Second Progress Report

1993 – 1995

February 1995

INFORMATIK

Max–Planck–Institut für Informatik

Im Stadtwald
66123 Saarbrücken
Germany

Contents

Part I

Overview

Research Programme

The institute is devoted to basic research in computer science, and in particular to the study of complex computer systems.

Complexity in computer systems arises for various reasons: A problem can be complex due to huge masses of data that have to be handled, sometimes in real time. For this sort of problem efficient algorithms and data structures as well as parallel processing are of great importance. Parallel algorithms are often designed for theoretical machines which abstract from the actual communication between their processors in one way or another. The problem of how we should actually build such machines is still unsolved.

Or complexity can mean logical complexity as we find it in large software systems, with many layers of abstraction, where applications from different problem domains interact with each other in often unpredictable ways. Here we need to apply methods based on mathematical logic in order to structure, reason about, and develop more systematically, such large systems.

Today's computer systems are very complex in the way that they consist of a large number of hardware components which operate concurrently and which are physically distributed, often in a non-local manner. We want to understand better the nature of such systems, how to develop them such that they behave predictably and as wanted and to ensure that they are, by the way they are constructed, insensitive to faults in their components.

Computer systems are more and more used to realize and simulate some part of the real or of an imaginary world. Such simulations have to deal with all of the forms of complexity we have mentioned above.

Structure

The institute is planned to eventually consist of five research groups, to fit this research programme, in the following areas:

1. Algorithms and Complexity
2. Programming Logics
3. Concurrent and Distributed Systems
4. Computer Architecture
5. Simulation and Virtual Reality

Since it opened in Dec. 1990 two research units have been installed, the Algorithms and Complexity group headed by K. Mehlhorn, and the Programming Logics group headed by H. Ganzinger. In addition to these two, a research group at the University of Potsdam, headed by M. Gössel, is associated with the institute. This group investigates fault tolerant computing.

At present 16 research associates, 26 doctorate students and 17 postdocs are affiliated with the institute. The scientific staff is complemented by an administrative unit (11 members, including secretaries), by a computing support unit (5 members of staff) and by our library (2 members of staff). The computing support unit currently operates a network of approximately 120 workstations.

Grants

The institute is involved in a number of projects related to research grants awarded by the European Union, by the German Science Foundation (DFG) and by the German Ministery of Research and Technology (BMFT); for the descriptions of these grants see sections II.9 and III.7.

Funding of these projects in 1994 was about 2,2 Mio. DM.

Results

In the two parts to follow we describe in detail, for the two research groups, the research programmes and results obtained in the period December 1993 through February 1995. We have continued to be very successful in our research, as documented by our many scientific publications, including about 130 articles in journals, books or proceedings of major international conferences.

Many of the institute's results are, in addition, available to the public through computer programs such as the LEDA library of efficient algorithms, the ACID collection for term indexing data structures, the SCAN system for second-order quantifier elimination and circumscription, and the SATURATE experimental theorem prover.

Teaching Activities

The institute makes an effort to offering a variety of courses to computer science students of the Universität des Saarlandes. Courses taught during the period of this report are listed in Sections II.6 and III.5. In the same period 3 doctoral dissertations and 4 habilitations have been successfully completed.

Professional Activities

Members of the institute have been involved in the organization of 11 workshops and conferences. In 17 cases we have been invited to join the program committee of major international conferences, not counting program committee memberships for national and international workshops. Finally, we serve on the editorial board of 11 scientific journals.

Professorships

The following members of the institute have been offered (tenured) professorships in 1993/94 which they have accepted.
Michael Hanus (Programming Logics Group): C3, RWTH Aachen.
Stefan Näher (Algorithms and Complexity Group): C3, U. Halle.
Rolf Socher (Programming Logics Group): C2, FH Emden.

