as similar to the work done by Guibas and Stolfi in deriving the quad-edge structure, one dimension higher. (126 pages, August 1987)

TR-126-87 Checkpointing Memory-Resident Databases
Kenneth Salem and Hector Garcia-Molina
A main memory database system holds all data in semiconductor memory. For recovery purposes, a backup copy of the database is maintained in secondary storage. The checkpointer is the component of the crash recovery manager responsible for maintaining the backup copy. Ideally, the checkpointer should maintain an almost-up-to-date backup while interfering as little as possible with the system's transaction processing activities. We present several algorithms for maintaining such a backup database, and compare them using an analytic model. Our results show some significant performance differences among the algorithms, and illustrate some of the performance tradeoffs that are available in designing such a checkpointer. (22 pages, December 1987, revised June 1988)

TR-127-87 A Shared Memory Architecture for Distributed Computing
Arvin Park (Thesis)
This thesis presents shared memory as a new paradigm for distributed computing architectures. It explains how shared memory can be used to simplify the programming of distributed applications, and how such systems can be efficiently implemented in a restricted domain of distributed computing known as moderately-coupled systems. The viability of moderately-coupled shared memory computing architectures is demonstrated through the implementation of Mind Meld, a versatile interconnection which can be used to combine separate microprocessor systems into a single moderately-coupled shared

memory distributed system. Application codes written for Mind Meld have proven the system's ability to improve resource sharing and to enhance parallel processing capabilities. They have also demonstrated that shared memory can greatly simplify the task of distributed system programming. Development of the Mind Meld system has generated new problems in multiprocessor system initialization. The Processor Identity Problem seeks to assign unique identities to a collection of identical processors which communicate through shared memory. Protocols that solve the Processor Identity Problem are developed and analyzed. (85 pages, December 1987)

TR-128-87 Analysis of Algorithms for the Configuration of Wafer Scale Linear Arrays in the Presence of Defects

Dimitris A. Doukas and Andrea S. LaPaugh

Wafer Scale Integration (WSI) is a new technology using Very Large Scale Integration (VLSI). The goal is to implement an entire system in a single silicon wafer containing the equivalent of hundreds of present-day chips. Given the high density and large number of elements in a wafer scale, we expect some components to be defective due to fabrication errors. Methods are needed to configure the good components into a working system. In this paper we examine four algorithms to connect the good elements in a linear WSI systolic array. We present experimental results obtained by simulation of the algorithms. We also present an analysis of performance which is applicable to three of the algorithms. The analysis is in term of the number of good elements we expect to utilize. (38 pages, December 1987)

TR-129-87 Scheduling Real-time Transactions

Robert Abbott and Hector Garcia-Molina

Scheduling transactions with real-time requirements presents many new problems. In this paper we discuss solutions for two of these problems: what is a reasonable method for modeling real-time constraints for database transactions? Traditional hard real-time constraints (e.g., dead-lines) may be too limited. Many transactions have soft deadlines and a more flexible model is needed to capture these soft time constraints. The second problem we address is scheduling. Time constraints add a new dimension to concurrency control. Not only must a schedule be serializable but it also should meet the time constraints of all the transactions in the schedule. (10 pages December 1987)

TR-130-88 Analysis of a Simple Yet Efficient Convex Hull Algorithm

Mordecai Golin and Robert Sedgewick

This paper is concerned with a simple, rather intuitive preprocessing step that is likely to improve the average-case performance of any convex hull algorithm. For n points randomly distributed in the unit square, we show that a simple linear pass through the points can eliminate all but $O(\sqrt{n})$ of the points by showing that a simple superset of the remaining points has size $c\sqrt{n} + o(\sqrt{n})$. We give a full implementation of the method, which should be useful in any practical application for finding convex hulls. Most of the paper is concerned with an analysis of the number of points eliminated by the procedure, including derivation of an exact expression for c. Extensions to higher dimensions are also considered. (11 pages, January 1988)

TR-131-88 Rotation Distance, Triangulations and Hyperbolic Geometry

Daniel Sleator, Robert E. Tarjan and William P. Thurston

A rotation in a binary tree is a local restructuring that changes the tree into another tree. Rotations are useful in the design of tree-based data structures. The rotation distance between a pair of trees is the minimum number of rotations needed to convert one tree into the other. In this paper we establish a tight bound of $2n - 6$ on the maximum rotation distance between two n-node trees for all large n, using volumetric arguments in hyperbolic 3-space. Our proof also gives a tight bound on the minimum number of tetrahedra needed to dissect a polyhedron in the worst case, and reveals connections among binary trees, triangulations, polyhedra, and hyperbolic geometry. (37 pages, January 1988)

TR-132-88 A Fast Las Vegas Algorithm for Triangulating a Simple Polygon

Kenneth L. Clarkson, Robert E. Tarjan and Christopher J. Van Wyk

We show how to use random sampling to triangulate a simple polygon in nearly linear expected time. (9 pages, January 1988)

TR-133-88 Real-time Concurrent Collection on Stock Multiprocessors

Andrew W. Appel, John R. Ellis and Kai Li

This paper presents the first garbage-collection algorithm that is efficient, real-time, concurrent, runs on

stock commercial uniprocessors and multiprocessors, and requires no change to compilers. Our algorithm is related to Baker's algorithm, in which objects are copied from "from space" to "to space" while the mutator is running. We maintain the invariant that registers point only into to-space by arranging to get a page fault from the virtual memeory system if certain pages are fetched from. We have data structures to allow us to scan the pages of to-space and the stack in arbitrary order. We have implemented the algorithm, and find that it performs efficiently in practice as well as in theory. (22 pages, February 1988)

TR-134-88 Lower Bounds to Randomized Algorithms for Graph Properties
Andrew Chi-Chih Yao

For any property P on n-vertex graphs, let $C(P)$ be the minimum number of edges needed to be examined by any decision tree algorithm for determining P. In 1975, Rivest and Vuillemin settled the Aanderra-Rosenberg Conjecture, proving that $C(P) = \Omega(n^2)$ for every nontrivial monotone graph property P. An intriguing open question is whether the theorem remains true when randomized algorithms are allowed. In this paper we show that $\Omega(n(\log n)^{1/12})$ edges need to be examined by any randomized algorithm for determining any nontrivial monotone graph property. (22 pages, February 1988)

TR-135-88 Properties of Multistage Interconnection Networks
Abdou S. Youssef (Thesis)

Regular rectangular multistage interconnection networks that are complete and have the single path property are increasingly important in large parallel computing systems. The efficiency of such networks is critical to the overall system performance, and it depends on the structure, functional capabilities and routing control of the network. Much of the previous research has focused on specific networks. The objective of this dissertation is to study and characterize an important class of such networks. In particular, the relationships between network functionality and topology, switch size, control and modularity. The one-to-one correspondence between topology and functionality is shown, necessary and sufficient totological conditions for a network to realize all the permutations of another network are established, and optimal algorithms to decide if a network realizes all the permutations of another are developed. The single path property makes control via control tags potentially efficient. Two different control schemes are introduced where the control tags are the same as, or a function of, destination tags. The structure of these "easy-to-control" networks is shown to be recursive. Based on this recursiveness, the networks of several interesting network subclasses are shown to be functionally equivalent, namelty, the subclass of doubly controllable networks, that is, easy-to-control from left to right and from right to left, the subclass of modular networks where all the stages are identical, and the subclass of r-ary networks, where the stages permute and transform r-ary digits. These single path networks do not realize all permutations. Multiple path Benes networks do but are a slow-to-control alternative. Accordingly, a new scheme for fast control of 3-stage Benes networks is introduced and studied, and several problems are shown to fit the new scheme, namely, bitonic sorting, FFT, tree algorithms and matrix computations. Also, the random walk computation is parallelized and shown to fit this control scheme. (216 pages, February 1988)

TR-136-88 On the Complexity of Partial Order Productions
Andrew Chi-Chih Yao

Let $P = (<_P, Y)$ be a partial order on a set $Y = \{y_1, y_2, \ldots, y_n\}$ of n elements. The problem of *P-production* is, given an input of n distinct numbers $x_1, x_2, \ldots, x_n$, find a permutation σ of $(1, 2, \ldots, n)$ such that $y_i <_P y_j$ implies $x_{\sigma(i)} < x_{\sigma(j)}$. Let $C(P), \bar{C}(P)$ be, respectively, the minimum number and the minimum average number of binary comparisons $x_i : x_j$ needed by any decision-tree algorithm to produce P. We prove that $C(P) = \Theta(\bar{C}(P))$. As an intermediate result, we show that $C(P) = O(\log_2(n!/\mu(P)) + n)$, where $\mu(P)$ is the number of permutations consistent with P, proving a conjecture of Saks. (14 pages, February 1988)

TR-137-88 Tight Lower Bounds for Shellsort
Mark Allen Weiss and Robert Sedgewick

Shellsort is a simple classic algorithm that runs competitively on both mid-sized and nearly sorted files. It uses an increment sequence, the choice of which can drastically affect the algorithm's running time. Due to the results of Pratt, the running time of Shellsort was long thought to be $\Theta(N^{3/2})$ for increment sequences that are "almost geometric," however, recent results have lowered the upper bound substantially, although the new bounds were not known to be tight. In this paper, we show that an increment sequence given by Sedgewick is $\Theta(N^{4/3})$ by analyzing the time required to sort a particularly

bad permutation. Extending this proof technique to various increment sequences seems to lead to lower bounds that in general always match the known upper bounds and suggests that Shellsort runs in $\Theta(N^{1+\epsilon/\sqrt{\log N}})$ for increment sequences of practical interest, and that no increment sequence exists that would make Shellsort optimal. (14 pages, February 1988)

TR-138-88 On Selecting the k Largest with Median Tests
Andrew Chi-Chih Yao
Let $W_k(n)$ be the minimax complexity of selecting the k largest elements of n numbers $x_1, x_2, \ldots, x_n$ by pairwise comparisons $x_i : x_j$. It is well known that $W_2(n) = n - 2 + \lceil \lg n \rceil$, and $W_k(n) = n + (k-1)\lg n + O(1)$ for all fixed $k \geq 3$. In this paper we study $W'_k(n)$, the minimax complexity of selecting the k largest, when tests of the form "Is x_i the median of $\{x_i, x_j, x_t\}$?" are also allowed. It is proved that $W'_2(n) = n + (k-1)\lg_2 n + O(1)$ for all $k \geq 3$. (9 pages, March 1988)

TR-139-88 Near-Optimal Time-Space Tradeoff for Element Distinctness
Andrew Chi-Chih Yao
It was conjectured Borodin et al. (*J. Comput. System Sci.* **22**, 1981, 351–64) that to solve the element distinctness problem requires $TS = \Omega(n^2)$ on a comparison-based branching program using space S and time T, which, if true, would be close to optimal since $TS = O(n^2 \log n)$ is achievable. Recently, Borodin et. al. (*SIAM J. on Comput.* **16**, 1987, 97–9) showed that $TS = \Omega(n^{3/2}(\log n)^{1/2})$. In this paper, we show a near-optimal tradeoff $TS = \Omega(n^{2-\epsilon(n)})$ where $\epsilon(n) = O(1/(\log n)^{1/2})$. (12 pages, March 1988)

TR-140-88 Compiling Separable Recursions
Jeffrey F. Naughton
In this paper we consider evaluating queries on relations defined by a combination of recursive rules. We first define separable recursions. We then give a specialized algorithm for evaluating selections on separable recursions. Like the Magic Sets and Counting algorithms, this algorithm uses selection constants to avoid examining irrelevant portions of the database; however, on some simple recursions this algorithm is $O(n)$, whereas the Magic Sets algorithm is $\Omega(n^2)$ and the Generalized Counting Method is $\Omega(2^n)$. (27 pages, March 1988)

TR-141-88 Benchmarking Multi-Rule Recursion Evaluation Strategies
Jeffrey F. Naughton
This paper presents an empirical comparison of the Semi-Naive, Generalized Magic Sets, Generalized Counting, and Separable query evaluation strategies as applied to queries on multi-rule recursions. I used each of the methods to evaluate queries over randomly generated relations. For each query there is a critical density range. If the base relations are below the density range, Generalized Magic Sets, Generalized Counting, and Separable provide roughly equal performance and are significantly better than Semi-Naive. Above the density range, Generalized Magic Sets degrades to Semi-Naive, Generalized Counting is much worse that Semi-Naive, while Separable is still significantly better than Semi-Naive. This suggests that special purpose algorithms such as Separable can greatly improve the performance of a recursive query processor. (25 pages, March 1988)

TR-142-88 Runtime Tags Aren't Necessary
Andrew W. Appel
Many modern programming environments use tag bits at runtime to distinguish objects of different types. This is particularly common in systmes with garbage collection, since the garbarge collector must be able to distinguish pointers from non-pointers, and to learn the length of records pointed to. The use of tag bits leads to inefficiency. They take up memory (though generally not too much); but more important, tag bits must be stripped off of data before arithmetic operations are performed, and re-attached to the data when it is stored into memory. This takes either extra instructions at runtime, or special tag-handling hardware, or both. This paper shows how the use of tag bits, record descriptor words, explicit type parameters, and the like can be avoided in languages (like ML) with static polymorphic typechecking. Though a form of tag will still be required for user-defined variant records, all other type information can be encoded once — in the program — rather than replicated many times in the data. This can lead to savings both in space and time. (10 pages, March 1988)

TR-143-88 Simple Generational Garbage Collection and Fast Allocation
Andrew W. Appel
Generational garbage collection algorithms achieve efficiency because newer records point to older records;

the only way an older record can point to a newer record is by a store operation to a previously-created record, and such operations are rare in many languages. A garbarge collector that concentrates just on recently allocated records can take advantage of this fact. This paper presents a simple, efficient, low-overhead version of generational garbarge collection that is suitable for implementation in a Unix environment. In addition, a scheme for quick record allocation is described. (11 pages, March 1988)

TR-144-88 Recipes for Geometry & Numerical Analysis — Part I: An Empirical Study
David Dobkin and Deborah Silver
Geometric computations, like all numerical procedures, are extremely prone to roundoff effor. However, virtually none of the numerical analysis literature directly applies to geometric calculations. Even for line intersection, the most basic geometric operation, there is no robust and efficient algorithm. Compounding the difficulties, many geometric algorithms perform iterations of calculations reusing previously computed data. In this paper, we explore some of the main issues in geometric computations and the methods that have been proposed to handle roundoff errors. In particular, we focus on one method and apply it to a general iterative intersection problem. Our initial results seem promising and will hopefully lead to robust solutions for more complex problems of computational geometry. (15 pages, March 1988)

TR-145-88 Minimean Optimal Key Arrangements in Hash Tables
Andrew Chi-Chih Yao
For an open-address hash function h and a set A of n keys, let $C_h(A)$ be the expected retrieval cost when the keys are arranged to minimize the expected retrieval cost in a full table. It is shown that, asymptotically for large n, when h satisfies a certain doubly dispersive property, as is the case for uniform hashing or double hashing, $C_h(A) = O(1)$ with probability $1 - o(1)$ for a random A. (18 pages, March 1988)

TR-146-88 Scheduling Real-time Transactions: A Performance Evaluation
Robert Abbott and Hector Garcia-Molina
Managing transactions with real-time requirements presents many new problems. In this paper we focus on two: How can we schedule transactions with deadlines? How do the real-time constraints affect concurrency control? We describe a new group of algorithms for scheduling real-time transactions which produce serializable schedules. We present a model for scheduling transactions with deadlines on a single processor memory resident database system, and evaluate the scheduling through detailed simulation experiments. (19 pages, February 1988)

TR-147-88 Supporting Probabilistic Data in a Relational System
Hector Garcia-Molina and Daryl Porter
It is often desirable to represent in a database entities whose properties cannot be deterministically classified. We develop a new data model that includes probabilities or confidences associated with the values of the attributes. Thus we can think of the attributes as random variables with probability distributions dependent on the entity the tuple purportedly describes. This new model offers a richer descriptive language allowing the database to more accurately reflect the uncertain real world. It also offers a new interpretation of information incompleteness. We study three sets of issues: the proper model for probabilistic data, the semantics of probabilistic data, and the choice of operators and language necessary to manipulate such data. (23 pages, February 1988)

TR-148-88 An Optimal Algorithm for Intersecting Line Segments in the Plane
Bernard Chazelle and Herbert Edelsbrunner
The main contribution of this work is an $O(n+k)$ time algorithm for computing all k intersections among n line segments in the plane. This time complexity is easily shown to be optimal. Within the same asymptotic cost our algorithm can also construct the subdivision of the plane defined by the segments and compute which segment (if any) lies right above (or below) each intersection and each endpoint. The algorithm has been implemented and performs very well. The storage requirement is on the order of $n + k$ in the worst case, but it is considerably lower in practice. To analyze the complexity of the algorithm we use an amortization argument based on a new combinatorial theorem on line arrangements. (70 pages, April 1988)

TR-149-88 Algorithms for Finding a Maximum Bipartite Subgraph for Special Classes of Graphs
Susan S. Yeh and Andrea S. LaPaugh
In this paper we present efficient polynomial algorithms for finding a maximum bipartite subgraph for

special classes of graphs. In general this problem is NP-complete, and it remains NP-complete even when the graph is restricted to be planar. However, putting further restrictions on the graph makes this problem tractable. We have developed algorithms for finding a maximum bipartite subgraph for proper circular-arc, circular-arc, permutation, and split graphs. In numerous occasions, we used one technique, namely dynamic programming, for the development of the algorithms. Furthermore, the algorithms we designed, with slight modifications, can be adapted to solve the maximum bipartite subgraph problem which includes a nontrivial subset of vertices. (27 pages, April 1988)

TR-150-88 Minimizing Expansions of Recursions
Jeffrey F. Naughton and Yehoshua Sagiv
In recent years function-free horn clauses have received a lot of attention as database query languages. Recursive definitions in such a language are particularly problematic in that they are hard to implement efficiently. As most evaluation procedures at least implicitly evaluate the expansion of the recursion, it is natural to consider optimizing the recursion by minimizing its expansion. In this paper we show how attempts to minimize expansions of recursions lead naturally to the issues of recursively redundant predicates and bounded recursions. We review current results, prove several new results about inter-element redundancy in expansions, and show how both recursively redundant predicates and bounded recursions are closely related to the existence of various types of paths in a graph constructed from the rule. (22 pages, April 1988)

TR-151-88 Telematics Research at Princeton — 1987
Robert Abbott, Rafael Alonso, Daniel Barbara, Luis Cova, Hector Garcia-Molina, Boris Kogan, Kriton Kyrimis, Kenneth Salem, Patricia Simpson and Annemarie Spauster
In this note we briefly summarize the database and distributed computing research we performed in the year 1987. In general terms, our emphasis was on studying and implementing mechanisms for efficient and reliable computing and data management. Our work can be roughly divided into 9 categories: the implementation of a high availability database system, real time database systems, multicast protocols, distributed file comparison, altruistic locking, information exchange networks, data caching in information systems, process migration, and load balancing. Due to space limitations, we concentrate on describing our own work and we do not survey the work of other researchers in the field. For survey information and references, we refer readers to some of our reports. (12 pages, May 1988)

TR-152-88 An Efficient Algorithm for Finding the CSG Representation of a Simple Polygon
David Dobkin, Leonidas Guibas, John Hershberger and Jack Snoeyink
We consider the problem of converting boundary representations of polyhedral objects into constructive-solid-geometry (CSG) representations. The CSG representations for a polyhedron P are based on the half-spaces supporting the faces of P. For certain kinds of polyhedra this problem is equivalent to the corresponding problem for simple polygons in the plane. We give a new proof that the interior of each simple polygon can be represented by a monotone boolean formula based on the half-planes supporting the sides of the polygon and using each such half-plane only once. Our main contribution is an efficient and practical $O(n \log n)$ algorithm for doing this boundary-to-CSG conversion for a simple polygon of n sides. We also prove that such nice formulae do not always exist for general polyhedra in three dimensions. (10 pages, May 1988)

TR-153-88 Searching for Empty Convex Polygons
David P. Dobkin, Herbert Edelsbrunner and Mark H. Overmars
A key problem in computational geometry is the identification of subsets of a point set having particular properties. We study this problem for the properties of convexity and emptiness. We show that finding empty triangles is related to the problem of determininng pairs of vertices that see each other in star-shaped polygon. A linear time algorithm for this problem which is of independent interest yields an optimal algorithm for finding all empty triangles. This result is then extended to an algorithm for finding empty convex r-gons ($r > 3$) and for determining a largest empty convex subset. Finally, extensions to higher dimensions are mentioned. (10 pages, May 1988)

TR-154-88 Faster Algorithms for the Shortest Path Problem
Ravindra K. Ahuja, Kurt Mehlhorn, James B. Orlin and Robert E. Tarjan
We investigate efficient implementations of Dijkstra's shortest path algorithm. We propose a new data structure, called the redistributive heap, for use in this algorithm. On a network with n vertices, m

edges, and nonnegative integer arc costs bounded by C, a one-level form of redistributive heap gives a time bound for Dijkstra's algorithm of $O(m + n \log C)$. A two-level form of redistributive heap gives a bound of $O(m + n \log C / \log \log C)$. A combination of a redistributive heap and a previously known data structure called a Fibonacci heap gives a bound of $O(m + n\sqrt{\log C})$. The best previously known bounds are $O(m + n \log n)$ using Fibonacci heaps alone and $O(m \log \log C)$ using the priority queue structure of Van Emde Boas, Kaas and Zijlstra. (14 pages, March 1988)

TR-155-88 Tight Bounds on the Stabbing Number of Spanning Trees in Euclidean Space
Bernard Chazelle
We tighten the analysis of a data structure for simplex range searching discovered recently by Welzl. Our main result is that any set of n points in E^d admits a spanning tree which cannot be cut by any hyperplane (or hypersphere) through more than roughly $n^{1-1/d}$ edges. This result yields quasi-optimal solutions to simplex range searching in the arithmetic model of computation. We also look at circular and polygonal range searching on a random access machine. Given n points in E^2, we derive a data structure of size $O(n \log n)$ for counting how many points fall inside a query convex k-gon (for arbitrary values of k). The query is $O(\sqrt{kn} \log n)$. If k is fixed once and for all (as in triangular range searching) then the storage requirement drops to $O(n)$. We also describe an $O(n \log n)$-size data structure for counting how many points fall inside a query circle in $O(\sqrt{n} \log^2 n)$ query time. (14 pages, May 1988)