Awards

Susanne Albers, member of the Algorithms and Complexity Group, has been awarded the Otto Hahn Medal of the Max-Planck-Gesellschaft in recognition of her excellent doctoral thesis about lookahead in on-line algorithms.
Kurt Mehlhorn has been awarded the Karl Heinz Beckurts Prize for his distinguished scientific contributions to the area of efficient algorithms and especially for his effort in making the results of his field accessible for practical exploitation in academia as well as in industry.

Part II

The Algorithms and Complexity Group

1 Personnel

Director:
Prof. Dr. Kurt Mehlhorn

Researchers:
Dr. Susanne Albers (until September 1994; now at ICSI, Berkeley, USA)
Dr. Srinivasa Arikati
Dr. Sunil Arya
Dr. Greg Barnes (until August 1994; now at University of Waterloo, Canada)
Dr. Phil Bradford (since September 1994; previously at Indiana University, USA)
Dr. Vasilis Capoyleas (since October 1994; previously at Purdue University, USA)
Dr. Shiva Chaudhuri
Dr. Devdatt Dubhashi (until August 1994; now at BRICS, Univ. of Aarhus, Denmark)
Dr. Rudolf Fleischer
Dr. Naveen Garg (since September 1994; previously at IIT, Delhi, India)
Privatdozent Dr. Torben Hagerup
Dr. Ramesh Hariharan (since September 1994; previously at Courant Institute, NYU, USA)
Dr. Pierre Kelsen (since October 1994; previously at Univ. of British Columbia, Canada)
Dr. Hans-Peter Lenhof
Dr. Sanjeev Mahajan (since September 1994; previously at Simon Fraser Univ., Canada)
Dr. Anil Maheshwari (until August 1994; now at TIFR, Bombay, India)
Prof. Dr. Kurt Mehlhorn
Dr. Michael Müller (until May 1994; now at RIB Bausoftware GmbH, Stuttgart, Germany)
Dr. Petra Mutzel (since November 1994; previously at Köln Universität, Germany)
Privatdozent Dr. Stefan Näher (until September 1994; now at Halle Universität, Germany)
Dr. Christine Rüb
Dr. Sanjeev Saluja (since October 1994; previously at TIFR, Bombay, India)
Dr. Stefan Schirra
Dr. Christian Schwarz (since September 1994; previously at ICSI, Berkeley, USA)
Dr. Jop Sibeyn
Dr. Michiel Smid
Dr. Phillipas Tsigas (since January 1995; previously at CTI, Univ. of Patras, Greece)
Dr. Christian Uhrig
Dr. Christos Zaroliagis

Ph.D. students:
Hannah Bast (since December 1994)
Christoph Burnikel
Jordan Gergov
Gerhard Klär (until January 1994; now at Techniker Krankenkasse, Hamburg, Germany)
Thomas Lauer
Markus Paul
Volker Priebe
Ronald Rasch (until August 1994; now at Techn. Univ. Bergakademie Freiberg, Germany)
Knut Reinert
Thomas Schilz
Erik Schwarzenecker

Christian Thiel

Secretaries:
Andrea Eßer
Ingrid Finkler
Martina Horn

2 Executive Summary

The goal of the research unit "Algorithms and Complexity" is to understand the computational complexity of algorithmic problems and to develop efficient algorithms and data structures for their solution. Our work spans from theory to practice. We have managed, quite successfully, to intertwine theory with practice; in fact, frequently the same researchers are involved in both. Our theoretical results appear in numerous publications in reputed conferences and journals and the practical work has resulted in useful, commercial-quality software. Besides we are also actively involved in teaching.

As in previous years, our research concentrates on computational geometry, parallel algorithms, data structures and graph algorithms, computational complexity, and implementation of algorithms. There is a slight shift in emphasis, though. We have increased our strength in combinatorial optimization and we have picked up more applied themes. We now briefly survey our results in each of the five areas and refer the reader to Section 3 for more details.