TR-156-88 Architectures for Two-Dimensional Lattice Computations with Linear Speedup
Steven D. Kugelmass (Thesis)
Many problems are characterized by the fact that they deal with data values distributed on a regular mesh, or *lattice*. They arise in a wide variety of applications such as image processing, computer vision, the solution of partial differential equations, and the simulation of cellular automata. This dissertation explores theoretical and practical questions in the design of massively parallel machines for lattice processing. (110 pages, June 1988)

TR-157-88 A Fast Las Vegas Algorithm for Triangulating a Simple Polygon
Kenneth L. Clarkson, Robert E. Tarjan and Christopher J. Van Wyk
We present a randomized algorithm that triangulates a simple polygon on n vertices in $O(n \log n)$ expected time. The averaging in the analysis of running time is over the possible choices made by the algorithm; the bound holds for any input polygon. (13 pages, May 1988)

TR-158-88 Query Processing in a Heterogeneous Retrieval Network
Patricia Simpson
The concept of a large-scale information retrieval network incorporating heterogeneous retrieval systems and users is introduced, and the necessary components for enabling term-based searching of any database by untrained end-users are outlined. We define a normal form for expression of queries, show that such queries can be automatically produced, if necessary, from a natural-language request for information, and give algorithms for translating such queries, with little or no loss of expressiveness, into equivalent queries on both Boolean and term-vector type retrieval systems. We conclude with a proposal for extending this approach to arbitrary database models. (17 pages, May 1988)

TR-159-88 Multiprocessor Main Memory Transaction Processing
Kai Li and Jeffrey F. Naughton
In this paper we describe an experiment designed to evaluate the potential transaction processing system performance achievable through the combination of multiple processors and massive memories. The experiment consisted of the design and implementation of a transaction processing kernel on stock multiprocessors. We found that with sufficient memory, multiple processors can greatly improve performance. A prototype implementation of the kernel on a pair of Firefly multiprocessors (each with five 1-MIP processors) runs the standard debit-credit benchmark at over 1000 transactions per second. (17 pages, June 1988)

TR-160-88 Negotiating Data Access in Federated Database Systems
Rafael Alonso and Daniel Barbara
The ever growing need for information is putting pressure on organizations to share data with their partners. However, although the different entities would like to share information, it is clear that each individual system administrator would like to preserve his or her control over the system. This concern with each system's autonomy has led to the notion of federated databases. Previous work in this area

has touched upon the topic of negotiating access in a federated database (i.e., determining what local information may be access by any particular remote user), but we feel that the topic has not been studied in depth. In this paper we propose a new scheme, based on the notion of quasi-copies, which may be used for interaction among autonomous databases. We also provide protocols for access negotiation, as well as simple cost models for estimating the expense involved in allowing remote access to information. One of the main advantages of our new scheme is that it provides a very precise way of establishing how much autonomy is given up by the owner of the information when he or she decides to share data. Finally, our approach may be used in both the case where the databases systems in question have a common query language (or there exist facilities for query translation), and when they do not. (14 pages, June 1988)

TR-161-88 Message Ordering in a Multicast Environment
Hector Garcia-Molina and Annemarie Spauster
A multicast group is a collection of processes that are the destinations of the same sequence of messages. These messages may originate at one or more source sites and the destination processes may run on one or more sites, not necessarily distinct. A multicast protocol ensures that the messages are delivered to the appropriate processes. Some applications require that the protocol provide some guarantees on the order in which messages are delivered. In this paper we characterize three ordering properties and discuss their solutions. We concentrate on the multiple group ordering property, which guarantees that two messages destined to two processes are delivered in the same relative order, even if they originate at different sources and are addressed to different multicast groups. We present a new protocol that solves the multiple group ordering problem. We address the issues of performance and reliability by providing comparisons with other techniques for ordering multicasts. In many cases this new algorithm solves the problem with greater efficiency than previous solutions without sacrificing reliability. (25 pages, June 1988)

TR-162-88 Copying Garbage Collection in the Presence of Ambiguous References
Andrew W. Appel and David R. Hanson
Garbage collection algorithms rely on invariants to permit the identification of pointers and to correctly locate accessible objects. These invariants translate into constraints on object layouts and programming conventions governing pointer use. There are recent variations of collectors in which the invariants are relaxed. Typically, rules governing pointer use are relaxed, and a "conservative" collection algorithm that treats all potential pointers as valid is used. Such pointers are "ambiguous" because integers and other data can masquerade as pointers. Ambiguous pointers cannot be modified and hence the objects they reference cannot be moved. Consequently, conservative collectors are based on mark-and-sweep algorithms. Copying algorithms, while more efficient, have not been used because they move objects and adjust pointers. This paper describes a variation of a copying garbage collector that can be used in the presence of ambiguous references. The algorithm constrains the layout and placement of objects, but not the location of referencing pointers. It simply avoids copying objects that are referenced directly by ambiguous pointers, reclaiming their storage on a subsequent collection when they are no longer ambiguously referenced. An implementation written in the ANSI C programming language is given. (9 pages, June 1988)

TR-163-88 A Tight Amortized Bound for Path Reversal
David Ginat, Daniel D. Sleator and Robert E. Tarjan
Path reversal is a form of path compression used in a disjoint set union algorithm and a mutual exclusion algorithm. We derive a tight upper bound on the amortized cost of path reversal. (5 pages, June 1988)

TR-164-88 Finding Minimum-Cost Flows by Double Scaling
Ravindra K. Ahuja, Andrew V. Goldberg, James B. Orlin and Robert E. Tarjan
Several researchers have recently developed new techniques that give fast algorithms for the minimum-cost flow problem. In this paper we combine several of these techniques to yield an algorithm running in $O(nm \log\log U \log(nC))$ time on networks with n vertices, m arcs, maximum are capacity U, and maximum arc cost magnitude C. The major techniques used are the capacity-scaling approach of Edmonds and Karp, the excess-scaling approach of Ahuja and Orlin, the cost-scaling approach of Goldberg and Tarjan, and the dynamic tree data structure of Sleator and Tarjan. For nonsparse graphs with large maximum arc capacity, we obtain a similar but slightly better bound. We also obtain a slightly better bound for the (noncapacitated) transportation problem. In addition, we discuss a capacity-bounding approach to the minimum-cost flow problem. (28 pages, June 1988)

TR-165-88 Load Balancing in Two Types of Computational Environments
Luis L. Cova

In many distributed systems it is possible to share the processing capabilities among the nodes. To accomplish this goal, a number of load balancing algorithms have been proposed in the literature. The purpose of this type of algorithm is to redistribute the system workload with the objective of equalizing the load at each node. In this report we discuss two types of distributed computing environments for which load balancing techniques yield increased performance: (1) pool of processors and (2) independently owned processors. Both environments are built around a loosely-coupled computer network, but in the former all computational nodes belong to one user community, while in the latter each machine belongs to a different individual or organization. We argue that these two environments should not be treated with the same type of load balancing algorithm due to their distinct nature. For the first environment we discuss a load balancing algorithm based on executing jobs at the least loaded processor in the network. For the second environment we introduce the concepts of lending and borrowing resources. For both environments we present empirical results to illustrate our approach to load balancing. (28 pages, January 1988)

TR-166-88 Lower Bounds on the Complexity of Polytope Range Searching
Bernard Chazelle

Polytope range searching is a central problem in multimensional searching, with applications to computer graphics, robotics, and database design. In its most elementary form, the problem can be stated as follows: Given a collection P of n weighted points in Euclidean d-space and a simplex q, compute the cumulative weight of $Pcapq$. The points are given once and for all and can be preprocessed. The simplex q, however, forms a query which must be answered on-line. We assume that the weights are chosen in a commutative semigroup and that the time to answer a query includes only the number of arithmetic operations performed by the algorithm. We prove that if m units of storage are available then the worst-case query time is $\Omega(n/\sqrt{m})$ in 2-space, and more generally, $\Omega((n/\log n)/m^{1/d})$ in d-space, if $d \geq 3$. These bounds also hold with high probability for a random set of points (drawn uniformly in the d-cube) and remains true if the queries are restricted to congruent copies of a fixed simplex. In the course of our investigation we also establish results of independent interest regarding a generalization of Heilbronn's problem. (31 pages, June 1988)

TR-167-88 Visibility and Intersection Problems in Plane Geometry
Bernard Chazelle and Leonidas J. Guibas

We develop new data structures for solving various visibility and intersection problems about a simple polygon P on n vertices. Among our results are a simple $O(n \log n)$ time algorithm for computing the illuminated subpolygon of P from a luminous side, and an $O(\log n)$ time algorithm for determining which side of P is first hit by a bullet fired from a point in a certain direction. The latter method requires preprocessing on P which takes time $O(n \log n)$ and space $O(n)$. The two main tools in attacking these problems are geometric duality on the two-sided plane and fractional cascading. (39 pages, June 1988)

TR-168-88 Optimizing Closure Environment Representation
Andrew W. Appel and Trevor T. Y. Jim

In lexically scoped languages with higher-order functions, any function may have free variables that are bound in the enclosing scope. The function can be compiled into machine code, but the values of the free variables will not be known until run time. Therefore a closure is used: a run-time data structure providing access to bindings of variables free in a given program fragment. Various schemes have been used to represent closures, from linked lists to flat vectors. Different representations will have different performance characteristics, notably in access time, creation time, storage space, and garbage collection. This paper describes some old representations and some new ones, and gives measurements of their efficiency. (7 Pages, July 1988)

TR-169-88 Vectorized Garbage Collection
Andrew W. Appel and Aage Bendiksen

Garbage collection can be done in vector mode on supercomputers like the Cray-2 and the Cyber 205. Both copying collection and mark-and-sweep can be expressed as breadth-first searches in which the "queue" can be processed in parallel. We have designed a copying garbage collector whose inner loop works entirely in vector mode. The only significant limitation of the algorithm is that if the size of the records is not constant, the implementation becomes much more complicated. We give performance

measurements of the algorithm as implemented for Lisp CONS cells on the Cyber 205. Vector-mode garbage collection performs up to 9 times faster than scalar-mode collection — a worthwhile improvement. (6 pages, July 1988)

TR-170-88 An Implementation of Reliable Broadcast Using an Unreliable Multicast Facility
Hector Garcia-Molina and Boris Kogan
Some computer communication networks provide a multicast facility for their users. This means that a host can hand a message to its server with more than one destination address. The network then tries to deliver the message to all specified destinations in an efficient way. However, it does not guarantee a reliable delivery. In this paper the problem of reliable broadcast on top of such a network is considered. The solution proposed makes use of the multicast facility to achieve efficiency. (17 pages, August 1988)

TR-171-88 Transitive Reduction in Parallel Via Branchings
Phillip Gibbons, Richard Karp, Vijaya Ramachandran, Danny Soroker and Robert E. Tarjan
We study the following problem: given a strongly connected digraph, find a minimal strongly connected spanning subgraph of it. Our main result is a parallel algorithm for this problem, which runs in polylog parallel time and uses $O(n^3)$ processors on a PRAM. Our algorithm is simple and the major tool it uses is computing a minimum-weigh branching with zero-one weights. We also present sequential algorithms for the problem that run in time $(m + n \cdot \log n)$. (16 pages, July 1988)

TR-172-88 On the Second Eigenvalue and Random Walks in Random d-Regular Graphs
Joel Friedman
(28 pages, August 1988)

TR-173-88 Bakunin Data Networks: Approach to Designing Highly Available Replicated Databases
Boris Kogan (Thesis)
Data replication in distributed databases is used to improve availability and provide faster read access. However, new problems for database management are introduced due to the need for maintaining the consistency of replicated copies. Dealing with communication failures is one of them. For example, when the network partitions, continued update activities may lead to data inconsistency. On the other hand, halting all updates until the network is reconnected is undesirable in many cases. This dissertation introduces a methodological framework for designing replicated databases that exhibit a high degree of availability (including the ability to make updates) in the face of communications failures that can partition the network. The framework is based on the simple notions of fragments and agents. It gives rise to a wide scope of options with varying degrees of data consistency and availability. The consistency criteria offered by these options include one-copy serializability as well as two new criteria: virtual serializability and fragmentwise serializability. (91 pages, October 1988)

TR-174-88 A Library for Incremental Update of Bitmap Images
David Dobkin, Eleftherios Koutsofios and Rob Pike
To achieve the maximum performance from bitmap displays, the screen must be used not just as an output device, but as a data structure that may cache computed images. In an interactive text or picture editor, that may mean converting the internal representation of what's being edited into a set of rectangles that tile the screen. Incremental updates of the image may then be done by rearranging some subset of the tiling using bitmap operations, independently of how the tiling was derived. We have taken the ideas used in the screen update algorithms for the sam text editor and generalized them so they may be applied to more structured documents than the simple character stream sam edits. The ideas have been tested by building a library and a simple interactive document editor that treat a document as a hierarchical structure that may include text, pictures, and variable spacing. The core of the library is operators to make incremental changes to the display while maintaining the hierarchical data structure that describes it. (8 pages, September 1988)

TR-175-88 Two-Dimensional Compaction: Computing the Shape Function of a Slicing Layout
Fook-Luen Heng (Thesis)
We investigate a two dimensional compaction scheme for a slicing layout. The compaction scheme treats wires as topological entities, i.e. only the paths of wires that connect terminals of the same nets are known but not their physical locations. Wires are not present physically in the layout during compaction, but they are expressed as constraints which describe the actual space required to accommodate them. This amounts to computing the shape function that captures the routing requirements of the layout.

We present an $O(k \cdot n^3)$ algorithm to compute the shape function of a stack of k two-component river routable channels with n nets, and a heuristic to approximate the shape function of a stack of multiple component river routable channels. We develop heuristics that use the algorithm and the heuristic above as their subroutines to approximate the shape function of a slicing layout under the restriction of river routing. The experimental results show the potential of the proposed compaction scheme. We study the asymptotic behavior of river routing and pitch aligning in uniform stacks and uniform arrays. We derive the condition under which river routing is better than pitch aligning for such uniform layouts. We discuss the problem of computing the shape function of a slicing layout with general channel routing. We use channel density as an estimate of channel width. We show an NP-completeness result for computing such a shape function. We present a pseudo-polynomial time algorithm to compute the shape function (using channel density as channel width) for a slicing layout of depth one, and we provide an efficient heuristic to compute such a shape function. Experimental results show that the heuristic produces shape functions which are very close to the exact shape function. (120 pages, October 1988)

TR-176-88 A Class of Randomized Strategies for Low-Cost Comparison of File Copies
Daniel Barbara and Richard J. Lipton
In this paper we present a class of algorithms for comparison of remotely located file copies that use randomized signatures. We are able to show a simple technique that sends on the order of $4^f \log(n)$ bits, where f is the number of pages in the file. We later show how to improve the bound in the number of bits sent, making them grow with f as $f \log(f)$ and with n as $\log(n) \log(log(n))$. A third class of algorithms is presented, in which the number of signatures grows with f as fr^f, where r can be made to approach 1. This class of techniques exhibit a worse asymptotic behavior, but they perform very well in practice. Previously published algorithms were aimed to diagnose 1 and 2 differing pages by sending $O(\log(n) \log(\log(n)))$ and $O \log^2(n) \log(\log(n)))$ bits respectively. Moreover, our techniques prove to be very competitive in practice sending less bits for the cases $f = 1$ and $f = 2$ respectively. (25 pages, September 1988)

TR-177-88 The Design of a Document Database
Chris Clifton, Hector Garcia-Molina and Robert Hagmann
In this paper, a Document Base Management System is proposed that incorporates conventional database and hypertext ideas into a document database. The Document Base operates as a server, users access the database through different application programs. The query language which applications use to retrieve documents is described. (14 pages, September 1988)

TR-178-88 Geometry, Graphics, and Numerical Analysis
Deborah E. Silver (Thesis)
Geometric computations (e.g. polygon/line intersections and detection, etc.), like all numerical procedures, are extremely prone to roundoff error. However, virtually none of the numerical analysis literature directly applies to geometric calculations. As a result, most of these applications resort to *ad hoc* methods to overcome the numerical difficulties. The goal of this research is to present and analyze systematic approaches to handle unreliability and enable more robust solutions to problems in applied computational geometry. (115 pages, October 1988)

TR-179-88 File Access Patterns
Carl Staelin
In this paper we examine patterns of file use which could be exploited by a file system to provide better service. We establish that many files have distinctive histories of access patterns. In particular, we are interested in finding which files are more likely to be accessed in the future, and also those files which are least likely to be accessed. These histories may be utilized by a hierarchical system to optimize file migration algorithms. (19 pages, September 1988)

TR-180-88 PRAM: A Scalable Shared Memory
Richard J. Lipton and Jonathan S. Sandberg
PRAM is a new scalable shared memory system. It is unlike any current shared memory system, precisely because PRAM can reach an inconsistent state. We show that standard programming techniques can avoid any effects of this inconsistency. The advantages of PRAM are that it is simpler, faster, and less expensive than existing systems. (13 pages, September 1988)

TR-181-88 A Deterministic View of Random Sampling and its Use in Geometry

Bernard Chazelle and Joel Friedman

The combination of divide-and conquer and random sampling has proven very effective in the design of fast geometric algorithms. A flurry of efficient probabilistic algorithms have been recently discovered, based on this happy marriage. We show that all those algorithms can be derandomized with only polynomial overhead. In the process we establish results of independent interest concerning the covering of hypergraphs and we improve on various probabilistic bounds in geometric complexity. For example, given n hyperplances in d-space and any integer r large enough, we show how to compute, in polynomial time a simplicial packing of size $O(r^d)$ which covers d-space, each of whose simplices intersects $O(n/r)$ hyperplanes. Also, we show how to locate a point among n hyperplanes in d-space in $O(\log n)$ query time, using $O(n^d)$ storage and polynomial preprocessing. (24 pages, September 1988)

TR-182-88 Allocation Without Locking

Andrew W. Appel

In a programming environment with both concurrency and automatic garbage collection, the allocation and initialization of a new record is a sensitive matter: if it is interrupted halfway through, the allocating process may be in a state that the garbage collector can't understand. In particular, the collector won't know which words of the new record have been initialized and which are meaningless (and unsafe to traverse). For this reason, parallel implementations usually use a locking or semaphore mechanism to ensure that allocation is an atomic operation. The locking significantly adds to the cost of allocation. This paper shows how allocation can run extremely quickly even in a multi-thread environment: open-coded, without locking. (3 pages, September 1988)

TR-183-88 Continuation-Passing, Closure-Passing Style

Andrew W. Appel and Trevor Jim

We implemented a continuation-passing style (CPS) code generator for ML. Our CPS language is represented as an ML datatype in which all functions are named and most kinds of ill-formed expressions are impossible. We separate the code generation into phases that rewrite this representation into ever-simpler forms. Closures are represented explicitly as records, so that closure strategies can be communicated from one phase to another. No stack is used. Our benchmark data shows that the new method is an improvement over our previous, abstract-machine based code generator. (11 pages, September 1988)

TR-184-88 Ordered and Reliable Multicast Communication

Hector Garcia-Molina and Annemarie Spauster

A multicast group is a collection of processes that are the destinations of the same sequence of messages. These messages may originate at one or more source sites and the destination processes may run on one or more sites, not necessarily distinct. A multicast protocol is responsible for the delivery of messages to the appropriate processes. Some applications require that the protocol provide some guarantees on the order in which messages are delivered. In addition, in many cases reliability of delivery is essential. In this paper we characterize three ordering properties and discuss their solutions. We concentrate on the multiple group ordering property, which guarantees that two messages destined to two processes are delivered in the same relative order, even if they originate at different sources and are addressed to different multicast groups. We present a new protocol that solves the multiple group ordering problem. We present extensive experimental results that illustrate the performance of our algorithm compared to other techniques for ordering multicasts. We address the issue of reliability by providing several alternatives for handling message loss and site crash recovery. We explore the usefulness of these with a comparison to the reliability guarantees of the other proposed solutions. In many cases our new algorithm solves the problem with greater efficiency than previous solutions without sacrificing reliability. (36 pages, October 1988)

TR-185-88 On Straight Selection Sort

Andrew Chi-Chih Yao

The expected value of $B_n = B'_n - (n-1)$, where B'_n is the number of right-to-left maxima encountered by the straight selection sort, is well known to be $(n+1)H_n - 2n$, but the variance of B_n has remained unanalyzed. In this paper, we derive an exact formula for the variance of B_n, and show that it is asymptotically equal to $\alpha n^{3/2}(1 + o(1))$, where $\alpha > 0$ is an explicitly defined constant. (66 pages, October 1988)

TR-186-88 A Parallel Algorithm for Finding A Blocking Flow in an Acyclic Network

Andrew V. Goldberg and Robert E. Tarjan

We propose a simple parallel algorithm for finding a blocking flow in an acyclic network. On an n-vertex, m-arc network, our algorithm runs in $O(n \log n)$ time and $O(nm)$ space using an m-processor EREW PRAM. A consequence of our algorithm is an $O(n^2(\log n)\log(nC))$-time, $O(nm)$-space, m-processor algorithm for the minimum-cost circulation problem, on a network with integer arc capacities of magnitude at most C. (9 pages, October 1988)

TR-187-88 Efficiency of the Primal Network Simplex Algorithm for the Minimum-Cost Circulation Problem
Robert E. Tarjan

We study the number of pivots required by the primal network simplex algorithm to solve the minimum-cost circulation problem. We propose a pivot selection rule with a bound of $n^{(\log n)/2+0(1)}$ on the number of pivots, for an n-vertex network. This is the first known subexponential bound. The network simplex algorithm with this rule can be implemented to run in $n^{(\log n)/2+0(1)}$ time. In the special case of planar graphs, we obtain a polynomial bound on the number of pivots and the running time. We also consider the relaxation of the network simplex algorithm in which cost- increasing pivots are allowed as well as cost-decreasing ones. For this algorithm we propose a pivot selection rule with a bound of $O(nm \cdot \min\{\log(nC), m \log n\})$ on the number of pivots, for a network with n vertices, m arcs, and integer arc costs bounded in magnitude by C. The total running time is $O(nm \log n \cdot \min\{(\log nC), m \log n\})$. This bound is competitive with those of previously known algorithms for the minimum-cost circulation problem. (42 pages, August 1988)

TR-188-88 Failure Recovery in Memory-Resident Transaction Processing Systems
Kenneth Salem (Thesis)

A main memory transaction processing system holds a complete copy of its database in semiconductor memory. We present and compare, in a common framework, a number of strategies for recovery management in main memory transaction processing systems. These include strategies for asynchronously checkpointing the primary (main memory) database copy, and for maintaining a transaction log. Though they are not directly concerned with recovery management, we also consider strategies for updating the primary database, since they affect the performance of the recovery manager. The recovery strategies are compared using an analytic performance model and a testbed implementation. The model computes two performance metrics: Processor overhead and recovery time. Processor overhead measures the impact of a recovery strategy during normal operation, i.e., the cost of preparing to recover from a failure. Recovery time is a measure of the cost of recovery once a failure has occurred. Generally, it is possible to reduce processor overhead by increasing recovery time, and vice versa. The model captures this tradeoff, the exact nature of which depends on which recovery strategies are used. Many of the recovery strategies have been implemented in a testbed, a working transaction processing system. The testbed allows recovery strategies to be combined and tested. It has been used to verify the performance model and to study other aspects of performance not considered in the model, such as data contention and transaction response times. The testbed runs on a large-memory VAX 11/785 using services provided by the Mach operating system. Our results indicate that the selection of a checkpointing strategy is the most critical decision in designing a recovery manager. In most situations, fuzzy, or unsynchronized, checkpointing strategies outperform highly sychronized alternatives. This is true even when synchronized checkpoints are combined with efficient logical logging strategies, which cannot be used with fuzzy checkpoints. (128 pages, December 1988)

TR-189-88 Simplified Linear-Time Jordan Sorting and Polygon Clipping
Khun Yee Fung, Tina M. Nicholl, Robert E. Tarjan and Christopher J. Van Wyk

The Jordan sorting problem is, given the intersection points of a Jordan curve with the x-axis in the order in which they occur along the curve, sort them into the order in which they occur along the x-axis. This problem arises in clipping a simple polygon against a rectangle (a "window") and in efficient algorithms for triangulating a simple polygon. Hoffman, Mehlhorn, Rosenstiehl, and Tarjan proposed an algorithm that solves the Jordan sorting problem in time linear in the number of intersection points, but their algorithm requires the use of a sophisticated data structure, the level-linked search tree. We propose a variant of the algorithm of Hoffman et al. that retains the linear time bound but simplifies both the operations required on the key data structure and the data structure itself. (16 pages, July 1988)

TR-190-88 Throughput of Long Self-Timed Pipelines
Mark R. Greenstreet and Kenneth Steiglitz

We explore the practical limits on throughput imposed by timing in a long, self-timed, circulating pipeline (ring). We first consider the case when computation, communication, and storage are combined in a single operation, and the time for this operation is random with an exponential distribution. This pipeline is amenable to queuing theory analysis, and we show that the asymptotic processor utilization is independent of the length of the pipeline, but is at most 25%. This suggests a design where computation and communication are separated from storage. We analyze this pipeline with various distributions of processing time, and show that linear speedup can again be achieved, but in this case with utilization approaching 100%. (17 pages, November 1988)

TR-191-88 Fast Allocation and Deallocation of Memory Based on Object Lifetimes
David R. Hanson

Dynamic storage management algorithms are based either on object sizes or object lifetimes. Stack allocation, which is based on object lifetimes, is the most efficient algorithm but restricts object lifetimes and creation times severely. Other algorithms based on object sizes, such as first fit and related algorithms, permit arbitrary lifetimes but cost more. More efficient variations capitalize on application-specific characteristics. Quick fit, for example, is more efficient when there are only a few object sizes. This paper describes a simple algorithm that is very efficient when there are few object lifetimes. In terms of instructions executed per byte allocated, the algorithm is almost half the cost of quick fit and les than twice the cost of stack allocation. Space for all objects with the same lifetime is allocated from a list of large arenas, and the entire list is deallocated at once. An implementation in ANSI C is included. (6 pages, November 1988)

TR-192-88 Applied Computational Geometry: Towards Robust Solutions of Basic Problems
David Dobkin and Deborah Silver

Geometric computations, like all numerical procedures, are extremely prone to roundoff error. However, virtually none of the numerical analysis literature directly applies to geometric calculations. Even for line intersection, the most basic geometric operation, there is no robust and efficient algorithm. Compounding the difficulties, many geometric algorithms perform iterations of calculations reusing previously computed data. In this paper, we explore some of the main issues in geometric computations and the methods that have been proposed to handle roundoff errors. In particular, we focus on one method and apply it to a general iterative intersection problem. Our initial results seem promising and will hopefully lead to robust solutions for more complex problems of applied computational geometry. (14 pages, December 1988)

TR-193-88 Efficiency of the Network Simplex Algorithm for the Maximum Flow Problem
Andrew V. Goldberg, Michael D. Grigoriadis, and Robert E. Tarjan

Goldfarb and Hao have proposed a network simplex algorithm that will solve a maximum flow problem on an n-vertex, m-arc network in at most nm pivots and $O(n^2m)$ time. In this paper we describe how to implement their algorithm to run in $O(nm \log n)$ time by using an extension of the dynamic tree data structure of Sleator and Tarjan. This bound is less than a logarithmic factor larger than that of any other known algorithm for the problem. (18 pages, October 1988)

TR-194-88 Probabilistic Analysis of a Closest Pair Algorithm
Mordecai Golin

In this paper we give a probabilistic analysis of the recent algorithm due to Hinrichs, Nievergelt, and Schorn for solving the closest pair problem. The analysis that we present is made under the algebraic decision tree model of computation (not requiring the use of floor functions) which distinguishes it from the grid-method techniques of Bentley, Weide, and Yao. The result that we derive is that – given n points uniformly distributed in the unit square and then sorted – a simple linear sweep through the points determines the closest pair in $O(n\sqrt{\log n})$ expected time. We close the paper by discussing the possibility that it might be possible to dispense with the $\sqrt{\log n}$ factor completely; that is, the sweep algorithm might actually have a linear expected running time. (10 pages, December 1988)

TR-195-88 System M: A Transaction Processing System for Memory Resident Data
Hector Garcia-Molina and Kenneth Salem

System M is an experimental transaction processing system that runs on top of the Mach operating system. Its database is stored in primary memory. This paper describes the structure and algorithms used in System M. The checkpointer is the component that periodically sweeps memory and propagates updates to a backup database copy on disk. Several different checkpointing (and logging) algorithms

were implemented, and their performance was experimentally evaluated. (23 pages, December 1988)

TR-196-88 Maintenance of Geometric Extrema
David Dobkin and Subhash Suri
A general technique is presented for maintaining the extrema of certain functions defined over sets of geometric objects. Maintenance is done as objects are inserted into and deleted from the data set. The technique is illustrated by presenting efficient algorithms for dynamically solving a variety of problems in computational geometry, robotics, VLSI design and operations research. (24 pages, December 1988)

TR-197-88 Profiling in the Presence of Optimization and Garbage Collection
Andrew W. Appel, Bruce F. Duba and David B. MacQueen
Profiling the execution of programs can be a great help in tuning their performance, and programs written in functional languages are no exception. The standard techniques of call-counting and statistical (interrupt-driven) execution time measurement work well, but with some modification. In particular, the program counter is not the best indicator of "current function." Our profiler inserts explicit increment and assignment statements into the intermediate representation, and is therefore very simple to implement and completely independent of the code-generator. (8 pages, November 1988)

TR-198-88 Management of a Remote Backup Copy for Disaster Recovery
Richard P. King, Nagui Halim, Hector Garcia-Molina and Christos A. Polyzios
A remote backup database system tracks the state of a primary system, taking over transaction processing when disaster hits the primary site. The primary and backup sites are physically isolated so that failures at one site are unlikely to propagate to the other. For correctness, the execution schedule at the backup must be equivalent to that at the primary. When the primary and backup sites contain a single processor, it is easy to achieve this property. However, this is harder to do when each site contains multiple processors and sites are connected via multiple communication lines. We present an efficient transaction processing mechanism for multiprocessor systems that guarantees this and other important properties. We also present a database initialization algorithm that copies the database to a backup site while transactions are being processed. (22 pages, December 1988)

TR-199-88 An Experimental Comparison of Initial Placement vs. Process Migration for Load Balancing Strategies
Kriton Kyrimis and Rafael Alonso
In this paper we describe an experimental comparison between initial placement and process migration to determine which is the better method to use to implement a load balancing system. A trace-based synthetic workload is described, which allows us to control the role that implementation overhead plays in such a system, and the two methods are compared for varying values of this overhead, from significant to negligible. (9 pages, December 1988)

TR-200-88 Distributing Workload Among Independently Owned Processors
Luis L. Cova and Rafael Alonso
In many distributed systems it is possible to share the processing capabilities among the nodes. To accomplish this goal, a number of load distribution algorithms have been proposed in the literature. In a network of independently owned processors (e.g. a network of workstations), load distribution schemes cannot consider the whole network as one unit and thus cannot try to optimize the overall performance. That is to say that load balancing schemes are not appropriate for this type of environment. Instead, the needs of each resource owner have to be considered. Therefore load sharing is more appropriate. Load sharing has been accomplished in some systems in an all or nothing fashion, i.e., if a node is idle then it becomes a candidate for executing a remote workload, otherwise it is not. Other attempts have used priority schemes where remote jobs are run with lower scheduling priority than local jobs. These styles of sharing are too restrictive in an environment where most resources are underutilized. Also, they do not scale well as the system load increases. We present a scheme that replaces these sharing approaches with a gradual one, i.e., where each machine in the network determines the amount of sharing it is willing to do. The scheme, called High-Low, makes sure that the service provided to local jobs of a lightly loaded node does not deteriorate by more than a predefined amount. It simultaneously helps improve the service at heavily loaded nodes. We empirically compare the different approaches to load distribution and show that it can be effective in a network of workstations. (36 pages, December 1988)

TR-201-89 Placement Problems Arising From Automatic Logic Compilation

William Wei-Lian Lin (Thesis)
Automatic logic compilation is the process of taking an input description consisting of Boolean logic equations and outputting an IC layout mask description implementing the equations. There are a number of problems that must be solved by the compiler. In this thesis, we shall discuss some placement problems that arose during the building of a logic compiler, the Weinberger Array Generator. (143 pages, June 1989)

TR-202-89 Querying a Network of Autonomous Databases
Patricia Simpson and Rafael Alonso
We consider the problem of enabling read-only access to a very large number of independent databases over a public network without compromising the autonomy or heterogeneity of the participants, both databases and queriers, who may number in the thousands or millions. From an examination of the network environment and its characteristics we derive general principles for designing applications involving autonomous systems, including statelessness, decentralization of control, and simple communication standards. Applying these principles, an architecture is developed for information exchange among diverse database systems. This architecture preserves autonomy by separating the functions of query interpretation, database discovery, and user interface into modules independently executable by any participant or subgroup of participants at any time. Subproblems arising in these three areas are discussed in detail, and solutions are presented. (18 pages, January 1989)

TR-203-89 Clocked Adversaries for Hashing
Richard J. Lipton and Jeffrey F. Naughton
A "clocked adversary" is a program that can time its operations and base its behavior on the results of those timings. While it is well known that hashing performs poorly in the worst case, recent results have proven that for the reference string programs, the probability of falling into a bad case can be driven arbitrarily low. We show that this is not true for clocked adversaries. This emphasizes the limits on the applicability of theorems on the behavior of hashing schemes on reference string programs, and raises a novel set of problems dealing with optimality of and vulnerability to clocked adversaries. (17 pages, February 1989)

TR-204-89 Communication Complexity: A Survey
László Lovász
The basic question in communication complexity theory is to find the minimum number of bits two processors have to communicate in order to find the value of a 2-variable function, if one knows the value of the first variable and the other knows the value of the second. This paper surveys some basic and recent results in this theory. The connections with VLSI are sketched. There are general lower and upper bounds involving the rank of the associated matrix and non-determinism. A recent upper bound by M. Saks and the author, extending an important result of Aho, Ullman and Yannakakis is presented. Using this, it is shown that for a large class of communication complexity problems, the square of the linear algebra lower bound on the communication complexity is an upper bound. The so-called Moebius function of lattices plays an important role. Further areas surveyed are randomized protocols and the applications of communication complexity in the other areas of complexity theory. (35 pages, March 1989)

TR-205-89 An Optimal Algorithm for Intersecting Three-Dimensional Convex Polyhedra
Bernard Chazelle
We describe a linear-time algorithm for computing the intersection of two convex polyhedra in 3-space. Applications of this result to computing intersections, convex hulls, and Voronoi diagrams are given. (29 pages, February 1989)

TR-206-89 Indexing in a Hypertext Database
Chris Clifton and Hector Garcia-Molina
Database indexing is a well studied problem. However, the advent of Hypertext databases opens new questions in indexing. Searches are often demarcated by pointers between text items. Thus the scope of the search may change dynamically, whereas traditional indexes cover a statically defined region such as a relation. We present a technique for indexing in hypertext databases. (15 pages, February 1989)

TR-207-89 Scheduling Real-Time Transactions with Disk Resident Data
Robert Abbott and Hector Garcia-Molina

Managing transactions with real-time requirements and disk resident data presents many new problems. In this paper we address several: How can we schedule transactions with deadlines? How do the real-time constraints affect concurrency control? How does the scheduling of IO requests affect the timeliness of transactions? How should exclusive and shared locking be handled? We describe a new group of algorithms for scheduling real-time transactions which produce serializable schedules. We present a model for scheduling transactions with deadlines on a single processor disk resident database system, and evaluate the scheduling algorithms through detailed simulation. (31 pages, February 1989)

TR-208-89 Odd Cycles, Bipartite Subgraphs, and Approximate Graph Coloring
Susan S. Wang Yeh (Thesis)
The graph coloring problem is to color the vertices of a graph so that no two adjacent vertices have the same color. This problem is not only NP-complete but also seems hard to approximate. In this thesis, we investigate the interplay of three different topics: odd cycles, bipartite subgraphs, and approximate graph coloring. Our primary goal is to design efficient approximate graph coloring algorithms with good performance. Our focus is on odd cycles and our central approach is to find bipartite subgraphs of graphs. We examine the role played by odd cycles of graphs in connection with graph coloring. We show that the presence or absence of certain size odd cycles gives graphs more structure and hence simplifies graph coloring. More specifically, we show that the absence of small odd cycles enables us to find large bipartite subgraphs. On the other hand, the absence of large odd cycles in a biconnected graph implies the absence of large even cycles, and these conditions imply that the graph contains special structures. We present efficient polynomial-time graph coloring algorithms which improve the performance guarantee on certain graphs, sometimes by a large margin, over the existing approximate graph coloring algorithms. For example, we can color triangle-free graphs with $O(\sqrt{n})$ colors, and in most cases, optimally color graphs with only 3-cycles and 5-cycles in odd cycles. We also present efficient polynomial-time algorithms for finding a maximum bipartite subgraph for special classes of graphs which include proper circular-arc, circular-arc permutation, and split graphs. On numerous occasions, we use one technique, namely dynamic programming, for the development of the algorithms. Furthermore, we show that we can modify the algorithms in a slight way to find a maximum bipartite subgraph that includes a nontrivial subset of vertices. (130 pages, June 1989)

TR-209-89 Estimating the Size of Path Relations
Richard J. Lipton and Jeffrey F. Naughton
We present a framework for estimating the size of path relations. We demonstrate the framework can be instantiated to provide a linear-time estimating algorithm for the size of the transitive closure of a sparse relation. We show that estimating the size of regular path relations over sparse graphs is at least as hard as estimating the size of transitive closures over arbitrary graphs. Our algorithm can be used to provide size estimates for generalizations of the transitive closure, including some recursive Datalog queries. Such estimating algorithms are essential if database systems that support recursive relations or fixpoints are to be able to optimize queries and avoid infeasible computations. (9 pages, January 1989)

TR-210-89 Shared Virtual Memory Accommodating Hetergeneity
Kai Li, Michael Stumm, David Wortman and Songnian Zhou
Heterogeneity exists in almost every research computing environment. Most operating systems accommodate heterogeneity by using a coherent file system that allows clients to access source files in a transparent way. In order to allow heterogeneous computer systems to cooperate, we have applied the framework of shared virtual memory to a heterogeneous computing environment and explore in detail how to accommodate heterogeneity. We have also designed a shared virtual memory system kernel, Mermaid, upon which one can build system components, programming languages, and application software using heterogeneous computers. We are currently implementing Mermaid on a network of SUN workstations and DEC Firefly multiprocessors. (22 pages, February 1989)

TR-211-89 Data Structures for Formal Verification of Circuit Designs
Steven Friedman (Thesis)
We address the problem of logic verification, for both combinational and sequential logic. We discuss the Binary Decision Diagram (or BDD) as a means to represent boolean functions. This representation is canonical with respect to the variable ordering and is usually compact. We present an $O(n^2 3^n)$ algorithm for finding the optimal variable ordering for a BDD, and compare it to a brute-force method and to heuristic approaches. We introduce a data structure called the projective BDD which is not

canonical but can be more compact than the BDD. We address the problem of verifying multiplier circuits using this structure. Finally, we present a form of deterministic finite automation called the nDFA, and apply it to the problem of sequential logic verification. (55 pages, June 1989)

TR-212-89 Fingerprinting Sets
Richard J. Lipton
We show how to efficiently compute hash functions that are invariant under permutations. These hash functions have a variety of applications to distributed computing problems. (5 pages, March 1989)

TR-213-89 Recursively Enumerable Languages Have Finite State Interactive Proofs
Richard J. Lipton
We show that interactive proof systems with 2-way probabilistic finite state verifiers can accept any recursively enumerable language. The same proof method also shows that the emptiness problem for 1-way probabilistic finite state machines is undecidable. (7 pages, March 1989)

TR-214-89 FACE: Enhancing Distributed File Systems for Autonomous Computing Environments
Rafael Alonso, Daniel Barbará and Luis L. Cova
As distributed systems become larger, the appropriate system architectures are those that provide a great deal of support for local node autonomy. Since in such autonomous computing environments we expect that servers will have the freedom to deny service to any user, we believe that the proper view of a server is not that it is the only place in which to obtain a service or resource, but rather a server is the best place to do so. In the case of CPU servers it is clear how the above may be accomplished: a job that is denied service at a server will simply be run locally, although perhaps in a degraded fashion. It is less clear what to do in the case of a file server. A possible approach consists of storing at the server the latest copy of all user files, while keeping local copies of older versions of the most crucial information. Thus, even after a failure users may be able to proceed with their work, although in a degraded manner. The idea of keeping local copies of key information has been called "stashing" in the literature. We augment the usefulness of stashing by combining it with the idea of "quasi-copies" (remote data copies that are allowed to diverge from the primary data but in a controlled, application-dependent fashion), which eases the cost of maintaining replicated data. In this paper we present the design of FACE, a distributed file system which allows users the option of using stashed files that are kept sufficiently consistent to fulfill user needs. The first FACE prototype has been designed and is currently being built via a series of enhancements to Sun's NFS. (19 pages, March 1989)

TR-215-89 A Probabilistic Relational Data Model
Daniel Barbará, Hector Garcia-Molina and Daryl Porter
It is often desirable to represent in a database entities whose properties cannot be deterministically classified. We develop a new data model that includes probabilities associated with the values of the attributes. The notion of missing probabilities is introduced for partially specified probability distributions. This new model offers a richer descriptive language allowing the database to more accurately reflect the uncertain real world. Probabilistic analogs to the basic relational operators are defined and their correctness is studied. (22 pages, January 1989)

TR-216-89 Network Flow Algorithms
Andrew V. Goldberg, Éva Tardos and Robert E. Tarjan
Network flow problems are central problems in operations research, computer science, and engineering and they arise in many real world applications. Starting with early work in linear programming and spurred by the classic book of Ford and Fulkerson, the study of such problems has led to continuing improvements in the efficiency of network flow algorithms. In spite of the long history of this study, many substantial results have been obtained within the last several years. In this survey we examine some of these recent developments and the ideas behind them. (79 pages, March 1989)

TR-217-89 Shiva: An Operating System Transforming a Hypercube into a Shared-Memory Machine
Kai Li and Richard Schaefer
The Shiva project at Princeton aims to develop an operating system supporting both the shared-memory and message-passing models of parallel computation for the second-generation message-passing multicomputers. Our initial system was designed and implemented for the Intel iPSC/2 hypercube multicomputer. It provides a large, coherent, shared virtual memory and a multihread interface – transforming a hypercube multicomputer into a shared-memory multiprocessor. The Shiva system also supports

message-passing among threads at low cost. Our preliminary performance measurements indicate that shared virtual memory is an effective strategy for implementing the next generation operating systems for hypercube multiprocessors. (16 pages, April 1989)

TR-218-89 Decreasing Channel Width Bounds by Channel Widening
Fook-Luen Heng, William W. Lin, Andrea S. LaPaugh and Ron Y. Pinter
Under the 2-layer Manhattan model for channel routing, the problem of minimizing the channel width, i.e., finding the smallest possible number of tracks required, is NP-complete. So, when a quick estimate of the width is needed (during placement, for example), one often uses a lower bound metric. Density is a commonly used metric, but another interesting metric is flux. When allowed to move some of the terminals in order to minimize the width, one usually tries to minimize a lower bound metric, hopefully thereby lowering the width. We consider the problem of decreasing a given metric to target value by adding empty columns. The empty columns add spaces in the corresponding positions on the top and bottom rows, so there are no changes in either the relative terminal orderings or the vertical alignments. This addition of columns does not affect the channel's density, but it can decrease the flux. Unfortunately, flux is not monotonically non-increasing as columns are added, so we derive from flux a new measure, smooth-flux. We present a polynomial-time algorithm for adding as few columns as possible in order to achieve a given target value for smooth-flux. Our algorithm also solves the Weighted Clique Cover problem on interval graphs. (21 pages, May 1989)

TR-219-89 Telematics Research at Princeton — 1988
Robert Abbott, Rafael Alonso, Daniel Barbará, Brad Barber, Matt Blaze, Chris Clifton, Luis Cova, Hector Garcia-Molina, Boris Kogan, Kriton Kyrimis, Christos Polyzois, Kenneth Salem, Patricia Simpson and Annemarie Spauster
In this note we briefly summarize the database and distributed computing research we performed in the year 1987. In general terms, our emphasis was on studying and implementing mechanisms for $efficient-and-reliable$ computing and data management. Our work can be roughly divided into 12 categories: load balancing, process migration, information exchange networks, negotiation in federated systems, a stashing file system implementation, probabilistic databases, distributed file comparison, remote backups, document databases, real time database systems, multicast protocols, and a new file system design. (17 pages, April 1989)

TR-220-89 A Runtime System
Andrew W. Appel
The runtime data structures of the Standard ML of New Jersey compiler are simple yet general. As a result, code generators are easy to implement, programs execute quickly, garbage colllectors are easy to implement and work efficiently, and a variety of runtime facilities can be provided with ease. (35 pages, May 1989)