Parallel and Distributed Algorithms (coordinators Torben Hagerup and Christine Rüb): We have developed algorithms for a variety of models of computation: PRAMs, fixed interconnection networks like hypercubes and meshes, and synchronous and asynchronous distributed machines. Problems under consideration are sorting, merging, routing, graph algorithms, load balancing, and a framework for the lock-free implementation of data-structures. Three specific results are:

Christine Rüb has studied the average-complexity of odd-even merge sort. She showed that its average running time on many networks is $O((n \log n)/p)$ provided that the size n of the input is at least the square of the number p of processors.

Job Sibeyn has continued to investigate sorting and routing algorithms on meshes. He has developed a simple deterministic algorithm that routes any permutation in time $2n + o(n)$ on an $n \times n$ mesh. This is optimal up to low-order terms.

On the applied side Christine Rüb and Hans-Peter Lenhof are working on the molecular dynamics simulation of synthetic polymers. They are using a CM5 (at GMD in Bonn) and an iPSC/860 (at Universität des Saarlandes) for their experiments.

Computational Geometry (coordinators Kurt Mehlhorn and Michiel Smid): We continued and extended our work on proximity problems: Several algorithms that result from our previous work were implemented using LEDA. Also, we started investigating new topics in this area, such as approximate nearest neighbor searching and the construction of spanners. We completed our work on unifying abstract furthest-site Voronoi diagrams. We worked extensively on many types of intersection searching problems, such as approximate range searching, generalized intersection searching, rectangle enclosure problems, intersection problems on moving objects, and collision detection problems for moving polyhedra. We started working on geometric optimization problems in low-dimensional space (such as computing the width or roundness of a point set and computing a largest empty anchored cylinder), and on geometric constraints, a research area with many applications in CAD. Two specific results are:

In a sequence of papers, Sunil Arya, David Mount, and Michiel Smid (with varying co-authors) have obtained much improved Euclidean spanners. A spanner is a graph defined on n points in the plane that approximates the complete graph in the sense that any two points are connected by a path whose length exceeds the Euclidean distance of the points by only a constant factor. Their construction gives spanners of bounded degree, low weight, and small diameter.

Elmar Schömer (from the Universität des Saarlandes) and Christian Thiel obtained the first subquadratic algorithms to test whether two polyhedra collide under translational or rotational movement.

Data Structures and Combinatorial Algorithms (coordinator Kurt Mehlhorn): Our work in this area spans a wide range, from probabilistic analysis of algorithms to on-line algorithms, and from algorithms for sparse graphs and networks to pattern matching. Besides, our evolving group in combinatorial optimization is involved in research on approximation algorithms for NP-complete problems and on the more applied side in graph drawing and the computation of ground states for Ising spin glasses. Two specific results are:

Susanne Albers exhibited a randomized algorithm for the list-update problem that is ϕ-competitive, where ϕ is the golden ratio.

Christos Zaroliagis (with co-authors from Patras University, his previous affiliation) explores the use of hammock-decompositions for obtaining fast sequential and parallel graph and network algorithms, e.g. in the all-pairs shortest paths problem. Let $\tilde{\gamma}$ be the minimum number of outerplanar graphs into which a graph decomposes. Then $1 \leq \tilde{\gamma} \leq n$. The running time of the new algorithms matches the previously best bounds when $\tilde{\gamma} = n$. For small $\tilde{\gamma}$, it is significantly smaller.

LEDA (coordinators Kurt Mehlhorn and Stefan Näher): We have further developed the LEDA platform for combinatorial and geometric computing. It is used by several hundred academic and industrial groups world-wide as the basis of their algorithms development. In January 1995 we have released version 3.1. Its most exciting new features are correct implementations of some basic geometric algorithms; they handle all degeneracies and avoid the dangers of inexact arithmetic. A significant part of LEDA is now so mature that several companies have made it the basis of serious software development. The spin-off company LEDA GmbH plans to maintain, distribute and further develop this part of LEDA.

Complexity Theory: This is a small effort compared to the other four. Nevertheless, we have worked on fundamental problems; namely, on the complexity of circuits, of directed $s-t$ connectivity, of linear logic, and of decision trees. A specific new result is:

Shiva Chaudhuri was able to show that the set of functions computable by uniform polynomial size circuits of constant depth properly includes those computable by uniform linear size circuits of constant depth.