TR-221-89 Random Walk Techniques for Protocol Validation
Daniel Barbará and José L. Palacios
In this paper we present a series of techniques based on random walks to perform state exploration in a reachability graph representing a protocol. Using a set of examples, we show experimental results that demonstrate the usefulness of the techniques. We also present the theoretical framework to prove why some of the techniques work better than others. (29 pages, June 1989)

TR-222-89 Faster Scaling Algorithms for General Graph Matching Problems
Harold N. Gabow and Robert E. Tarjan
This paper presents an algorithm for minimum cost matching on a general graph with integral edge costs, that runs in time close to the best known bound for cardinality matching. Specifically, let n, m and N denote the number of vertices, number of edges, and largest magnitude of a cost, respectively. The best known time bound for maximum cardinality matching is $O(\sqrt{nm})$. The new algorithm for minimum cost matching has time bound $O(\sqrt{n\alpha(m,n)\log nm}\log(nN))$. A slight modification of the new algorithm finds a maximum cardinality matching in the same time as above, $O(\sqrt{n}m)$. Other applications of the new algorithm are given, including an efficient implementation of Christofides' travelling salesman approximation algorithm and efficient solutions to update problems that require the linear programming duals for matching. (48 pages, April 1989)

TR-223-89 Almost-Optimum Parallel Speed-ups of Algorithms for Bipartite Matching and Related Problems

Harold N. Gabow and Robert E. Tarjan
This paper focuses on algorithms for matching problems that run on an EREW PRAM with p processors. Given is a bipartite graph with n vertices, m edges, and integral edge costs at most N in magnitude. An algorithm is presented for the assignment problem (minimum cost perfect bipartite matching) that runs $O(\sqrt{n}m\log(nN)(\log p)/p)$ time and $O(m)$ space, for $p \leq m/(\sqrt{n}\log 2n)$. This bound is within a factor of $\log^p$ of optimum speed-up of the best known sequential algorithm, which in turn is within a factor of $\log(nN)$ of the best known bound for the problem without costs (maximum cardinality matching). Extensions of the algorithm are given, including an algorithm for maximum cardinality bipartite matching with slightly better processor bounds, and similar results for bipartite degree-constrained subgraph problems (with and without costs). (30 pages, January 1989)

TR-224-89 Princeton SystemsFest Proceedings
These are the proceedings of the Princeton University Department of Computer Science SystemsFest a one day workshop on computer systems. Included are abstracts of the talks, as well as short writeups of the discussion following each talk. (39 pages, February 1989)

TR-225-89 A Spider User's Guide
Norman Ramsey
This manual explains how to use the Spider program to generate a WEB system for any programming language. It describes the syntax of the Spider description file used to describe a programming language, giving several examples. It does not say how to use the generated WEB system; for that see "The Spidery WEB System of Structured Documentation." (33 pages, August 1989)

TR-226-89 The Spidery WEB System of Structured Documentation
Norman Ramsey
This manual describes how to write programs in the WEB language, useing WEB systems generated by Spider. Most of the material is taken verbatim from Donald Knuth's original memo introducing WEB. It contains a brief introduction to the idea of literate programming, a short explanation of how to run WEAVE and TANGLE, and a list of all of the control sequences that can be used in WEB programs and their effects. (12 pages, August 1989)

TR-227-89 Triangulating A Nonconvex Polytope
Bernard Chazelle and Leonidas Palios
This paper is concerned with the problem of partitioning a three-dimensional polytope into a small number of elementary convex parts. The need for such decompositions arises in tool design, computer-aided manufacturing, finite-element methods, and robotics. Our main result is an algorithm for decomposing a polytope with n vertices and r reflex edges into $O(n + r^2)$ tetrahedra. This bound is asymptotically tight in the worst case. The algorithm is simple and practical. Its running time is $O((n + r^2)\log r)$. (17 pages, August 1989)

TR-228-89 Maintaining Bridge-Connected and Biconnected Components Online
Jeffery Westbrook and Robert E. Tarjan
We consider the twin problems of maintaining the bridge-connected components and the biconnected components of a dynamic undirected graph. The allowed changes to the graph are vertex and edge insertions. We give an algorithm for each problem. With simple data structures, each algorithm runs in $O(n\log n + m)$ time, where n is the number of vertices and m is the number of operations. We develop a modified version of the dynamic trees of Sleator and Tarjan that is suitable for efficient recursive algorithms, and use it to reduce the running time of the algorithms for both problems to $O(m\alpha(m,n))$, where α is a functional inverse of Ackermann's function. This time bound is optimal. All of the algorithms use $O(n)$ space. (40 pages, August 1989)

TR-229-89 Algorithms and Data Structures for Dynamic Graph Problems
Jeffrey Westbrook (Thesis)
We give data structures, algorithms, and lower bounds for three problems on dynamic graph, i.e., graphs that are being modified on-line. We consider the problem of maintaining the connected components of a graph undergoing edge insertions and backtracking edge deletions. We analyze several algorithms for the set union with backtracking problem, a variant of the classical disjoint set union (equivalence) problem in which an extra operation, de-union undoes the most recently performed union operation not yet undone. The most efficient such algorithms have an amortized running time of $O(\log n/\log\log n)$ per

operation, where n is the total number of elements in all the sets. We prove that any separable pointer-based algorithm for the problem requires $\Omega(\log n / \log\log n)$ time per operation, thus showing that our upper bound an amortized time is tight. We use these fast algorithms to build an algorithm with the same resource bounds that maintains the connected components of an undirected graph undergoing edge insertion and backtracking edge deletion. By a simple reduction, the lower bounds for set union with backtracking can be applied to the connected components problem. We examine the twin problems of maintaining the bridge-connected components or the biconnected components of a graph being changed by vertex and edge insertions. We give a basic algorithm that is suitable for both problems. With simple data structures, this algorithm runs in $O(n \log n + m)$ time, where n is the number of vertices and m is the number of operations. We develop a modified version of the dynamic trees of Sleator and Tarjan that is suitable for efficient recursive algorithms, and use it to reduce the running time of the algorithms for both problems to $O(m\alpha(m,n))$, where α is a functional inverse of Ackermann's function. This time bound is optimal. All of the algorithms use $O(n)$ space. We investigate the problem of maintaining a minimum spanning forest and the connected components of an embedded, edge-weighted planar graph undergoing changes in edge weight as well as edge and vertex insertion and deletion. To implement the algorithms, we develop a data structure called an edge-ordered dynamic tree, which is a variant of the dynamic tree data structure of Sleator and Tarjan. Using this data structure, our algorithms run in $O(\log n)$ amortized time per operation and $O(n)$ space. (86 pages, October 1989)

TR-230-89 Some Graphs with Small Second Eigenvalue
Joel Friedman
In this article we discuss several very simple graphs which have fairly small eigenvalues. Perhaps the most important result is that one of the graphs is a Cayley graph whose corresponding Cayley hypergraphs have a second eigenvalue which is essentially as small as a Cayley hypergraph can have. Also, these graphs are very simple to write down, and the eigenvalue calculation is based on the trace method and well known results about exponential sums. (14 pages, October 1989)

TR-231-89 Detecting the Intersection of Convex Objects in the Plane
David P. Dobkin and Diane L. Souvaine
Numerous applications require intersection detection. Many algorithms have been developed for telling whether two polygons intersect. We have extended one such algorithm to allow us to determine in $C \log n$ operations whether two convex planar regions intersects. Our algorithm is significant because it can be presented as a combination of two ideas. First, there is a revision of previous algorithms for detecting whether two convex polygons intersect. Second, there is a general method for transforming algorithms which work for polygons to make them work for piecewise curved boundaries. The constant C depends strictly upon the complexity of the piecewise curves. The algorithms presented here have been implemented and details of their implementation are included. (28 pages, October 1989)

TR-232-89 On the Second Eigenvalue of Hypergraphs
Joel Friedman and Avi Wigderson
We define a notion of second eigenvalue for regular hypergraphs. It turns out that a random hypergraph has a very small second eigenvalue. A class of applications of graphs with small second eigenvalue would be greatly improved if we could explicity construct hypergraphs with as small a second eigenvalue as random ones have. Unfortunately we have no such constructions as yet. In this paper we define the second eigenvalue, discuss the second eigenvalue of random hypergraphs, discuss some constructions and applications. (25 pages, November 1989)

TR-233-89 EZ Processes
David R. Hanson and Makoto Kobayashi
EZ is a system that integrates the facilities provided separately by traditional programming languages and operating systems. This integration is accomplished by casting services provided by traditional operating services as *EZ* language features. *EZ* is a high-level string processing language with a persistent memory. Traditional file and directory services are provided by *EZ*'s strings and associative tables, and tables are also used for procedure activations. This paper describes processes in *EZ*, which are procedure activations that execute concurrently and share the same, persistent virtual address space. They are semantically similar to 'threads' or 'lightweight processes' in some operating systems. Processes are values. They are just associative tables and have all of the characteristics of other *EZ* values, including persistence.

Examples of their use and a brief overview of their implementation are included. (10 pages, November 1989)

TR-234-89 Augmenting Availability in Distributed File Systems
Rafael Alonso, Daniel Barbara and Luis L. Cova
A very important feature in distributed systems is the availability of the services offered by the system. Traditionally, distributed systems provide the services by centralizing their administration at remote servers. This leaves the door open for increased dependency on these servers and the network connecting them to the local facilities. This may be a problem in environments where availability is a concern, e.g., very large distributed systems. In this paper, we describe the design of a distributed file system called FACE, that uses the notions of stashing (keeping local copies of key information) and quasi-copies (data copies that are allowed to diverge from the primary data but in a controlled, application-dependent fashion), to augment the availability of crucial files even when the file server that contains them is no longer reachable. We also report on the first FACE prototype that has been designed and implemented via a series of enhancements to Sun's Network File System. (27 pages, October 1989)

TR-235-89 Visibility with a Moving Point of View
Marshall Bern, David Dobkin, David Eppstein and Robert Grossman
We investigate 3-d visibility problems in which the viewing position moves along a straight flightpath. Specifically we focus on two problems: determining the points along the flightpath at which the topology of the viewed scene changes, and answering rayshooting queries for rays with origin on the flightpath. Three progressively more specialized problems are considered: general scenes, terrains, and terrains with vertical flightpaths. (11 pages, November 1989)

TR-236-89 Data Structures for Formal Verification of Circuit Designs
Steven Friedman (Thesis)
We address the problem of logic verification, for both combinational and sequential logic. Our focus is on increasing the size and range of circuits for which formal verification, as opposed to test vector generation and simulation, is feasible. We discuss the Binary Decision Diagram (or BDD) as a means of representing boolean functions. This representation is canonical with respect to the variable ordering and is usually compact. We present an original algorithm for finding the optimal variable ordering for a BDD. This algorithm is additionally useful in the synthesis of a certain type of circuit. We introduce a data structure called the projective BDD which is not canonical but can be more compact than the BDD. We develop an algorithm for verifying multiplier circuits using this structure. Finally, we present a form of deterministic finite automaton, the n-DFA, whose transitions are labeled with boolean functions. Using this structure, we develop an algorithm for the problem of sequential logic verification. For each of these algorithms—to do BDD minimization, multiplier verification, and verification of sequential circuits—we report the running times of actual implementations and compare them to benchmarks in order to demonstrate the practicality of our methods. (60 pages, January 1990)

TR-237-89 On Bounded Round Multi-Prover Interactive Proof Systems
Jin-yi Cai, Anne Condon and Richard J. Lipton
We compare bounded round multi-prover interactive proof systems (MIP's) with unbounded round interactive proof systems (IPS's). We show that for any constant ϵ, any language accepted by an unbounded round IPS has a bounded round, 2-prover MIP that has error probability ϵ, resolving an open problem of Fortnow, Rompel and Sipser. To obtain this result, we show that a certain 1-round MIP that simulates the computation of an unbounded round IPS can be executed many times in parallel to significantly reduce its probability of error. (9 pages, December 1989)

TR-238-89 Playing Games of Incomplete Information
Jin-yi Cai, Anne Condon and Richard J. Lipton
We study two-person games of cooperation and multi-prover interactive proof systems. We first consider a two person game G, which we call a free game, defined as follows. A Boolean function ϕ_G is given. Player I and II each pick a random number i and j in private, where $1 \leq i, j \leq s$, and then each chooses a private number $f(i)$ and $g(j), 1 \leq f(i), g(j) \leq s$. If $\phi_G(i, j, f, (i), g(j)) = 1$, then both players win, otherwise they lose. The objective of both players is to win collectively. We ask whether, if such a game is played n times in parallel, the probability of winning all the games decays exponentially in n. Before this work, it was unknown if $w(G^n) \to 0$ as $n \to \infty$. We show

Theorem If G is a non-trivial free game, then there exists a $q < 1 - e^{-3s}/2$ and a universal constant c_0, such that the winning probability of G^n is at most $c_0 q^n$.

Theorem If G is a non-trivial game, then there exists a $q < 1 - e^{-3s}/2$ and a universal constant c_0, such that the winning probability of G^n is at most $c_0 q^n$.

Theorem Let G be a non-trivial, $(1,2)$-limited free game. Let $w(G) = 1 - \epsilon$. Then if $n = \lceil 1/\epsilon \rceil$, $w(G^n) \leq 11/12$.

(10 pages, December 1989)

TR-239-89 Real-Time, Concurrent Checkpoint for Parallel Programs
Kai Li, Jeffrey F. Naughton and James S. Plank
We have developed and implemented a checkpointing and restart algorithm for parallel programs running on commercial uniprocessors and shared-memory multiprocessors. The algorithm runs concurrently with the target program, interrupts the target program for small, fixed amounts of time and is transparent to the checkpointed program and its compiler. The algorithm achieves its efficiency through a novel use of address translation hardware that allows the most time-consuming operations of the checkpoint to be overlapped with the running of the program being checkpointed. (15 pages, December 1989)

TR-240-89 How to Store a Triangular Matrix
Andrea S. LaPaugh, Richard J. Lipton and Jonathan S. Sandberg
We consider the problem of storing a triangular matrix so that each row and column is stored as a vector, i.e. the locations form an arithmetic progression. Storing rows and columns as vectors can speed up access significantly. We show that there is no such storage method that does not waste approximately one half of the computer memory. (7 pages, December 1989)

TR-241-90 A Dynamic File Caching Strategy that Preserves File Consistency
Kriton Kyrimis and Rafael Alonso
In this paper we present a dynamic file caching strategy that improves the average job response time while maintaining file consistency. By making the assumption that the read/write ratio of each file is fixed throughout the system, this strategy reduces to a simple criterion that can be applied using only local information for making caching decisions. In the second half of the paper, we present a set of simulation experiments that we conducted in order to evaluate the performance of this algorithm compared to the simple caching strategies of always caching and never caching files, in various network configurations. (26 pages, January 1990)

TR-242-90 Simple Hardware for Fast Interprocessor Communication
M. R. Greenstreet and K. Li
This paper presents a new approach for implementing point-to-point communication in multiprocessors: STARI, Self-Timed At Receiver's Input. STARI is a hybrid approach combining techniques from self-timed and synchronous designs. STARI designs are immune from clock-skew related problems and can operate faster than either purely synchronous or purely self-timed designs. Using a generic 2.0-μ CMOS process, transmission rates in excess of 250 Mbits/sec can be achieved for each wire connecting communicating processors. (11 pages, January 1990)

TR-243-90 Maintenance of a Minimum Spanning Forest in a Dynamic Planar Graph
D. Eppstein, G.F. Italiano, R. Tamassia, R.E. Tarjan, J. Westbrook and M. Yung
We give efficient algorithms for maintaining a minimum spanning forest of a planar graph subject to on-line modifications. The modifications supported include changes in the edge weights, and insertion. (29 pages, January 1990)

TR-244-90 Automatic Analysis of One-Parameter Planar Ordinary Differential Equations
By Intelligent Numeric Simulation
Elisha Sacks
This paper describes research on automating the analysis of physical systems modeled by ordinary differential equations. It describes a program called POINCARE that analyzes one-parameter autonomous planar systems at the level of experts through a combination of theoretical dynamics, numerical simulation, and geometric reasoning. The input is the system, a bounding box for the system state, a bounding interval into open subintervals of equivalent behavior bounded by bifurcation points, classifies the bifurcation points, and constructs representative phase diagrams for the subintervals. It detects

local and global generic one-parameter bifurcations. It constructs the phase diagrams by identifying fixed points, saddle manifolds, and limit cycles and partitioning the remaining trajectories into open regions of uniform asymptotic behavior. It obtains the region boundaries by numeric simulation, guided by theoretical knowledge. It determines the behavior near fixed points from the Jacobian of the system, scans outward to find basins of attractors and limit cycles, and stops at the bounding box. (32 pages, January 1990)

TR-245-90 An Advisor For Flexible Working Sets
Rafael Alonso and Andrew Appel
The traditional model of virtual memory working sets does not account for programs that can adjust their working sets on demand. Examples of such programs are garbage-collected systems and databases with block cache buffers. We present a memory-use model of such systems, and propose a method that may be used by virtual memory managers to advise programs on how to adjust their working sets. Our method tries to minimize memory contention and ensure better overall system response time. We have implemented a memory "advice server" that runs as a non-privileged process under Berkeley Unix. User processes may ask this server for advice about working set sizes, so as to take maximum advantage of memory resources. Our implementation is quite simple, and has negligible overhead, and experimental results show that it results in sizable performance improvements. (10 pages, February 1990)

TR-246-90 File System Design Using Large Memories
Carl Staelin and Hector Garcia-Molina
Over the last few years tremendous strides have been made in CPU performance, but without corresponding strides in I/O performance. Consequently, operating systems must be redesigned to minimize the impact of the I/O bottleneck. Memory is becoming cheaper and may be effectively used to improve I/O performance, but more sophisticated methods than simple demand caching may exist to make caching more effective. Analysis of file use in current file systems has shown the existence of file access patterns which may be exploited to provide better I/O performance. Using experimental data, we show what implications this has for file system design. We outline the design of the iPcress File System which uses both a large disk cache and other techniques to improve file system performance. Preliminary performance numbers for iPcress are also presented. (11 pages, June 1990)

TR-247-90 Coordinating Multi-Transaction Activities
H. Garcia-Molina, D. Gawlick, J. Klein, K. Kleissner, and K. Salem
Data processing applications must often execute collections of related transactions. We propose a model for structuring and coordinating these multitransaction activities. The model includes mechanisms for communication between transactions, for compensating transactions after an activity has failed, for dynamic creation and binding of activities, and for checkpointing the progress of an activity. (16 pages, February 1990)

TR-248-90 Two Epoch Algorithms for Disaster Recovery
Hector Garcia-Molina, Christos A. Polyzois and Robert Hagmann
Remote backup copies of databases are often maintained to ensure availability of data even in the presence of extensive failures, for which local replication mechanisms may be inadequate. We present two versions of an epoch algorithm for maintaining a consistent remote backup copy of a database. The algorithms ensure scalability, which makes them suitable for very large databases. The correctness and the performance of the algorithms are discussed, and an additional application for distributed group commit is given. (24 pages, February 1990)

TR-249-90 Efficient Polygon Triangulation
Bernard Chazelle
We give a deterministic algorithm for triangulating a simple polygon of n vertices in $O(n \log n)$ time. (30 pages, February 1990)

TR-250-90 Placement of Processes and Files in Distributed Systems
Kriton Kyrimis (Thesis)
In this thesis, we examine ways of improving the average job performance in a distributed system by controlling on which machine processes or files are placed. We describe an implementation of process migration under Sun UNIX, and use this implementation in experiments to assess the relative merits of process migration and initial placement as load-balancing strategies. Finally, we examine dynamic

file caching in distributed systems where maintaining file consistency is important. Using an analytical model, we derive a general caching algorithm. In a series of simulation studies, we compare the performance of this algorithm with that of the simpler strategies of never caching and always caching files for various network configurations. (64 pages, June 1990)

TR-251-90 Algorithmic Discrete Mathematics
Claire Kenyon and László Lovász
Lecture notes on course in Algorithmic Discrete Mathematics taught by Prof. László Lovász. The goal of this course is to present a number of general principles for the design of algorithms. (106 pages, March 1990)

TR-252-90 Algorithms for Bichromatic Line Segment Problems and Polyhedral Terrains
Bernard Chazelle, Herbert Edelsbrunner, Leonidas J. Guibas and Micha Sharir
We consider a variety of problems on the interaction between two sets of line segments in two and three dimensions. These problems range from counting the number of intersecting pairs between m blue segments and n red segments in the plane (assuming that two line segments are disjoint if they have the same color) to finding the smallest vertical distance between two non-intersecting polyhedral terrains in three-dimensional space. We solve these problems efficiently by using a varient of recent combinatorial and algorithmic techniques involving arrangements of lines in three-dimensional space, as developed in a companion paper. (14 pages, March 1990)

TR-253-90 Debugging Standard ML Without Reverse Engineering
Andrew P. Tolmach and Andrew W. Appel
We have built a novel and efficient replay debugger for our Standard ML compiler. Debugging facilities are provided by instrumenting the user's source code; this approach, made feasible by ML's safety property, is machine-independent and back-end independent. Replay is practical because ML is normally used functionally, and our compiler uses continuation-passing style; thus most of the program's state can be checkpointed quickly and compactly using call-with-current-continuation. Together, instrumentation and replay support a simple and elegant debugger featuring full variable display, polymorphic type resolution, stack trace-back, breakpointing, and reverse execution, even though our compiler is very highly optimizing and has no run-time stack. (12 pages, March 1990)

TR-254-90 Design of the PRAM Network
Jonathan Sandberg
The Princeton University PRAM computer network provides process-to-process data transmission rates equivalent to memory access rates with negligible latency penalties and currently costs less than 1,000.00 dollars per gigabit of network bandwidth. The PRAM network architecture described in this report scales to 10,000 node computer networks using fast optical technology. The cost of such a PRAM network implementation is expected, in the near term, to fall to $100.00 per gigabit of network bandwidth. Princeton University has been running a prototype hetrogeneous wide area PRAM network since early 1988. The system includes Apple Macintosh IIs, IBM PC/ATs and Clones, and Sun Series 3 workstations running Mac OS, DOS, OS/2, Xenix, Sun OS, and Mach. We have 6 PLAN boards (4×4 optical switches designed and fabricated at Princeton University) cascaded together to provide a single 14-way PRAM shared memory. The longest run of fiber in the network is one kilometer, however, the current prototype will support fiber runs up to three kilometers without repeaters. We have produced over 100 PRAM interface cards for the ISA, Nu, and VME buses as well as prototype PRAM-Capture networked data acquisition cards. All the prototype boards use inexpensive off-the-shelf parts for the bus interface, memory, and dual memory port arbitration. All fiber optic duplex serial ports are implemented with 125 MHz Advanced Micro Devices Taxi chips clocked at 6MHz in the 9-bit mode. The electrical to optical conversion is performed through 850 nm ODLs manufactured by Sumitomo Electric and Hewlett-Packard. (19 pages, April 1990)

TR-255-90 Determining the Separation of Preprocessed Polyhedra — A Unified Approach
David P. Dobkin and David G. Kirkpatrick
We show how (now familiar) hierarchical representations of (convex) polyhedra can be used to answer various separation queries efficiently (in a number of cases, optimally). Our emphasis is i) the uniform treatment of polyhedra separation problems, ii) the use of hierarchical representations of primitive objects to provide implicit representations of composite or transformed objects, and iii) applications to natural

problems in graphics and robotics. Among the specific results is an $O(\log |P| \cdot \log |Q|)$ algorithm for determining the separation of polyhedra P and Q (which have been individually preprocessed in at most linear time). (20 pages, April 1990)

TR-256-90 Princeton SystemsFest II

Proceedings of the Princeton SystemsFest II held at the Department of Computer Science on April 24 1990. (18 pages, April 1990)

TR-257-90 Slimming Down by Adding; Selecting Heavily Covered Points

B. Chazelle, H. Edelsbrunner, L.J. Guibas, J. E. Hershberger, R. Seidel and M. Sharir

We show that for any set P of n points in three-dimentional space there is a set Q of $O(n^{1/2}\log^3 n)$ points so that the Delaunay triangulation of $P \cup Q$ has at most $O(n^{3/2}\log^3 n)$ edges — even though the Delaunay triangulation of P may have $\Omega(n^2)$ edges. The main tool of our construction is the following geometric covering result: For any set P of n points in three-dimensional space and any set S of m spheres, where each sphere passes through a distinct point pair in P, there exists a point x, not necessarily in P, that is enclosed by $\Omega(\frac{m^2}{n^2 \log^6 \frac{n^2}{m}})$ of the spheres in S. Our results generalize to arbitrary fixed dimensions, to geometric bodies other than sphere, and to geometric structures other than Delaunay triangulations. (18 pages, April 1990)

TR-258-90 Competitive Paging as Cache Size Varies

Neal E. Young

Recent efforts to analyze paging strategies have examined the *competiveness* of various strategies — the worst-case ratio of the number of page faults made by a strategy on a sequence to the minimum number possible for any strategy with the same cache size. Two generalizations of competitive analysis have been considered: comparing strategies using different size caches and allowing randomized strategies. In this report, we study how well a randomized strategy can compare against a strategy with a smaller cache, and we study how competitiveness is related to the rate at which page fault rate decreases as the cache size increases. First, we show that the marking algorithm of Fiat et al. using a cache of size k has an expected number of faults within a factor of about $\ln \frac{h}{(k-h+1)}$ times the minimal number using a smaller cache of size h, and that this factor, called the (h,k)-competitiveness of the strategy, is within a factor of about two of optimal. Second, we show that for any fixed sequence, to the extent that the fault rate decreases only polynomially as the cache size increases, the optimal deterministic (k,k)-competitiveness is reduced to $O(\ln k)$ for most choices of k, and the optimal randomized competitiveness is reduced to $2\ln\ln k + O(1)$ for most k. (19 pages, April 1990)

TR-259-90 Some Complexity Questions Related to the Query $\prod_{1\le k\le m}(X_{ik} - X_{jk}) : 0$?

H. F. Ting

In this paper, we study the computational power of the $\prod_{1\le k\le m}(X_{ik} - X_{jk}) : 0$ decision tree model which allows queries of the type $\prod_{1\le k\le m}(X_{ik} - S_{jk}) : 0$. We prove that, in this model, $n + (k-1)\lceil \log n \rceil + 0(1)$ queries are necessary and sufficient to select the k largest of n distinct real numbers. We also prove that $2n + o(n)$ queries of the type $\prod_{1\le k\le m}(X_{ik} - X_{jk}) : 0?$ are sufficient to select the kth largest of n numbers. On another classical issue of complexity studies, namely the time-space tradeoff of a problem, we show that our model is closely related to the binary-comparison decision tree model. (12 pages, April 1990)

TR-260-90 The Polynomial Hierarchy is Provable by Two Provers in One Round

Jin-yi Cai

We consider multiprover interactive proof systems. We show that the polynomial hierarchy is provable by two provers in one round. (4 pages, December 1989)

TR-261-90 Generating Sparse Spanners for Weighted Graphs

Ingo Althofer, Gautam Das, David Dobkin and Deborah Joseph

Given a graph G, a subgraph G' is a t-spanner of G, if for every $u, v \in V$, the distance from u to v in G' is at most t times longer than the distance in G. In this paper we give a very simple algorithm for constructing sparse spanners for arbitrary weighted graphs. We then apply this algorithm to obtain

specific results for planar graphs and Euclidean graphs. We discuss the optimality of our results and present several nearly matching lower bounds. (12 pages, April 1990)

TR-262-90 Concurrent Programming in ML
Norman Ramsey
I have added concurrency to the functional language Standard ML. The model of concurrency is derived from CSP, but processes do not communicate directly; they interact by communicating over channels. Also, processes and channels can be created dynamically. Three features of Standard ML — polymorphism, garbage collection, and modules – work exceptionally well with concurrency. Polymorphic functions that combine the concurrent primitives can be used in many programs. One can write processes that do not explicitly provide for their own termination; the garbage collector reclaims them when they can no longer communicate. Finally, one can use the Standard ML modules system to make small adjustments in the meanings of the primitives. The implementation takes only 220 lines of Standard ML. It uses call with current continuation, `callcc`, to simulate concurrent execution on a sequential machine. `callcc` is implemented efficiently by the Standard ML of New Jersey compiler, which uses no runtime stack. (18 pages, April 1990)

TR-263-90 On Evaluating Boolean Functions with Unrealiable Tests
Claire Kenyon and Andrew C. Yao
We consider the problem of evaluating a boolean function $P(x_1, \ldots, x_n)$ by asking queries of the form "$x_i =?$," and receiving answers which may not always be truthful. Assuming that the total number of lies does not exceed E, we present an algorithm with cost $O(n + s_P E + t_P E)$, where s_P is the maximal size of a minterm of $P(x)$ and t_P is the maximal size of a maxterm. We also prove that if P is monotone, then any algorithm for evaluating P must ask $\Omega(s_P E + t_P E)$ queries for some input. (10 pages, April 1990)

TR-264-90 Triangulating a Simple Polygon in Linear Time
Bernard Chazelle
We give a deterministic algorithm for triangulating a simple polygon in linear time. The basic strategy is to build a coarse approximation of a triangulation in a bottom-up phase and then use the information computed along the way to refine the triangulation in a top-down phase. The main tools used are (i) the polygon-cutting theorem, which provides us with a balancing scheme, and (ii) the planar separator theorem, whose role is essential in the discovery of new diagonals. Only elementary data structures are required by the algorithm. In particular, no dynamic search trees, finger trees, or point location structures are needed. We mention a few applications of our algorithm. (35 pages, May 1990)

TR-265-90 Short Encodings of Evolving Structures
Daniel D. Sleator, Robert E. Tarjan and William P. Thurston
A derivation in a transformational system such as a graph grammar may be redundant in the sense that the exact order of the transformations may not affect the final outcome; all that matters is that each transformation, when applied, is applied to the correct substructure. By taking advantage of this redundancy, we are able to develop an efficient encoding scheme for such derivations. This encoding scheme has a number of diverse applications. It can be used in efficient enumeration of combinatorial objects or for compact representation of program and data structure transformations. It can also be used to derive lower bounds on lengths of derivations. We show for example that $\Omega(n \log n)$ applications of the associative and commutative laws are required in the worst case to transform an n-variable expression over a binary associative, commutative operation into some other equivalent expression. Similarly, we show that $\Omega(n \log n)$ "diagonal flips" are required in the worst case to transform one n-vertex numbered triangulated planar graph into some other one. Both of these lower bounds have matching upper bounds. An $O(n \log n)$ upper bound for associative, commutative operations was known previously, whereas we obtain here an $O(n \log n)$ upper bound for diagonal flips. (26 pages, April 1990)

TR-266-90 Probabilistic Analysis of Geometric Algorithms
Mordecai J. Golin (Thesis)
This thesis is divided into four chapters. In the first chapter we describe the subtleties involved in probabilistically analyzing simple algorithms in computational geometry. We also work through a few easy examples of such analyses. The first example analyzes Quicksort, the second analyzes Quickhull, and the third analyzes interpoint distances between uniformly distributed points in hypercubes or hypertori. In the second chapter we present a simple but effective preprocessing algorithm for calculating convex

hulls. The algorithm is short and intuitive. It does a fast linear scan through the points, identifying many which are not on the convex hull and can therefore be eliminated. We perform an exact analysis of this algorithm showing that, given n points distributed uniformly in the unit square, only about $8\sqrt{n}$ of them remain after the preprocessing step: in higher dimensions only $c_d n^{1-1/d}$ will remain. We present the results of simulations comparing our mathematical analysis to reality. Finally, we end with a discussion of what distinguishes this algorithm from certain obvious variants. In the third chapter we analyze closest pair algorithms. First, we analyze a sweep-line closest-pair algorithm, one similar in spirit to Hoey and Shamos' divide and conquer algorithm for solving the same problem. Our result is that, given n points uniformly distributed in the unit square and then sorted, there is a six line algorithm that finds the closest pair in $O(n)$ expected time. Moreover, this algorithm uses no complicated data structures. We then analyze a second algorithm, one that finds the closest pair using a modified version of Bentley and Papadimitriou's nearest neighbor projection algorithm. Our result, again, is that after the sorting stage, the linear scan stage of this new algorithm also finds the closest pair in $O(n)$ expected time. The fourth chapter discusses the calculation of maxima. More specifically it analyzes a new heuristic, the Move-To-Front Algorithm, recently developed by Bentley, Clarkson and Levine. We prove mathematically that the Move-To-Front Algorithm is extremely efficient; on average it performs only one comparison per input point except on those in an asymptotically negligible subset. (145 pages, June 1990)

TR-267-90 Unique Binary Search Tree Representations and Equality-testing of Sets and Sequences
Rajamani Sundar and Robert E. Tarjan
Given an ordered universe U, we study the problem of representing each subset of U by a unique binary search tree so that dictionary operations can be performed efficiently. We exhibit representations that permit the execution of dictionary operations in optimal time when the dictionary is sufficiently sparse or sufficiently dense. We apply unique representations to obtain efficient data structures for maintaining a collection of sets/sequences under queries that test the equality of a pair of objects. In the process, we devise an interesting method for maintaining a dynamic, sparse array. (14 pages, November 1989)

TR-268-90 More Efficient Bottom-Up Tree Pattern Matching
J. Cai, R. Paige, and R. Tarjan
Pattern matching in trees is fundamental to a variety of programming language systems. However, progress has been slow in satisfying a pressing need for general purpose pattern matching algorithms that are efficient in both time and space. We offer asymptotic improvements in both time and space to Chase's bottom-up algorithm for pattern preprocessing. Our preprocessing algorithm has the additional advantage of being incremental with respect to pattern additions and deletions. We show how to modify our algorithm using a new decomposition method to obtain a space/time tradeoff. Finally, we trade a log factor in time for a linear space bottom-up pattern matching algorithm that handles a wide subclass of Hoffmann and O'Donnell's Simple Patterns. (15 pages, May 1990)

TR-269-90 Testing Parallel Simulators for Two-Dimensional Lattice-Gas Automata
Richard Squier and Kenneth Steiglitz
We describe a test method for lattice-gas automata of the type introduced by Frisch, Hasslacher, and Pomeau. The test method consists of inserting test patterns into the initial state of the automaton and using a graphics display to detect errors. The test patterns are carefully constructed limit cycles that are disrupted by errors occurring at any level of the simulator system. The patterns can be run independently to test the system for debugging purposes, or they can be run as sub-simulations embedded in a larger lattice-gas simulation to detect faults at runtime. We describe the use of this method on a prototype parallel machine for lattice-gas simulations, and discuss the range of systems that can make use of this type of test method. The test patterns detect all significant one-bit errors. We include experimental results indicating that multiple bit errors are unlikely to escape detection. (46 pages, June 1990)

TR-270-90 A Code Generation Interface for ANSI C
Christopher W. Fraser and David R. Hanson
`lcc` is a retargetable, production compiler for ANSI C; versions for the VAX, the Motorola 68020 and MIPS have been in use for over a year and a half. It is smaller and faster than generally available alternatives, and its local code is about as good. This report describes the interface between the target-independent front end and the target-dependent back ends. The interface consists of shared data structures, a few functions, and a dag language. While this approach couples the front and back ends tightly,

it results in an efficient, compact compiler. The interface is illustrated by detailing a complete code generator that emits naive VAX code. (37 pages, June 1990)

TR-271-90 Comparison of Tree and Straight-Line Clocking for Long Systolic Arrays
Marios D. Dikaiakos and Kenneth Steiglitz
A critical problem in building long systolic arrays lies in efficient and reliable synchronization. We address this problem in the context of synchronous systems by introducing probabilistic models for two alternative clock distribution schemes: tree and straight-line clocking. We present analytic bounds for the Probability of Failure and the Mean Time to Failure, and examine the trade-offs between reliability and throughput in both schemes. Our basic conclusion is that as the one-dimensional systolic array gets very long, tree clocking becomes more advantageous over straight-line clocking. (23 pages, June 1990)

TR-272-90 Data Sharing in a Large Heterogeneous Environment
Rafael Alonso, Daniel Barbara and Steve Cohn
The increased availability of networking technology, as well as the economic pressure for inter-company cooperation have created a great deal of interest in federated (or heterogeneous) database systems. Although there has been much work recently in this area, most of it addresses (implicitly or explicitly) an environment in which there are only relatively few cooperating entities. In this paper we explore the issues involved in sharing information among a large collection of independent databases. First, we discuss some of the distinguishing features that characterize such large scale environments (such as size, autonomy, and heterogeneity). We then outline a multi-step information sharing process for those systems, and present an architecture supporting that exchange. We conclude by providing a detailed description of a working prototype based on our architecture, and present some measurements of its performance. (24 pages, July 1990)

TR-273-90 LEFTY: A Two-view Editor for Technical Pictures
Eleftherios E. Koutsofios (Thesis)
This thesis describes LEFTY, a two-view graphics editor for technical pictures. This editor has no hardwired knowledge about specific picture layouts or editing operations. Each picture is described by a program that contains functions to draw the picture and functions to perform editing operations that are appropriate for the specific picture. Primitive user actions, like mouse and keyboard events, are bound to functions in this program. Besides the graphical view of the picture itself, the editor presents a textual view of the program that describes the picture. Programmability and the two-view interface allow the editor to handle a variety of pictures, but are particularly useful for pictures used in technical contexts, e.g., graphs and trees and VLSI layouts. LEFTY can communicate with other processes. This feature allows it to use existing tools to compute specific picture layouts and allows external processes to use the editor to display their data structures. (86 pages, October 1990)

TR-274-90 CLOVER: A Timing Constraints Verification System
Dimitirs Doukas and Andrea S. LaPaugh
Correct timing is a critical issue in hardware design, especially in the case of bus interfaces. In this paper we present the design and implementation of a timing verification tool, CLOVER. CLOVER provides the designer of a digital circuit with an integrated environment where he can describe his design, formally specify the timing constraints governing the implementation, obtain the timing behavior of the implemented design and verify it against the stated timing constraints. Our timing constraints specification language, which is used to formally express timing constraints, is a general language but has been designed particularly for the description of asynchronous designs. The language is based on dependency graphs and vectors of signal values over the time. We give specifications for well known timing interface problems to illustrate the expressive power of our language. Finally, we specify and verify an implementation of the multibus design frame. (28 pages, July 1990)

TR-275-90 Graph Decompositions with Applications to Circle and Permutation Graphs
Csaba Peter Gabor (Thesis)
Just as integers can be decomposed into prime factors, it is possible to define primeness on graphs so that graphs may also be decomposed into their prime components. Under the definitions presented in this work, several non-isomorphic graphs may yield identical factorings. An efficient implementation of an algorithm which Cunningham introduced to decompose graphs is shown. A second algorithm to decompose graphs is also shown. This second algorithm is shown because it is also used in the

inductive proof of a property of prime graphs — each vertex in a prime graph must be contained in a certain induced subgraph of the prime graph. A close link between prime graphs and certain classes of intersection graphs, namely circle graphs and permutation graphs, is established, and a recognition algorithm is presented for each. (99 pages, October 1990)

TR-276-90 Virtual Memory Primitives for User Programs
Andrew W. Appel and Kai Li
Memory Management Units (MMUs) are traditionally used by operating systems to implement disk-paged virtual memory. Some operating systems allow user programs to specify the protection level (inaccessbile, read-only, read-write) of pages, and allow user programs to handle protection violations, but these machanisms are not always robust, efficient, or well-matched to the needs of applications. We survey several user-level algorithms that make use of page-protection techniques, and analyze their common characteristics, in an attempt to answer the question, "What virtual-memory primitives should the operating system provide to user processes, and how do today's operating systems provide them?" (13 pages, July 1990)

TR-277-90 Scalable Shared Memory Interconnections
Dimitiros Nikolaou Serpanos (Thesis)
This dissertation presents an architecture and describes an implementation for a high-performance, scalable shared memory interconnection. The architecture is based on a scalable shared memory model called PRAM. Conventional shared memory multiprocessors provide high performance but they do not scale well to either a large number of processors or over long distances. The PRAM network is scalable and allows heterogeneous processors to be interconnected achieving high effective data transfer rates and low latencies. An implemented prototype interconnects IBM AT, SUN-3 and MAC-II machines demonstrating performance improvements over conventional high-performance scalable multiprocessors. The successful prototype implementation proves that high-performance, low-cost, scalable shared memory interconnections can be built and combine high performance with scalability. (98 pages, October 1990)

TR-278-90 METEOR: A Constraint-based FIR Filter Design Program
K. Steiglitz, T. W. Parks and J. F. Kaiser
The usual way of designing a filter is to specify a filter length and a nominal response, and then to find a filter of that length which best approximates that response. In this paper we propose a different approach: specify the filter only in terms of upper and lower limits on the response, find the shortest filter length which allows these constraints to be met, and then find a filter of that length which is farthest from the upper and lower constraint boundaries in a mini-max sense. Previous papers have described methods for using an exchange algorithm for finding a feasible linear-phase FIR filter of a given length if one exists, given upper and lower bounds on its magnitude response. The resulting filter responses touch the constraint boundaries at many points, however, and are not good final designs because they do not make the best use of the degrees of freedom in the coefficients. We use the simplex algorithm for linear programming to find the best linear-phase FIR filter of minimum length, as well as to find the minimum feasible length itself. The simplex algorithm, while much slower than exchange algorithms, also allows us to incorporate more general kinds of constraints, such as constraints which force the magnitude response to be a concave function in a particular band. Very flat passband magnitude characteristics can be obtained by constraining the passband to be a concave-downward function. We give examples that illustrate how the proposed and the usual approaches differ, and how the new approach can be used to design filters with flat passbands, filters which meet point constraints, minimum-phase filters, and bandpass filters with controlled transition band behavior. (19 pages, August 1990)

TR-279-90 Ein Kleiner Filter Compiler
Kenneth Steiglitz
A "little" compiler (actually, a C pre-processor) is described that translates an Intermediate Filter Language (IFL) into filter code. The main motivation is to provide an easy way to experiment with filters for digital sould synthesis. The compiler itself comprises 60 lines of C and produces C. The new keyword `tap` is available, which creates a buffer for storing the current value of the signal, which is then available earlier or later in the computation as a feedback or feed-forward term, respectively. C code in the filter description is passed largely unchanged, except for variables of the form it Si which is the signal value from the ith buffer. In this way, the usual FIR and IIR filters can be represented easily, as well as much more general filters and signal generators. The compiler is portable, requires small computer

resources, uses text input that can be generated by other programs, and is easy to modify. (11 pages, August 1990)

TR-280-90 Reconfigurability and Reliability of Systolic/Wavefront Arrays
Edwin Hsing-Mean Sha and Kenneth Steiglitz
In this paper we study fault-tolerant redundant structures for maintaining reliable arrays. In particular we assume the desired array (application graph) is embedded in a certain class of regular, bounded-degree graphs called dynamic graphs. We define the degree of reconfigurability DR, and DR with distance DR^d, of a redundant graph. When DR (respectively DR^d) is independent of the size of the application graph, we say the graph is finitely reconfigurable, FR (resp. locally reconfigurable, LR). We show that DR provides a natural lower bound on the time complexity of any distributed reconfiguration algorithm, and that there is no difference between being FR and LR on dynamic graphs. We then show that if we wish to maintain both local reconfigurability, and a fixed level of reliability, a dynamic graph must be of dimension at least one greater than the application graph. Thus, for example, a one-dimensional systolic array cannot be embedded in a one-dimensional dynamic graph without sacrificing either reliability or locality of reconfiguration. (28 pages, August 1990)

TR-281-90 A Rapid Hierarchical Radiosity Algorithm for Unoccluded Environments
Pat Hanrahan and David Salzman
This paper presents a linear-time radiosity algorithm for scenes containing large mutually unoccluded polygonal patches. It subdivides pairs of patches adaptively to build a hierarchical data structure with n elements at the leaves, and it encodes all the light transport between component polygonal elements. Given a required numerical precision, determined here by the specified bounds for maximum solid angle F_ϵ and minimum area A_ϵ, our algorithm reduces the number of form factor calculations and interactions to $O(n)$ in the worst case and $O(\sqrt{n})$ in the best case. Standard techniques for shooting and gathering can then be used with the data structure. The best previous radiosity algorithms represented the element-to-element transport interactions with n^2 form factors. (24 pages, August 1990)