Stefan Näher completed his Habilitation in 1994, and he is now associate professor of Computer Science at the University of Halle. The Habilitation of Michiel Smid has just been completed (February 1995). Two graduate students (G. Klär and R. Rasch) completed their Ph.D. work, ten Ph.D students are currently working in the group. Section 7 surveys their topics and expected completion dates. Most of the researchers in the group are on two-year post-doc contracts or three-year graduate student scholarships and hence there is considerable fluctuation. We run an intensive seminar and lecture program to spur interaction within the group. We have two 45-minute "noon seminars" per week, which every group member has to attend, and two 90-minute lectures per week. The lectures are reserved for two to four week intensive treatments of "hot" topics, see Section 8 for more details. We run an intensive visitor program; more than 30 guests visited our group for stays up to three months, cf. Section 4 for details.

The group contributes to the master's program in Computer Science at the Universität des Saarlandes. Ten master students have finished their master's thesis under our supervision in the last 12 months. We have offered thirteen courses and seminars in the last year, cf. Section 6 for details. The group is involved in five national and international research projects. Section 9 gives details.

3 Research Themes

This section describes the work (theoretical and practical) done by our group. It is divided into five subsections, one per each working area. To be easily distinguishable, references regarding work done by members of the group (during the period considered in this report) are marked with a • .

3.1 Parallel and Distributed Computing

3.1.1 Parallel Merge Sorting

Investigator: Christine Rüb

Sorting is an important and ubiquitous problem in parallel computation. This section discusses two results concerning parallel sorting algorithms. The first one shows that Batcher's odd-even merge sort can be implemented to run optimally, for random large inputs, on many existing parallel machines, and the second one shows that sorting by merging can, on the hypercube model, be optimal (in the worst as well as in the average case) only when the size of the input is relatively large compared with the number of processors available.

Batcher's odd-even merge sort is a well-known parallel algorithm for sorting. Originally this algorithm was described as a comparator network of size $O(n \log^2 n)$ that sorts n elements in time $O(\log^2 n)$ [1]. However, it can also be used as a sorting procedure on a parallel computer. In this case there will, in general, be fewer processors than input elements. Odd-even merge sort can then be implemented by substituting all comparisons (and exchanges) between two elements by splits of two sorted lists at the median of their union.

In practice, when comparing different sorting algorithms, odd-even merge sort has not been used often. Instead, another sorting algorithm proposed by Batcher [1], namely bitonic sorting, has been used in the comparisons. Because of its small constant factors, bitonic sort is quite efficient on many parallel machines. One reason that has been mentioned for this preference is that odd-even merge sort is not as "composable" [2] as bitonic sort.

In [4] we show, however, that odd-even merge sort can be implemented (by keeping the communication between processors to a minimum) so that it performs on the average much better than bitonic sort. More precisely, we show that the average running time of odd-even merge sort can be $O((n/p)(\log n + \log p \log(1 + p^2/n)))$ (with small constant factors), whereas the average running time of bitonic sort is $\Theta((n/p)(\log n + \log^2 p))$. ($n$ is the size of the input and p is the number of processors used. The average is taken over all possible ways to store the input elements evenly distributed among the processors.) In particular, odd-even merge sort needs on the average much less communication than bitonic sort; this is important since communication is still relatively expensive on existing parallel machines.

Because of its small constant factors, this implementation of odd-even merge sort yields, for random inputs, good speedups on many existing parallel machines. Its main advantages are that it is comparison-based, i.e., can sort arbitrary elements, that the elements are distributed evenly among the processors at the end, and that it is adaptive. The latter means that it will work even faster for presorted inputs. Also, there are no problems when the input contains many identical keys like in Sample Sort [2], one of the fastest known parallel sorting algorithms.