TR-282-90 Resource Management in Federated Computing Environments
Luis L. Cova (Thesis)
This dissertation demonstrates that resource management mechanisms designed to support several degrees of autonomy allow the establishment of effective cooperation among autonomous computing sites. Using two distinct resources, storage and processing, this dissertation introduces two distinct notions for autonomy: functional autonomy and service autonomy. For functional autonomy, it is shown that it is practical to implement a network file system that provides increased client independence from the file server while maintaining a useful level of functionality. For service autonomy, a load sharing mechanism has been implemented that provides sites' owners with ways to control how much performance degradation local jobs will perceive when remote jobs are serviced. This dissertation claims that, although some very large distributed systems will be formed by the growth of integrated distributed systems, most will be created by joining together a number of separate, already existing, and independent distributed systems. In an environment composed of a multitude of cooperating sub-systems the autonomy of each entity must be respected in order to convince the owners of each of the separate components to join the federation. Consequently, any viable distributed system architecture must support the notion of autonomy if it is to scale at all in the real world. (122 pages, October 1990)

TR-283-90 Clustering Active Disk Data To Improve Disk Performance
Carl Staelin and Hector Garcia-Molina
We show that clustering the active data of a file system in the center of the disk is a viable and effective means of improving system I/O performance. We demonstrate that it can reduce the average seek delay and in the presence of disk queues it can also reduce the average rotational delay. We also present experimental results which show that file access patterns are strongly skewed, and that file activity levels are relatively stable over time, making them a good predictor of future file activity levels. Using simulations, we investigate two techniques for reorganizing the disk data, and we measure sensitivity to imperfect predictions of future file activity due to drifting file activity levels. We demonstrate significant performance improvements over a full spectrum of file system use, from lightly loaded systems through heavily loaded systems with large disk queues, with performance benefits increasing as the system load increases. (17 pages, September 1990)

TR-284-90 Scheduling and Bin Packing: A Study of the Worst-case Performance Bounds
Weizhen Mao (Thesis)

In this dissertation, we study the worst-case performance bounds of various algorithms of scheduling problem and bin packing problem. For the scheduling problem with m parallel machines and precedence constraints, let ω be the total elapsed time of the execution of all tasks in the optimal non-preemptive schedule, and let ω' be the total alapsed time in the optimal preemptive schedule. C. L. Liu conjectured twenty years ago that the ratio $\frac{\omega}{\omega'}$ is no greater than $\frac{2m}{(m+1)}$ and this bound is also the best possible. We will prove the conjecture for a number of cases, such as when the precedence constraint is empty, or when the task system satisfies certain conditions; for other cases the previous result $\frac{\omega}{\omega'} \leq \frac{(2m-1)}{m}$ given by R. R. Muntz will be substantially improved. The bin packing, a special scheduling problem, is to pack a list of reals in (0,1] into unit-capacity bins using the minimum number of bins. Let $R[A]$ be the limiting worst-case value for the ratio $A(L)/L^*$ as $L^* \rightarrow \infty$, where $A(L)$ denotes the number of bins used when the approximation algorithm A is applied to the list L, and $L*$ denotes the minimum number of bins needed to pack L. For Next-k-Fit (NkF) and Best-k-Fit (BkF), which are linear-time approximation algorithms for bin packing, it was known that both $R[NkF]$ and $R[BkF]$ are in the interval $[1.7+\frac{3}{10k}, 2]$. In this dissertation, two precise bounds $R[NkF] = 1.7 + \frac{3}{10(k-1)}$ and $R[RkF] = 1.7 + \frac{3}{10k}$ are proved. (73 pages, October 1990)

TR-285-90 The Spectra of Infinite Hypertrees
Joel Friedman

We develop a model of regular, infinite hypertrees, to mimic for hypergraphs what infinite trees do for graphs. We then examine two notions of spectra or "first eigenvalue" for the infinite tree, obtaining a precise value for the first notion and obtaining some estimates for the second. The results indicate agreement of the first eigenvalue of the infinite hypertree with the "second eigenvalue" of a random hypergraph of the same degree, to within logarithmic factors, at least for the first notion of first eigenvalue. (20 pages, September 1990)

TR-286-90 Average-Case Analysis of Graph-Searching Algorithms
Sarantos Kapidakis (Thesis)

We estimate the expected value of various search quantities for a variety of graph-searching methods, for example depth-first search and breadth-first search. Our analysis applies to both directed and undirected random graphs, and it covers the range of interesting graph densities, including densities at which a random graph is disconnected with a giant connected component. We estimate the number of edges examined during the search, since this number is proportional to the running time of the algorithm. We find that for hardly connected graphs, all of the edges might be examined , but for denser graphs many fewer edges are generally required. We prove that any searching algorithm examines $\Theta(n \log n)$ edges, if present, on all random graphs with n nodes but not necessarily on the complete graphs. One property of some searching algorithms is the maximum depth of the search. In depth-first search, this depth can be used to estimate the space needed for the recursion stack. For random graphs of any density, even for disconnected graphs, we prove that this space is $\Theta(n)$. On the other hand, the depth of breadth-first search is $\Theta(\log n / \log(pn))$, where p is the probability of the existence of an edge. The size of the data structure needed by any searching algorithm is proved to be $\Theta(n)$. If the search terminates at a particular node, any searching algorithm needs a data structure of size $\Theta(n)$, and examines only $\Theta(n)$ edges. Finally, we derive similar results for variants of the above searching algorithms, more general classes of searching algorithms, and for random graphs with multiple edges. These results are verified through simulations. The techniques employed to permit simulations of big graphs (few millions of nodes) and the results obtained are of general interest, especially to those performing similar experiments. (147 pages, October 1990)

TR-287-90 Scheduling I/O Requests with Deadlines: A Performance Evaluation
Robert K. Abbott and Hector Garcia-Molina

A real-time computing system allocates resources to tasks with the goal of meeting individual task deadlines. The CPU, main memory, I/O devices and access to shared data all should be managed with the goal of meeting task deadlines. This paper examines the I/O scheduling problem in detail. We present a detailed, realistic model for studying this problem in the context of a system which executes real-time transactions. The model takes advantage of the fact that reading from the disk occurs before a transaction commits while writing to the disk usually occurs after the transaction commits. We develop

new algorithms that exploit this fact in order to meet the deadlines of individual requests. The algorithms are evaluated via detailed simulation and their performance is compared with traditional disk scheduling algorithms. (12 pages, October 1990)

TR-288-90 Counting and Cutting Cycles of Lines and Rods in Space
B. Chazelle, H. Edelsbrunner, L. Guibas, R. Pollack, R. Seidel, M. Sharir and J. Snoeyink
A number of rendering algorithms in computer graphics sort three-dimensional objects by depth and assume that there is no cycle that makes the sorting impossible. One way to resolve the problem caused by cycles is to cut the objects into smaller pieces. In this paper we address the problem of estimating how many such cuts are always sufficient. We also consider a few related algorithmic and combinatorial geometry problems. For example, we demonstrate that n lines in space can be sorted in randomized expected time $O(n^{4/3+\epsilon})$, provided they define no cycle. We also prove an $O(n^{7/4})$ upper bound on the number of points in space so that there are n lines with the property that for each point there are at least three non-coplanar lines that contain it. (17 pages, October 1990)

TR-289-90 Verification and Sensitivity Analysis of Minimum Spanning Trees in Linear Time
Brandon Dixon, Monika Rauch and Robert E. Tarjan
Komlós has devised a way to use a linear number of binary comparisons to test whether a given spanning tree of a graph with edge costs is a minimum spanning tree. The total computational work required by his method is much larger than linear, however. We describe a linear-time algorithm for verifying a minimum spanning tree. Our algorithm combines the result of Komlós with a preprocessing and table look-up method for small subproblems and with a previously known almost-linear-time algorithm. Additionally, we present an optimal deterministic algorithm and a linear-time randomized algorithm for sensitivity analysis of minimum spanning trees. (12 pages July 1990)

TR-290-90 Quasi-Optimal Upper Bounds for Simplex Range Searching and New Zone Theorems
Bernard Chazelle, Micha Sharir and Emo Welzl
This paper presents quasi-optimal upper bounds for simplex range searching. The problem is to preprocess a set P of n points in $\Re^d$ so that, given any query simplex q, the points in $P \cap q$ can be counted or reported efficiently. If m units of storage are available ($n < m < n^d$) then we show that it is possible to answer any query in $O(n^{1+\epsilon}/m^{1/d})$ query time after $O(m^{1+\epsilon})$ preprocessing. This bound, which holds on a RAM or a pointer machine, is almost tight. We also show how to achieve $O(\log^{d+1} n)$ query time at the expense of $O(n^{d+\epsilon})$ storage, for any fixed $\epsilon > 0$. To fine tune our results in the reporting case we also establish new zone theorems for arrangements and merged arrangements of planes in 3-space, which are of independent interest. (22 pages, October 1990)

TR-291-90 Real-Time Concurrent Collection in User Mode
Kai Li
We previously presented a real-time, concurrent garbage-collection algorithm that uses the virtual memory page protection hardware to synchronize collector threads and mutator threads. The algorithm requires the mutator threads to access protected pages that prevent collector threads from accessing. This paper investigates three other alternatives to achieve such a goal: page-copying, multiple address mapping, and page sharing in different address spaces. We will present our experiment with the page-copying version and compare it with the kernel-mode, simple stop-and-copy, and sequential real-time versions. (5 pages, October 1990)

TR-292-90 A Note on Poset Geometries
Joel Friedman
In this note we describe how varying the geometric representation of a poset can be applied to "poset balancing." We show that the 1/3, 2/3 balancing property holds for a certain class of posets whose number of relations is sufficiently small, in a certain sense. (9 pages, October 1990)

TR-293-90 The Best Case of Heapsort
Robert Sedgewick and Russel Schaffer
Heapsort is a fundamental algorithm whose precise performance characteristics are little understood. It is easy to show that the number of keys moved during the algorithm is $N \lg N + O(N)$ in the worst case, and it is conjectured that the average case performance is the same, though that seems very difficult to prove. In this paper, we show by construction that the *best case* for the number of moves is $\sim \frac{1}{2} N \lg N$. This destroys the conjecture that Heapsort is asymptotically flat (a possible easy road to the average-case

asymptotic analysis), and also implies that a variant of Heapsort suggested by Floyd is not asymptotically optimal in the worst case. The construction was suggested by results of an empirical study that involved generating and analyzing hundreds of billions of heaps. Other facts learned from this study about the distribution of the number of moves required by Heapsort are presented. (12 pages, November 1990)

TR-294-90 Lines in Space: Combinatorics and Algorithms
Bernard Chazelle, Herbert Edelsbrunner, Leonidas J. Guibas, Micha Sharir, and Jorge Stolfi
Questions about lines in space arise frequently as subproblems in 3-dimensional Computational Geometry. In this paper we study a number of fundamental combinatorial and algorithmic problems involving arrangements of n lines in 3-dimensional space. Our main results include:

1. A tight $\Theta(n^2)$ bound on the maximum combinatorial description complexity of the set of all oriented lines that have specified orientations relative to the n given lines.
2. A similar bound of $\Theta(n^3)$ for the complexity of the set of all lines passing above the n given lines.
3. A randomized preprocessing procedure using $O(n^{2+\epsilon})$ time and storage, for any $\epsilon > 0$, that builds a structure supporting $O(\log n)$-time queries for testing if a line lies above all the given lines.
4. An $O(n^{4/3+\epsilon})$ randomized expected time algorithm, for any $\epsilon > 0$, that tests the "towering property:" do n given red lines lie all above n given blue lines? The tools used to obtain these results include Plücker coordinates for lines in space and random sampling in geometric problems.

(24 pages, January 1990)

TR-295-90 Distributed Processing of Filtering Querines in HyperFile
Chris Clifton and Hector Garcia-Molina
Documents, pictures, and other such non-quantitative information pose interesting new problems in the database world. We have developed a language for queries which serves as an extension of the browsing model of hypertext systems. The query language and data model fit naturally into a distributed environment. We discuss a simple and efficient method for processing distributed queries in this language. Results of experiments run on a distributed data server using this algorithm are presented. (22 pages, November 1990)

TR-296-90 Some Fast Algorithms on Graphs and Trees
Heather Donnell Booth (Thesis)
Presented are three fast algorithms solving problems concerning trees or graphs. We give a linear-time algorithm for finding a balanced decomposition of a k-ary tree where k is a constant not related to the imput size. A different linear-time algorithm for finding a balanced decomposition of a binary tree was discovered previously by Guibas, Leven, Hershberger, Sharir and Tarjan. We also consider the problem of finding a minimum-cost maximum flow in a series-parallel network. The algorithm presented here runs in $O(m \log\log m)$ time on an m-edge series-parallel network and requires $O(m \log\log m)$ space. This is an improvement over the algorithm presented by Bein, Brucker and Tamir, which runs in $O(m \log m + mn)$ time, where n is the number of vertices. In an effort to improve the space requirement we also present an algorithm which uses $O(m)$ space but runs in $O(m \log m \log\log m)$ time. Finally, we consider the problem of finding all replacement edges for a minimum spanning tree of a planar graph. We present an algorithm for solving this problem which runs in linear time. This algorithm also performs sensitivity analysis for the minimum spanning tree, shortest path, and network flow problems. The first two algorithms presented rely on the use of balanced binary trees for efficient representation of data. We give an overview of the relevant red-black tree and finger tree techniques in introductory chapter. (104 pages, January 1991)

TR-297-90 A New Specification Model for Timing Constraints and Efficient Methods for their Verification
Dimitris Doukas (Thesis)
Correct timing is a critical issue in hardware design, especially in the case of asynchronous systems such as buses. As a result timing verification becomes an important ingredient of the circuit analysis problem. Furthermore, given the non-trivial timing constraints characterizing the temporal behavior of interface circuits, it is clear that the problem of timing verification is tightly bound to the problem of temporal behavior specification. In this dissertation we present a new specification model which significantly extends the type of constraints which can be expressed. The model is based on dependency graphs and vectors of signal values over time. The timing constraints are verified using a new verification methodology based on "event graphs". Our verification method is to compare the intended circuit

behavior, described in our specification model, with the behavior of the implemented circuit. In order to obtain the behavior of the implemented circuit we use an event driven simulation approach. The implemented behavior of the circuit is derived in the form of an "event graph." To avoid the penalty of an exhaustive simulation, an extended value system is used. The specification model and the event graph verification methodology are integrated into a new timing verification tool: CLOVER. CLOVER provides the designer of a digital circuit with an integrated environment where he can describe his design, formally specify the timing constraints governing the implementation, obtain the timing behavior of the implemented design and verify it against the stated timing constraints. Although the current system does not allow hierarchical verification, the problem of designing a good hierarchical verification methodology is also addressed in this dissertation. CLOVER has been used for the analysis and verification of two interface circuits: the Multibus Design Frame and the SPUR PCC-to-SPC interface. We present a summary of the results for these circuits. (152 pages, January 1991)

TR-298-90 A Free-Market Exchange for Information

R. J. Lipton and J. S. Sandberg

All members of the high-technology community possess a wide variety of valuable information. Such information ranges from operation procedures for Computer Aided Manufacturing hardware to a 100 line spreadsheet program. Some world class high-technology experts (e.g. Nobel Prize winners or Industry leaders) possess unique information or insights that cannot be independently reproduced. Other members of the high-technology community familiar with specific subsystems of profitable or critical products, possess information that can only be reproduced at significant expense. Examples of such information include: specific development information about a software module used in a large defense system, the curvature of a mirror deployed in a satellite, or the location in the company database of last month's sales forecast. The value of the information possessed by employees of profitable corporations, researchers in leading laboratories, or administrators in government offices is hard to overestimate. "You can take my factories," boasted Henry Ford, "burn up my buildings, but give me my people and I'll build the businesses back again." People know the approximate value of a given piece of information. They generally have some informal mechanism for trading (and sometimes selling) the information, although frequently, there is no formally established marketplace for them to sell much of the valuable information they possess. We propose to establish an electronic marketplace for a broad class of high-technology information including such topics as: computer software and hardware, electronic components, high-technology consumer products, systems integration, and data communications systems. More specifically we define and discuss a free market-based Information Exchange for software systems development and maintenance that promises to increase the flow of valuable information through laboratory, company, and international computer networks. Establishing a worldwide electronic network for purchasing technical information will accelerate the pace of innovation and the degree of adaptation of computing, communications, and information technologies in analysis, design, and manufacturing processes. Perhaps most importantly, a international information market will enhance and motivate the human resource base to efficiently meet the economic and technological needs of the new decade. (14 pages, December 1990)

TR-299-90 Fully Persistent Lists with Catenation

James R. Driscoll, Daniel D. K. Sleator and Robert E. Tarjan

In this paper we consider the problem of efficiently implementing a set of side-effect-free procedures for manipulating lists. These procedures are:

$makelist(d)$**:** Create and return a new list of length 1 whose first and only element is d.

$first(X)$**:** Return the first element of list X.

$pop(X)$**:** Return a new list that is the same as list X with its first element deleted. This operation does not effect X.

$catenate(X, Y)$**:** Return a new list that is the result of catenating list X and list Y, with X first, followed by Y (X and Y may be the same list). This operation has no effect on X or Y.

We show how to implement these operations so that $makelist(d)$ runs in constant time and consumes constant space, and $catenate(X, Y)$ runs in time (and consumes space) proportional to $\log\log k$, where k is the number of list operations done up to the time of the catenation. All of these time and space bounds are amortized over the sequence of operations. (17 pages, December 1990)

TR-300-90 Computational Kinematics
Leo Joskowicz and Elisha Sacks
We present an implemented computational theory of the kinematic analysis of mechanisms composed of rigid parts, such as door locks, gearboxes, and transmissions. The computational complexity of kinematic analysis depends on the shapes and motion types of the parts of the mechanism, and is in general intractable. We identify engineering restrictions on the shapes and motion types that make the analysis feasible. We show that most mechanisms satisfy the restrictions by surveying 2500 mechanisms in a standard mechanical engineering encyclopedia. We describe an efficient kinematic analysis program for feasible mechanisms. The input is the shapes and the initial positions of the parts. The output is a concise symbolic description of the mechanism's configuration space, which characterizes its kinematic behavior. The program computes the configuration space by identifying potential degrees of freedom, constructing configuration spaces for all pairs of interacting parts, and incrementally composing the pairwise configuration spaces. (35 pages, December 1990)

TR-301-91 A Linear-Time Algorithm for Finding an Ambitus
B. Mishra and R. E. Tarjan
We devise a linear-time algorithm for finding an ambitus in an undirected graph. An ambitus is a cycle in a graph containing two distinguished vertices such that certain different groups of bridges (called B^P-, B^Q- and B^{PQ}-bridges) satisfy the property that a bridge in one group does not interlace with any bridge in the other groups. Thus, an ambitus allows the graph to be cut into pieces, where, in each piece, certain graph properties may be investigated independently and recursively, and then the pieces can be pasted together to yield information about these graph properties in the original graph. In order to achieve a good time-complexity for such an algorithm employing the divide-and-conquer paradigm, it is necessary to find an ambitus quickly. We also show that, using ambitus, linear-time algorithms can be devised for abiding-path-finding and nonseparating-induced-cycle-finding problems. (34 pages, January 1991)

TR-302-91 Literate Programming on a Team Project
Norman Ramsey and Carla Marceau
We used literate programming on a team project to write a 33,000-line program for the Synthesizer Generator. The program, Penelope, was written using WEB, a tool designed for writing literate programs. Unlike other WEB programs, many of which have been written by WEB's developer or by individuals, Penelope was not intended to be published. We used WEB in the hope that both our team and its final product would benefit from the advantages often attributed to literate programming. The WEB source served as good internal documentation throughout development and maintenance, and it continues to document Penelope's design and implementation. Our experience also uncovered a number of problems with WEB. (11 pages, February 1991)

TR-303-91 A Retargetable Compiler for ANSI C
Christopher W. Fraser and David R. Hanson
`lcc` is a new retargetable compiler for ANSI C. Versions for the VAX, Motorola 68020, SPARC, and MIPS are in production use at Princeton University and at AT&T Bell Laboratories. With a few exceptions, little about `lcc` is unusual – it integrates several well-engineered, existing techniques – but it is smaller and faster than most other C compilers, and it generates code of comparable quality. `lcc`'s target-independent front end performs a few simple, but effective, optimizations that contribute to good code; examples include simulating register declarations and partitioning switch statement cases into dense tables. It also implements target-independent function tracing and expression-level profiling. (20 pages, February 1991)

TR-304-91 Efficient, Scalable Architectures for Lattice-Gas Computations
Richard K. Squier (Thesis)
The subject of this dissertation is finding an architecture for two-dimensional cellular automata computations that is verifiably correct, and the fastest and cheapest possible. The motivating problems for this work are large-scale scientific computations; the hardware context is that of application-specific processors attached to general-purpose systems. While the conclusions are derived for a specific class of two-dimensional cellular automata, the so-called "lattice gasses," they are applicable to a wide range of similar problems. By developing and applying solutions to discrete isoperimetric problems to a pebbling game, upper bounds on throughput for machines computing problems with data dependency graphs based on the undirected graphs for the Hardy-de Pazzis-Pomeau (HPP) and Frisch-Hasslacher-Pomeau

(FHP) lattice-gasses are shown. A particular architecture, the "Wide Serial Architecture" (WSA), is shown to be within a factor of approximately 6 of the bound for the HPP-like computations, and within a factor of about 4.5 of the bound for the FHP-like computations. Besides the insight they provide into the optimal computational strategy, the methods of solution of the isoperimetric problems, such as the use of the Wulff Crystal, are of interest in their own right. An analytic study of the least-cost configuration for a multiple-pipeline WSA machine is undertaken. The use of overlap-save method is described, and the efficiency of the WSA architecture using this method is derived. A numerical search is used to find the least cost machine configuration as the problem size is scaled. A slightly super-linear speedup is shown over a moderate range of problem sizes, and the least-cost machine parameters are described. Finally, a data-embedded, specification-based testing technique for FHP lattice gasses is introduced. The tests consist of limit cycles in the cellular automaton, and their error detection coverage is shown empirically to be good. Their use in software, hardware, and system debugging is described, as well as their use as runtime simulation error detectors. Machine design tradeoffs relating to testability are discussed. (182 pages, June 1991)

TR-305-91 Markov Analysis of Qualitative Dynamics
Jon Doyle and Elisha P. Sacks
Commonsense sometimes predicts events to be likely or unlikely rather than merely possible. We extend methods of qualitative reasoning to predict the relative likelihoods of possible qualitative behaviors by viewing the dynamics of a system as a Markov chain over its transition graph. This involves adding qualitative or quantitative estimates of transition probabilities to each of the transitions and applying the standard theory of Markov chains to distinguish persistent states from transient states and to calculate recurrence times, settling times, and probabilities for ending up in each state. Much of the analysis depends solely on qualitative estimates of transition probabilities, which follow directly from theoretical considerations and which lead to qualitative predictions about entire classes of systems. Quantitative estimates for specific systems are derived empirically and lead to qualitative and quantitative conclusions, most of which are insensitive to small perturbations in the estimated transition probabilities. The algorithms are straightforward and efficient. (24 pages, February 1991)

TR-306-91 String Processing Languages
Ralph E. Griswold and David R. Hanson
In most traditional programming languages, character strings are arrays of characters, and string processing is programmed with low-level array operations, such as indexing. In high-level string-processing languages, strings are treated at a higher conceptual level. Strings are first-class values in these languages, and most offer a rich set of operations that manipulate strings as atomic values. Examples include concatenation, substring identification, transformation, and pattern matching. This report surveys some of the past and present string-processing and their facilities that deal with strings. Included are brief descriptions of Comit, the SNOBOL languages, and Icon. This report will appear in the third edition of the *Encyclopedia of Computer Science and Engineering*, to be published by Van Nostrand Reinhold. (18 pages, February 1991)

TR-307-91 Evaluation of Memory System Extensions
Kai Li and Karin Petersen
A traditional memory system for a uniprocessor consists of one or two levels of cashe, a main memory and a backing store. One can extend such a memory system by adding inexpensive but slower memories into the memory hierarchy. This paper uses an experimental approach to evaluate two methods of extending a memory system: direct and caching. The direct method adds the slower memory into the memory hierarchy by putting it at the same level as the main memory, allowing the CPU to access the slower memories directly; whereas the caching method puts the slower memory between the main memory and the backing store, using the main memory as a cache for the slower memory. We have implemented both approaches and our experiments indicate that applications with very large data structures can benefit significantly using an extended memory system, and that the direct approach outperforms the caching approach in memory-bound applications. (10 pages, March 1991)

TR-308-91 A Note on Matrix Rigidity
Joel Friedman
In this paper we give an explicit construction of $n \times n$ matrices over finite fields which are somewhat rigid, in that if we change at most k entries in each row, its rank remains at least $Cn(\log_q k)/k$, where q

is the size of the field and C is an absolute constant. Our matrices satisify a somewhat stronger property, which we explain and call "strong rigidity." We introduce and briefly discuss strong rigidity, because it is in a sense a simpler property and may be easier to use in giving explicit constructions. (6 pages, June 1990)

TR-309-91 An $O(m \log n)$-Time Algorithm for the Maximal Planar Subgraph Problem
Jiazhen Cai, Xiaofeng Han and Robert E. Tarjan
Based on a recursive version of Hopcroft and Tarjan's planarity testing algorithm, we develop an $O(m \log n)$-time algorithm to find a maximal planar subgraph. (25 pages, March 1991)

TR-310-91 Dynamic Perfect Hashing: Upper and Lower Bounds
Martin Dietzfelbinger, Anna Karlin, Kurt Mehlhorn, Friedhelm Meyer auf der Heide, Hans Rohnert and Robert E. Tarjan
The dynamic dictionary problem is considered: provide an algorithm for storing a dynamic set, allowing the operations insert, delete, and lookup. A dynamic perfect hashing strategy is given: a randomized algorithm for the dynamic dictionary problem that takes $O(1)$ worst-case time for lookups and $O(1)$ amortized expected time for insertions and deletions; it uses space proportional to the size of the set stored. Furthermore, lower bounds for the time complexity of a class of deterministic algorithms for the dictionary problem are proved. This class encompasses realistic hashing-based schemes that use linear space. Such algorithms have amortized worst-case time complexity $\Omega(\log n)$ for a sequence of n insertions and lookups; if the worst-case lookup time is restricted to k then the lower bound becomes $\Omega(k \cdot n^{1/k})$. (34 pages, March 1991)

TR-311-91 Efficient Maximum Flow Algorithms
Robert E. Tarjan
Discovering efficient algorithms to compute flows in networks has been a goal of researchers in operations research and computer science for over 35 years. In this paper we review some recent developments in algorithms for the single-commodity maximum flow problem, and we comment on possible additional improvements. Our criterion for efficiency is theoretical worst-case running time on large sparse problems, ignoring constant factors. (10 pages, March 1991)

TR-312-91 Ordered and Reliable Multicast Communication
Annemarie Spauster (Thesis)
Multicasting (sending a message to a subset of sites in a network) has become a popular mechanism for interprocess communication in distributed systems. Many applications (e.g., distributed databases) require that a message be transmitted to multiple processes. One indisputably desirable quality of any type of message transmission is reliability. A message that is sent from process A to process B should indeed arrive. Even better, it should arrive within a reasonable amount of time. For distributed applications, it is also often desirable for multidestination messages to arrive at the destination processes in a consistent order. In a database application, if update requests are headed to two destinations with copies of the data, delivering the requests in the same order at both destinations helps maintain consistency. This is just one example of how consistent message delivery simplifies distributed applications. In this thesis, we consider enhancing multicast communication by providing ordering and reliability properties. We present an algorithm, the propagation graph algorithm, that guarantees a strong ordering property: multiple group ordering. Experimental analysis of the propagation graph technique demonstrates that it is efficient compared to other solutions and exhibits a clear load/delay tradeoff. We also present several types of reliability that multicast ordering algorithms can provide. We address how to achieve these types of reliability with the propagation graph algorithm. Further, we clarify the issue of reliability by presenting a formal model of message ordering and by presenting formal definitions of reliability properties. These properties are then applied to the propagation graph solution. (119 pages, June 1991)

TR-313-91 Probabilistic Behavior of Shortest Paths Over Unbounded Regions
Andrew Chi-Chih Yao
Let $k > 1$ and P be a probability distribution over R^k with all its absolute μ-th moments being finite for some $\mu > k/(k-1)$. Let $v_1, v_2, \ldots$ be an infinite sequence of random points, each independently distributed according to P. It is shown that the length of the shortest traveling-salesman's tour through $v_1, v_2, \ldots, v_n$ is, for large n, almost surely around $\alpha_P n^{(k-1)/k}$ for some constant α_P. This proves a conjecture of Beardwood, Halton and Hammersley (*Proc. Camb. Phil. Soc.* **55** (1959), 299-327).

(18 pages, March 1991)

TR-314-91 Prolegomena to any Future Qualitative Physics
Elisha P. Sacks and Jon Doyle
We evaluate the success of the qualitative physics enterprise in automating expert reasoning about physical systems. The field has agreed, in essentials, upon a modeling language for dynamical systems, a representation for behavior, and an analysis method. The modeling language consists of generalized ordinary differential equations containing unspecified constants and monotonic functions; the behavioral representation decomposes the state space described by the equations into discrete cells; and the analysis method traces the transitory response using sign arithmetic and calculus. The field has developed several reasoners based on these choices over some fifteen years. We demonstrate that these reasoners exhibit severe limitations in comparison with experts and can analyze only a handful of simple systems. We trace the limitations to inappropriate assumptions about expert needs and methods. Experts ordinarily seek to determine asymptotic behavior rather than transient response, and use extensive mathematical knowledge and numerical analysis to derive this information. Standard mathematics provides complete qualitative understanding of many systems, including those addressed so far in qualitative physics. Preliminary evidence suggests that expert knowledge and reasoning methods can be automated directly, without restriction to the accepted language, representation and algorithm. We conclude that expert knowledge and methods provide the most promising basis for automating qualitative reasoning about physical systems. (31 pages, April 1991)

TR-315-91 Checkpointing Multicomputer Applications
Kai Li, Jeffrey F. Naughton and James S. Plank
Efficient checkpointing and resumption of multicomputer applications is essential if multicomputers are to support time-sharing and the automatic resumption of jobs after a system failure. We present a checkpointing scheme that is transparent, imposes overhead only during checkpoints, requires minimal message logging, and allows for quick resumption of execution from a checkpointed image. Since checkpointing multicomputer applications poses requirements different from those posed by checkpointing general distributed systems, existing distributed checkpointing schemes are inadequate for multicomputer checkpointing. Our checkpointing scheme makes use of special properties of multicomputer interconnection networks to satisfy this new set of requirements. (16 pages, April 1991)

TR-316-91 Reliable Reconfigurable Structures for Array Architectures
Edwin Hsing-Mean Sha and Kenneth Steiglitz
This paper describes some explicit constructions for reconfigurable array architectures. Given a working architecture (application graph), we add redundant hardware to increase reliability. The degree of reconfigurability, DR, of a redundant graph is a measure of the cost of reconfiguration after failures. When DR is independent of the size of the application graph, we say the graph is finitely reconfigurable, FR. We present a class of simple layered graphs with a logarithmic number of redundant edges, which can maintain both finite reconfigurability and a fixed level of reliability for a wide class of application graphs. By sacrificing finite reconfigurability, we show that by using expanders we can construct highly reliable structures with the asymptotically optimal number of edges for one-dimensional and tree-like array architectures. (24 pages, April 1991)

TR-317-91 Lecture Notes on Evasiveness of Graph Properties
László Lovász and Neal Young
These notes cover the first eight lectures of the class *Many Models of Complexity* taught by László Lovász at Princeton University in the Fall of 1990. The first eight lectures were on evasiveness of graph properties and related topics; subsequent lectures were on communication complexity and Kolmogorov complexity and are covered in other sets of notes. The fundamental question considered in these notes is, given a function, how many bits of the input an algorithm must check in the worst case before it knows the value of the function. The algorithms considered are deterministic, randomized, and non-deterministic. The functions considered are primarily graph properties — predicates on edge sets of graphs invariant under relabeling of the edges. (49 pages, April 1991)

TR-318-91 Randomized Parallel Algorithms for Trapezoidal Diagrams
Kenneth L. Clarkson, Richard Cole and Robert E. Tarjan
We describe randomized parallel CREW PRAM algorithms for building trapezoidal diagrams of line

segments in the plane. For general segments, we give an algorithm requiring optimal $O(A + n \log n)$ expected work and optimal $O(\log n)$ time, where A is the number of intersecting pairs of segments. If the segments form a simple chain, we give an algorithm requiring optimal $O(n)$ expected work and $O(\log n \log\log n \log^* n)$ expected time, and a simpler algorithm requiring $O(n \log^* n)$ expected work. The serial algorithm corresponding to the latter is the simplest known algorithm requiring $O(n \log^* n)$ expected operations. For a set of segments forming K chains, we give an algorithm requiring $O(A + n \log^* n + K \log n)$ expected work and $O(\log n \log\log n \log^* n)$ expected time. The parallel time bounds require the assumption that enough processors are available, with processor allocations every $\log n$ steps. (10 pages, April 1991)

TR-319-91 Empirical Studies of Competitive Spinning for Shared-Memory Multiprocessors
Anna R. Karlin, Kai Li, Mark S. Manasse and Susan Owicki
A common operation in multiprocessor programs is acquiring a lock to protect access to shared data. Typically, the requesting thread is blocked if the lock it needs is held by another thread. The cost of blocking one thread and activating another can be a substantial part of program execution time. Alternatively, the thread could spin until the lock is free, or spin for a while and then block. This may avoid context-switch overhead, but processor cycles may be wasted in unproductive spinning. This paper studies seven strategies for determining whether and how long to spin before blocking. Of particular interest are competitive strategies, for which the performance can be shown to be no worse than some constant factor times an optimal off-line strategy. The performance of five competitive strategies is compared with that of always blocking, always spinning, or using the optimal off-line algorithm. Measurements of lock-waiting time distributions for five parallel programs were used to compare the cost of synchronization under all the strategies. Additional measurements of elapsed time for some of the programs and strategies allowed assessment of the impact of synchronization strategy on overall program performance. Both types of measurements indicate that the standard blocking strategy performs poorly compared to mixed strategies. Among the mixed strategies studied, adaptive algorithms perform better than non-adaptive ones. (21 pages, March 1991)

TR-320-91 The Demarcation Protocol: A Technique for Maintaining Arithmetic Constraints in Distributed Database Systems
Daniel Barbará and Hector Garcia-Molina
Traditional protocols for distributed database management have high message overhead, lock or restrain access to resources during protocol execution, and may become impractical for some scenarios like real-time systems and very large distributed databases. In this paper we present the demarcation protocol; it overcomes these problems through the use of explicit arithmetic consistency constraints as the correctness criteria. The method establishes safe limits as "lines drawn in the sand" for updates and gives a way of changing these limits dynamically, enforcing the constraints at all times. (24 pages, April 1991)

TR-321-91 Long-Term Caching Strategies for Very Large Distributed File Systems
Matt Blaze and Rafael Alonso
This paper examines the feasibility of using long term (disk based) caches in very large distributed file systems (DFSs). We begin with an analysis of file access patterns in a distributed Unix workstation environment, and identify properties of use to the DFS designer. We then introduce long-term caching strategies that maintain consistency while dramatically reducing the load on file servers. We describe a number of algorithms for maintaining client caches, and present the results of a trace-driven simulation that shows how relatively small disk-based caches can be used to reduce server traffic by 60–90%. Finally, we outline possible mechanisms for dynamically organizing these caches into adaptive hierarchies to allow arbitrary scaling of the number of clients and the use of low-bandwidth communication networks. A small (2 or 3 level) hierarchy, coupled with smart caching techniques, has the potential to reduce traffic by an order of magnitude or more over a flat scheme. (13 pages, April 1991)

TR-322-91 An Algorithmic Approach to Extremal Graph Problems
Xiaofeng Han (Thesis)
The main purpose of this thesis is to study three problems concerning extremal graphs. The first is the problem of finding a minimal 2-edge connected subgraph of a 2-edge connected graph, the second is the problem of finding a minimal 2-connected subgraph of a 2-connected graph, and the third is the problem of finding a maximal planar subgraph of a nonplanar graph. These problems are extensions of the 2-edge connectivity problem, the 2-connectivity problem, and the planarity testing problem in graph theory.

All three problems are important in the theory of extremal graphs. They also have useful applications in computer network and electronic circuit design. (130 pages, June 1991)

TR-323-91 Hyperfile, A Database Manager for Documents
Christopher Wade Clifton
Documents, pictures, and other such non-quantitative information pose interesting new problems in the database world. Such data has traditionally been stored in file systems, which do not provide the security, integrity, or query features of database management systems. We have developed HyperFile, a data server that provides query facilities (as well as some other database features) while maintaining the flexibility and efficiency of a file system. HyperFile is based on the hypertext notion of free-form objects connected by links. Hypertext systems "query" their database by browsing (reading objects and following links.) We present a query interface that maintains much of the flavor of browsing, allowing the user to specify a single query rather than manually following links. This eliminates the repeated user interactions of hypertext browsing, and allows the hypertext model to be extended to larger and less structured databases. An algorithm for processing HyperFile queries is presented. We also show how to extend this algorithm for distributed query processing, and present experimental results from a distributed HyperFile server. Another issue explored is indexing. In HyperFile, searches are often demarcated by pointers between items. Thus the scope of the search may change dynamically, whereas traditional indexes cover a statically defined region such as a relation. This demands new indexing techniques. Some ideas on indexing in HyperFile are presented, as well as experiments in a large HyperFile database. Also presented is a sample HyperFile application. This is a "browser" that uses menus to guide the user in constructing HyperFile queries. (104 pages, June 1991)

TR-324-91 Garbage Collection Alternatives for Icon
Mary F. Fernandez and David R. Hanson
Copying garbage collectors are becoming the collectors of choice for very high-level languages and for functional and object-oriented languages. Copying collectors are particularly efficient for large heaps because their execution time is proportional only to the amount of accessible data, and they identify and compact this data in one pass. In contrast, mark-and-sweep collectors execute in time proportional to the heap size and require a second pass to compact accessible data. The performance of existing systems with old mark-and-sweep collectors might be improved by replacing their collectors with copying collectors. This paper explores this possibility by describing the results of replacing the mark-and-sweep collector in the Icon programming language with four alternative collectors, three of which are copying collectors. Copying collectors do indeed yield better performance, but at a significant cost in space. Measurements suggest that the mark-and-sweep alternative is best for Icon programs that have 1–5MB heaps. (16 pages, May 1991)

TR-325-91 Computations Over Infinite Groups
Jin-yi Cai
We propose the study of a wide variety of infinite groups from a computational complexity point of view. We raise some important structural questions on these groups from a computational aspect. The purpose of this paper is to invite the attention of both the theoretical computer science community and the combinatorial group theorists that a fruitful area of cross fertilization may be offering itself. As a specific problem, we consider randomly generated groups and their isomorphism problem. (11 pages, June 1991)

TR-326-91 Callee-save Registers in Continuation-Passing Style
Andrew W. Appel and Zhong Shao
Continuation-passing style (CPS) is a good abstract representation to use for compilation and optimization: it has a clean semantics and is easily manipulated. We examine how CPS expresses the saving and restoring of registers in source-language procedure calls. As CPS-conversion is usually written, the context of the calling procedure is saved in a "continuation closure" – a single variable that is passed as an argument to the function being called. This closure is a record containing bindings of all the free variables of the continuation; that is, registers that hold values needed by the caller "after the call" are written to memory in the closure, and fetched back after the call. Consider the procedure-call mechanisms used by conventional compilers. In particular, registers holding values needed after the call must be saved and later restored. The responsibility for saving registers can lie with the caller (a "caller-saves" convention) or with the called function (a "callee-saves"). In practice, to optimize memory traffic compilers find it

useful to have some caller-saves registers and some callee-saves. Clearly the usual translation into CPS is a caller-saves convention. We explain how to express callee-save registers in Continuation-Passing Style, and give measurements showing the resulting improvement in execution time. (16 pages, August 1991)

TR-327-91 Polygon Triangulation in $O(N \log\log N)$ Time with Simple Data Structures
David G. Kirkpatrick, Maria M. Klawe and Robert E. Tarjan
We give a new $O(n \log\log n)$-time deterministic algorithm for triangulating simple n-vertex polygons, which avoids the use of complicated data structures. In addition, for polygons whose vertices have integer coordinates of polynomially bounded size, the algorithm can be modified to run in $O(n \log^* n)$ time. The major new techniques employed are the efficient location of horizontal visibility edges that partition the interior of the polygon into regions of approximately equal size, and a linear-time algorithm for obtaining the horizontal visibility partition of a subchain of a polygonal chain, from the horizontal visibility partition of the entire chain. The latter technique has other interesting applications, including a linear-time algorithm to convert a Steiner triangulation of a polygon into a true triangulation. (16 pages, June 1991)

TR-328-91 Probabilistic Diagnosis of Hot Spots
Kenneth Salem, Daniel Barbará and Richard J. Lipton
Commonly, a few objects in a database account for a large share of all database accesses. These objects are called hot spots. The ability to determine which objects are hot spots opens the door to a variety of performance improvements. Data reorganization, migration, and replication techniques can take advantage of knowledge of hot spots to improve performance at low cost. In this paper we present some techniques that can be used to identify those objects in the database that account for more than a specified percentage of database accesses. Identification is accomplished by analyzing a string of database references and collecting statistics. Depending on the length of the reference string and the amount of space available for the analysis, each technique will have a non-zero probability of false diagnosis, i.e., mistaking "cold" items for hot spots and vice versa. We compare the techniques analytically and show the tradeoffs among time, space and the probability of false diagnoses. (28 pages, June 1991)

TR-329-91 Standard ML of New Jersey
Andrew W. Appel and David B. MacQueen
The Standard ML of New Jersey compiler has been under development for five years now. We have developed a robust and complete environment for Standard ML that supports the implementation of large software systems and generates efficient code. The compiler has also served as a laboratory for developing novel implementation techniques for a sophisticated type and module system, continuation based code generation, efficient pattern matching, and concurrent programming features. (13 pages, June 1991)

TR-330-91 The Analysis of Heapsort
Russel Schaffer and Robert Sedgewick
Heapsort is a fundamental algorithm whose precise performance characteristics have proven difficult to analyze. It is easy to show that the number of keys moved during the algorithm when sorting a random file of N distinct elements is $N \lg N + O(N)$ in the worst case, and it has long been conjectured that the average case performance is the same. No specific results on the average case or even the best case have been found despite the algorithm's standing as a classic method that is in widespread use. In this paper, we resolve these questions by showing that the best case for the number of moves is $\sim \frac{1}{2} N \lg N$) and that the average number of moves is $\sim N \lg N$. This essentially completes the analysis of the algorithm, though there is another quantity that contributes to the leading term of the running time that requires more intricate arguments, and we have little specific information about the distribution beyond what is implied by our asymptotic results. These results were initially suggested by results of an empirical study that involved generating and analyzing hundreds of billions of heaps. The distribution of the number of moves required by Heapsort for small N is presented in an Appendix. (13 pages, January 1991)

TR-331-91 Scheduling Real-Time Transactions: A Performance Evaluation
Robert Kilburn Abbott (Thesis)
This thesis describes a new group of algorithms for scheduling real-time transactions, or transactions with deadlines. The algorithms are evaluated via a detailed simulation study. Our results show that under a wide range of workloads, the real-time algorithms perform significantly better than a conventional

transaction scheduling algorithm. This thesis also studies the problem of scheduling disk requests with deadlines. Three new algorithms are proposed and their performance is evaluated via simulation. One real-time algorithm, FD-SCAN, was found to perform better than all algorithms tested, both real-time and conventional, under a wide range of experimental conditions. (331 pages, October 1991)

TR-332-91 An Euclidean Metric for Genetic Sequence Comparison
K. Balasubramanian (Thesis)
This thesis introduces a new representation for genetic sequences in the form of geometric points or vectors. It is based on the relative frequencies of the various (small) fixed length substrings of $k-tuples$ of the DNA or protein sequences. This effectively transforms the sequence from a variable sized string to fixed size vector. We show that this transformation preserves, under certain circumstances, the edit distince between sequences, a widely used measure for comparing genetic sequences. This fact is used to develop a linear time heuristic for sequence comparison, an improvement over the quadratic time dynamic programming based algorithms currently in widespread use. The transformation from a variable sized sequence to a fixed size vector representation allows computational geometric techniques to be applied to the study of genetic sequences. In particular, we develop a method of comparing several sequences simulatneously without having to compare each pair of sequences separately. This results in a substantial reduction in the complexity of the problem of multiple sequence comparison and clustering. This can be applied as a filter to extract interesting sets of sequences from a large database for more thorough study as well as an indexing method, to locate the database sequences most likely to be related to a query sequence. (103 pages, October 1991)

TR-333-91 Point Location Among Hyperplanes and Unidirectional Ray-Shooting
Bernard Chazelle and Joel Friedman
We present an algorithm for locating a query point q in an arrangement of n hyperplanes in d-space. The size of the data structure is $O(n^d)$ and the time to answer any query is $O(\log n)$. Unlike previous data structures, our solution will also report, in addition to the face of the arrangement that contains q, the first hyperplane that is hit (if any) by shooting the point q in some fixed direction. Actually, if this ray-shooting capability is all that is needed, or if one only desires to know a single vertex of the face enclosing q, then the storage can be reduced to $O(n^d/(\log n)^{\lceil d/2\rceil-\epsilon})$, for any fixed $\epsilon > 0$. (9 pages, June 1991)

TR-334-91 Computing a Face in an Arrangement of Line Segments and Related Problems
Bernard Chazelle, Herbert Edelsbrunner, Leonidas Guibas, Micha Sharir and Jack Snoeyink
We present a randomized incremental algorithm for computing a single face in an arrangement of n line segments in the plane that is fairly simple to implement. The expected running time of the algorithm is $O(n\alpha(n)\log n)$. The analysis of the algorithm uses a novel approach that generalizes and extends the Clarkson-Shor analysis technique. We also present a few extensions of the technique, obtaining efficient randomized incremental algorithms for constructing the entire arrangement of a collection of line segments, and for computing a single face in an arrangement of Jordan arcs. (20 pages, June 1991)

TR-335-91 Cutting Hyperplanes for Divide-and-Conquer
Bernard Chazelle
Given n hyperplanes in E^d, a $(1/r)$-cutting is a collection of simplices which together cover E^d and such that the interior of each simplex intersects at most n/r hyperplanes. We present an algorithm for computing a $(1/r)$-cutting of (asymptotically) minimum size in $O(nr^{d-1})$ time. If, as is the case in practice, the lists of cutting hyperplanes must be explicitly provided, then the algorithm is optimal. Our result bridges a gap in a recent algorithm of Matousek by extending its performance to all values of r; the previous bound was restricted to $r \leq n^{1-\delta}$, for any fixed $\delta > 0$. To attain our goal, we show that optimal cuttings can be refined by composition. This is interesting in its own right, because it leads to the improvement and the simplification of a number of geometric algorithms, e.g., point location among hyperplanes, counting segment intersections, Hopcroft's line/point incidence problem, linear programming in fixed dimension. One of the main tools used in the cutting construction is a proof that ϵ-approximations can be used to estimate how many vertices of a hyperplane arrangement fall inside a given simplex. In a different development, to be reported elsewhere, we have used this lemma to derive an optimal deterministic algorithm for computing convex hulls in higher dimensions. Thus, we suspect that the lemma will have other useful applications in computational geometry. (14 pages, June 1991)

TR-336-91 An Optimal Convex Hull Algorithm for Points Sets in Any Fixed Dimension
Bernard Chazelle

We present a deterministic algorithm for computing the convex hull of n points in E^d in optimal $O(n^{\lfloor d/2 \rfloor})$ time, for $d < 3$. This result settles an open question of long standing: optimal solutions were previously known only in two and three dimensions or alternatively by allowing randomization. The algorithm involves a fairly elaborate dynamic search process, whose fine points are clarified by using an analogy with statistical thermodynamics: this allows us to uncover some unexpected phenomena relating to the convergence of the algorithm. By duality, the algorithm can be used to construct the full lattice structure of the feasible set of n linear constraints in optimal $O(n \log n + n^{\lfloor d/2 \rfloor})$ time, for any fixed d. As an immediate corollary, we obtain an algorithm for computing the Voronoi diagram of n points in d-space in optimal $O(n \log n + n^{\lceil d/2 \rceil})$ time, which is also a new result. (36 pages, June 1991)

TR-337-91 Human Language Comprehension is NP-Complete
Eric Sven Ristad

Consider the computational problem of understanding the utterances of a human language that contain pronouns. In order to completely understand such utterances, the language user must determine the intended reference of each pronoun in a given utterance. For example, in order to comprehend the english sentence "Jocasta loved her son," the hearer might determine that the possessive pronoun her refers to Jocasta, the queen of Thebes. We prove that two different models of this broad computational problem are NP-hard, and argue that the problem remains in NP even if our language models account for the computationally complex phenomenon of syntactic ellipsis. These complexity results are based directly on human linguistic knowledge, and are invariant across linguistic theories. No knowledge of linguistic theory is needed to understand the analysis, only knowledge of English. (42 pages, June 1991)

TR-338-91 Improved Algorithms for Bipartite Network Flow
Ravindra K. Ahuja, James B. Orlin, Clifford Stein and Robert E. Tarjan

In this paper, we study network flow algorithms for bipartite networks. A network $G = (V, E)$ is called bipartite if its vertex set V can be partitioned into two subsets V_1 and V_2 such that all edges have one endpoint in V_1 and the other in V_2. Let $n = |V|, n_1 = |V_1|, n_2 = |V_2|, m = |E|$ and assume without loss of generality that $n_1 \leq n_2$. We call a bipartite network unbalanced if $n_1 \ll n_2$ and balanced otherwise. (This notion is necessarily imprecise.) We show that several maximum flow algorithms can be substantially sped up when applied to unbalanced networks. The basic idea in these improvements is a two-edge push rule that allows us to "charge" most computation to vertices in V_1, and hence develop algorithms whose running times depend on n_1 rather than n. For example, we show that the two-edge push version of Goldberg and Tarjan's FIFO preflow push algorithm runs in $O(n_1 m + n_1^3)$ time and that the analogous version of Ahuja and Orlin's excess scaling algorithm runs in $O(n_1 m + n_1^2 \log U)$ time, where U is the largest edge capacity. We also extend our ideas to dynamic tree implementations, parametric maximum flows, and minimum-cost flows. (41 pages, May 1991)

TR-339-91 Error Detection in Arrays Via Dependency Graphs
Edwin Hsing-Mean Sha and Kenneth Steiglitz

This paper describes a methodology based on dependency graphs for doing concurrent run-time error detection in systolic arrays and wavefront processors. It combines the projection method of deriving systolic arrays from dependency graphs with the idea of input-triggered testing. We call the method ITRED, for Input-driven Time-Redundancy Error Detection. Tests are triggered by inserting special symbols in the input, and so the approach gives the user flexibility in trading off throughput for error coverage. Correctness of timing is proved at the dependency graph level. The method requires no extra PE's and little extra hardware. We propose several variations of the general approach and derive corresponding constraints on the modified dependency graphs that guarantee correctness. One variation performs run-time error correction using majority voting. Examples are given, including a dynamic programming algorithm, convolution, and matrix multiplication. (26 pages, August 1991)

TR-340-91 Some Geometric Aspects of Graphs and their Eigenfunctions
Joel Friedman

We study two mathematical notions, that of nodal regions for eigenfunctions of the Laplacian, and that of fiber products, in the context of graph theory. We formulate analogous notions and theorems for graphs and their eigenpairs. These techniques suggest new ways of studying problems related to spectral theory of graphs. (33 pages, November 1991)

TR-341-91 A Smart Interface for Numerical Software
Elisha Sacks
The paper describes a smart interface that makes numerical libraries easy to use and reliable by exploiting mathematical theory, symbolic algebra, and descriptions of the input/output formats of numerical subroutines. The interface accepts a high-level problem description, selects the appropriate numerical subroutine, programs the problem in subroutine format, runs the program, corrects for numerical errors and special conditions, and returns the output in a high-level format. The current interface manages a root finder, a continuation package, an ordinary differential equation integrator, and a Lyapunov exponent calculator. I describe these functions and illustrate their use in a program that analyzes ordinary differential equations. (9 pages, August 1991)

TR-342-91 Automated Analysis of a Heart Model
Elisha Sacks
I analyze Rigney and Goldberger's mathematical model of the heart's rocking during periocardial effusion. The analysis predicts the motion of the heart for heart rates between 50 and 130 and for a range of initial positions and velocities: the heart rocks once every other beat (exhibits alternans) at heart rates between 104 and 117 and rocks once per beat otherwise. The predictions agree with the published clinical data, which shows alternans at heart rates between 100 and 124. They also agree with Rigney and Goldberger's numerical simulation, which shows one rock per beat at heart rate 75 and alternans at heart rate 105. I describe a general purpose analysis program that analyzed the heart model automatically. (10 pages, August 1991)

TR-343-91 Lower Bounds in Geometric Searching
Burton Rosenberg (Thesis)
We study algorithms for geometric range searching, particularly for the problems of reporting and counting points inside of axis-parallel rectangles and simplices in Euclidean d-space. Lower bounds are discussed as well as the models of computation in which the lower bounds hold. The relevance of these models to practical computing is considered. We then present two new lower bounds for geometric range searching. Related to the problem of computing partial sums off-line, we show that given an array A with n entries in an additive semi-group, and m intervals of the form $I_k = [i, j]$, where $0 < i < j \leq n$, then the computation of $A[i] + \ldots + [j]$ for all I_k will require $\Omega(n + m\alpha(m, n))$ semigroup additions. Here α is the functional inverse of Ackermann's function. Related to the problem of simplex range reporting we prove that given a collection P of n points in d-space, any data structure which can be modeled on a pointer machine and which can report the r points inside of an arbitrary d-simplex in time $O(n^{\delta} + r)$ will require storage $\Omega(n^{d(1-\delta)-\epsilon})$, for any fixed $\epsilon > 0$. Both of these lower bounds are tight within small functional factors. (64 pages, October 1991)

TR-344-91 A Theory for Deadlocks
Y. C. Tay and W. Tim Loke
Deadlock detection is an elementary problem in computer science, yet algorithms for detecting deadlocks over resources in a distributed system are curiously prone to errors. This anomaly calls for a theory that will help us understand existing algorithms and design new algorithms. Surprisingly, most papers in the literature have either no difinition of what a deadlock is, or a bad definition; this fact itself accounts for many of the errors. The first task in developing a theory is therefore to choose an appropriate definition for a deadlock. Since this theory is to be used for the analysis and synthesis of detection algorithms, it should focus on what an algorithm can observe (in this sense, it is an operational theory); accordingly, the definition chosen here is in terms of locally observable facts, and does not use real time. The theory begins with a rigorous formulation of the interaction between processes and resources, and its development is via logical deduction, rather than operational arguments. The results examine the effect of aborting processes on deadlock detection, clarify the difference between a process and a resource, and reveal the structure of detection algorithms. To illustrate its application, the theory is used to analyze several errors and algorithms in the literature. (43 pages, August 1991)

TR-345-91 CLOVER: A User Guide
Dimitris Doukas and Andrea S. LaPaugh
CLOVER is a timing verification system for digital systems. It was designed to handle medium size asynchronous systems, particularly interface systems, but can be used with synchronous systems as well. A wide range of basic components can be used with CLOVER, including user-defined components. To

use CLOVER, the designer provides a description of the design in the hardware description language PDL-e and a description of the timing constraints that the design should satisfy in CLOVER's constraint language ATCSL. CLOVER derives an event graph for the design using an event-based timing simulator and checks the satisfaction of each constraint within the event graph. In this report we present a detailed description on the use of CLOVER from the user perspective. A detailed documentation of all CLOVER's features is given, accompanied by simple but illustrative examples. (41 pages, August 1991)

TR-346-91 Computer Science 111
Ken Steiglitz, Jeff Blum and Jonathan Thompson
This is the lab manual for COS 111, an introductory course in computer science for liberal arts students, as it was taught for the first time in the fall semester of 1991. It includes a general introduction, topic outline, a set of nine laboratory exercises for the NeXT, and a collection of CheatSheets that provide supporting material for the labs. The topics are:

1. Introduction
2. Communications [news, mail, ftp, telnet],
3. Graphics [PostScript, lpr, Edit, Yap, TopDraw, etc.],
4. Presentation [WriteNow, troff, etc.],
5. Sound [SoundEditor],
6. Programming in C,
7. Selection sort,
8. More sound, using C, and
9. Information processing [grep, wc, sort].

The lectures themselves included an introduction to complexity theory [big-oh notation, analysis of sorting, etc.] and computability theory [Turing machines, the halting problem, etc.]. (150 pages, September 1991)

TR-347-91 High Performance File System Design
Carl Hudson Staelin (Thesis)
File systems and I/O subsystems should be *smart*; they can analyze how they are being used and tune themselves dynamically to improve their performance. File systems should select caching and disk placement strategies on a per-file basis, and they should use system-wide disk reorganization strategies. For example, systems should be able to reorganize the data on disk automatically during idle periods so that system performance is improved during future periods of peak load. This dissertation presents the design and analysis of iPcress, a prototype of a next-generation file system. iPcress is a smart, high-performance, reliable file system. It uses statistical information collected on a per-file basis to tune itself. iPcress has a framework in which various optimizations can be performed by the file system automatically. It is extensible; other optimization techniques can be incorporated easily, so that the system may evolve. In addition, iPcress can incorporate a variety of file access and placement techniques and choose the best combination of techniques for each file dynamically. A sample smart optimization – clustering active disk data in the center of the disk – is described; it increases disk throughput by 30% (88 pages, October 1991)

TR-348-91 Competitive Paging and Dual-Guided On-Line Weighted Caching and Matching Algorithms
Neal Young (Thesis)
This thesis presents research done by the author on competitive analysis of on-line problems. An on-line problem is a problem that is given and solved one piece at a time. An on-line strategy for solving such a problem must give the solution to each piece knowing only the current piece and preceding pieces, in ignorance of the pieces to be given in the future. We consider on-line strategies that are competitive (guaranteeing solutions whose costs are within a constant factor of optimal) for several combinatorial optimization problems: paging, weighted caching, the k-server problem, and weighted matching. We introduce variations on the standard model of competitive analysis for paging: allowing randomization, allowing resource-bounded lookahead, and loose competitiveness, in which performance over a range of fast memory sizes is considered and noncompetitiveness is allowed provided the fault rate is insignificant. Each variation leads to substantially better competitive ratios. We present a general technique for

competitive analysis of linear optimization problems: competitive analyses are obtained by using linear programming duality to obtain bounds on the optimal cost. The technique is implicit in previous work on paging, weighted caching, and weighted matching. We generalize the implicit previous use of the technique, obtaining the greedy dual algorithm for weighted caching. The strategy generalizes the least-recently-used and first-in-first-out algorithms for paging and the balance algorithm for weighted caching. The analysis strengthens a previous analysis of the balance algorithm for weighted caching. We explore the linear programming dual of the k-server problem, showing that the k-server problem is a special case of on-line minimum-weight matching, revealing close relationships between on-line weighted caching and assignment algorithms, and showing how duality can yield potential function analyses. (69 pages, October 1991)

TR-349-91 Distributed EZ
Alvaro E. Campos and David R. Hanson
EZ is a system that integrates traditional operating systems and programming languages into a very high-level, persistent, string processing language. This paper describes the design and initial implementation of a distributed memory manager that distributes *EZ*'s virtual address space transparently among a network of homogeneous computers. The design adapts the techniques used in recent implementations of shared virtual memory for use in *EZ*'s persistent environment. Unlike most implementations of shared virtual memory, control information is distributed and migrates. This memory manager works in concert with a distributed mark-and-sweep garbage collector, which is also concurrent and real-time. This collector trades time for space and minimal disruption of mutators, which reduces communication costs. (14 pages, September 1991)

TR-350-91 Ray Shooting in Polygons Using Geodesic Triangulations
Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas Guibas, John Hershberger, Micha Sharir and Jack Snoeyink
Let $\mathcal{P}$ be a simple polygon with n vertices. We present a simple decomposition scheme that partitions the interior of $\mathcal{P}$ into $O(n)$ so-called geodesic triangles, so that any line segment interior to $\mathcal{P}$ crosses at most $2\log n$ of these triangles. This decomposition can be used to preprocess $\mathcal{P}$ in a very simple manner, so that any ray-shooting query can be answered in time $O(\log n)$. The data structure required $O(n)$ storage and $O(n \log n)$ preprocessing time. By using more sophisticated techniques, we can reduce the preprocessing time to $O(n)$. We also extend our general technique to the case of ray-shooting amidst k polygonal obstacles with a total of n edges, so that a query can be answered in $O(\sqrt{k}\log n)$ time. (15 pages, September 1991)

TR-351-91 Literate Programming Tools Need Not Be Complex
Norman Ramsey
When it was introduced, literate programming meant WEB. Desire to use WEB with languages other than Pascal led to the implementation of many versions. WEB is complex, and the difficulty of using WEB creates an artificial barrier to experimentation with literate programming. noweb provides much of the functionality of WEB, with a fraction of the complexity. noweb is independent of the target programming language, and its formatter-dependent part is a 40-line shell script. noweb is extensible, because it uses two representations of programs: one easily manipulated by authors and one easily manipulated by tools. I give examples of the use of noweb and I mention applications that have been written using noweb with different programming languages. (23 pages, October 1991)

TR-352-91 Debuggable Concurrency Extensions for Standard ML
Andrew P. Tolmach and Andrew W. Appel
We are developing an interactive debugger with reverse execution for the language Standard ML extended to include concurrent threads in the style of Modula-2+. Our debugging approach is based on automatic instrumentation in the source language of the user's source code; this makes the debugger completely independent of the compiler back-end, run-time system, and target hardware. The debugger operates entirely inside the concurrency model and has no special concurrency privileges. In this paper, we consider some of the challenges of debugging a non-deterministic concurrent symbolic language "in itself." Issues considered include logging non-deterministic activity, obtaining more secure semantics for our concurrency primitives, controlling distributed computations, and defining suitable time models. We conclude by suggesting an alternative simulation-based approach to dealing with non-determinism. (12 pages, October 1991)

TR-353-91 Dynamic Hierarchical Caching in Large-Scale Distributed File Systems
Matt Blaze and Rafael Alonso
Most Distributed File Systems (DFSs) are based on a flat client-server model in which each client interacts directly with the file server for all file operations. While this model works well for relatively small systems in which the file server has adequate capacity for all its clients, it does not scale to large numbers of clients or systems in which the clients are connected to the server through low-bandwidth links. Server traffic can be reduced substantially if clients keep even a modest-sized cache of previously read files. Intuitively, the benefits of caching can be increased by organizing clients into a hierarchy, in which only a small number of machines communicate directly with the file server, providing intermediate caching services to machines below them in the hierarchy. While this potentially reduces server traffic for widely shared files, it can introduce a significant delay for clients low in the hierarchy for access to files with a low degree of sharing. This paper describes a simple method for constructing dynamic hierarchies on a file-by-file basis. The results of a trace-driven simulation of a dynamic hierarchical filesystem are presented, yielding a reduction in server traffic of a factor of more than two for shared files compared with a flat scheme and without a large increase in client access time. An algorithm to maintain cache consistency with low overhead by detecting missed cache invalidation messages is given. (9 pages, October 1991)

TR-354-91 Processing of Read-Only Queries at a Remote Backup
Christos A. Polyzois and Hector Garcia-Molina
Remote backup systems are often used to provide high data availability. Updates are typically propagated to the backup via a log, which decouples the backup from the primary. We show that this decoupling can lead to efficient installation of updates in batches and efficient processing of read-only queries, by eliminating or reducing access conflicts between updates and queries. We present several methods for query processing at the backup and evaluate their performance analytically. (28 pages, November 1991)

TR-355-91 NFS Tracing by Passive Network Monitoring
Matt Blaze
Traces of filesystem activity have proven to be useful for a wide variety of purposes, ranging from quantitative analysis of system behavior to trace-driven simulation of filesystem algorithms. Such traces can be difficult to obtain, however, usually entailing modification of the filesystems to be monitored and runtime overhead for the period of the trace. Largely because of these difficulties, a surprisingly small number of filesystem traces have been conducted, and few sample workloads are available to filesystem researchers. This paper describes a portable toolkit for deriving approximate traces of NFS activity by non-intrusively monitoring the Ethernet traffic to and from the file server. The toolkit uses a promiscuous Ethernet listener interface to read and reconstruct NFS-related RPC packets intended for the server. It produces traces of the NFS activity as well as a plausible set of corresponding client system calls. The tool is currently in use at Princeton and other sites, and is available via anonymous ftp. (11 pages, November 1991)

TR-356-91 Computing Minimal Spanning Subgraphs in Linear Time
Xiafeng Han, Pierre Kelsen, Vijaya Ramachandran and Robert Tarjan
Let P be a property of undirected graphs. We consider the following problem: given a graph G that has property P, find a minimal spanning subgraph of G with property P. We describe two related algorithms for this problem and prove their correctness under some rather weak assumptions about P. We devise a general technique for analyzing the worst-case behavior of these algorithms. By applying the technique to 2-edge-connectivity and biconnectivity, we obtain a tight lower bound of $\Omega(m + n \log n)$ on the worst-case sequential running time of the above algorithms for these properties; this resolves open questions posed earlier with regard to these properties. We then give refinements of the basic algorithms that yield the first linear-time algorithms for finding a minimal 2-edge-connected spanning subgraph and a minimal biconnected spanning subgraph of a graph. (31 pages, December 1991)

TR-357-91 On the Bit Extraction Problem
Joel Friedman
Consider a coloring of the n-dimensional Boolean cube with $c = 2^s$ colors in such a way that every k-dimensional subcube is equicolored, i.e. each color occurs the same number of times. We show that for such a coloring we necessarily have $(k-1)/n \geq \theta_c = (c/2-1)/(c-1)$. This resolves the "bit extraction" or t-resilient functions problem in many cases, such as $c - 1|n$, proving that XOR type colorings are

optimal. We also study the problem of finding almost equicolored colorings when $(k-1)/n < \theta_c$, and of classifying all optimal colorings. (15 pages, December 1991)

TR-358-91 Derandomizing an Output-Sensitive Convex Hull Algorithm in Three Dimensions
Bernard Chazelle and Jiri Matousek

We consider the computation of the convex hull of a given n-point set in three-dimensional Euclidean space in an output-sensitive manner. Clarkson and Shor proposed an optimal randomized algorithm for this problem, with an expected running time of $O(n \log h)$, where h denotes the number of points on the surface of the convex hull. In this note we point out that the algorithm can be made deterministic by using recently developed techniques, thus obtaining an optimal deterministic algorithm. (5 pages, December 1991)

Copies of some of these reports can be purchased by contacting

Technical Reports
Department of Computer Science
Princeton University
35 Olden Steet
Princeton, New Jersey 08544-2087

Some reports are available in PostScript form via anonymous `ftp` from `ftp.cs.Princeton.EDU` (InterNet address 128.112.152.13) in the directory `reports`. The file `README` therein gives details, and the file `INDEX` lists the numbers and titles of all reports. Fetching `README` and `INDEX` is accomplished by the following commands. As suggested, use your email address as the password.

```
% ftp ftp.cs.Princeton.EDU
anonymous
your email address
cd reports
get README
get INDEX
quit
```