The hypercube interconnection model for a parallel computer is defined as follows. We are given p (p a power of two) processors $P_0, ..., P_{p-1}$ that are connected in the form of a $\log p$-dimensional hypercube. I.e., each processor P_i is connected to all processors P_j such that the binary representation of i and j differ in exactly one position. The development of a deterministic algorithm for sorting on the hypercube that is work-optimal and runs in polylogarithmic time is a long-standing open problem and has attracted considerable interest. There are, however, several work-optimal deterministic algorithms for sorting on the PRAM-model that run in polylogarithmic time. All of these algorithms have in common that they sort by repeatedly merging pairs of sorted sequences. Thus a question to ask is: is the same possible for the hypercube, i.e., is it possible to merge two sorted lists of altogether n elements using $O(n)$ work and polylog(n) time? In [3] we show that this is not the case. Specifically, we show the following. Let n be the size of the input and let p be the number of processors available. Then sorting by repeatedly merging pairs of lists of elements can, for the average case, not be optimal if $p \geq n^{0.5+\epsilon}$ for any $\epsilon > 0$.

References

[1] K.E. Batcher. Sorting networks and their applications. *Proceedings, AFIPS Spring Joint Computer Conference*, pages 307–314, 1968.

[2] G.E. Blelloch, L. Dagum, S.J. Smith, K. Thearling, and M. Zagha. An evaluation of sorting as a supercomputer benchmark. Technical Report RNR-93-002, NAS Applied Research Branch, 1993.

• [3] Ch. Rüb. Lower bounds for merging on the hypercube. *Proc. 2nd Italian Conference on Algorithms and Complexity, LNCS 778*, pages 213–222, 1994.

• [4] Ch. Rüb. On the average running time of odd-even merge sort. *Proc. 12th Symposium on Theoretical Aspects of Computer Science, STACS '95*, 1995. To appear.

3.1.2 Parallel Graph Algorithms

Investigators: Srinivasa Arikati, Shiva Chaudhuri, Torben Hagerup, Anil Maheshwari and Christos Zaroliagis

Recently much of our research into parallel algorithms has been focussed on parallel graph algorithms. We understand the term in a wide sense, including parallel algorithms for computing various quantities defined by input graphs and networks (this covers the bulk of our work) as well as parallel algorithms for unrelated problems that use graphs internally as part of their computation.

Graph Decompositions

Graph decompositions have played an important role in sequential computation. Their role in parallel computation is perhaps even more prominant, since quite often the efficient parallel solution of many problems requires the invention and use of original, novel approaches radically different from those used to solve the same problems sequentially. In other cases, the novel parallel solution paradigm stems from a non-trivial parallelization of a specific sequential method (e.g. merge-sort for EREW PRAM optimal sorting).

Motivated by the second paradigm, we have given an efficient parallel algorithm for decomposing an n-vertex, m-edge graph G into a number, $\tilde{\gamma}$, of outerplanar subgraphs (called *hammocks*) satisfying certain separator properties [22]. Our work is based on the sequential hammock decomposition technique introduced by Frederickson [15] and the parallel ear decomposition technique [23], thus we call it the *hammock-on-ears decomposition*. The number of hammocks produced is closely related to certain embedding properties of G (e.g. genus) and varies from 1 up to $\Theta(m)$ [15]. Roughly speaking, $\tilde{\gamma}$ is a measure of the topological complexity of G. The better the topological characteristics of G are, the smaller the value of $\tilde{\gamma}$ becomes. (For example, if G is outerplanar, then $\tilde{\gamma} = 1$.) Our algorithm runs in $O(\log n \log\log n)$

(resp. $O(\log n)$) time using $O(n+m)$ CREW (resp. CRCW) PRAM processors [22]. The hammock-on-ears decomposition implies a general framework for solving graph problems efficiently in parallel, especially when the input graph is sparse, i.e. $m = O(n)$. The main idea of the technique is to partially reduce the solution of a given problem Π on G to the solution of Π on subgraphs with a particular nice structure (hammocks) and then combine the partial solutions. The value of the technique is demonstrated by improving previous bounds for certain instances of shortest paths and related problems on sparse graphs. The details of these results are given in the rest of this subsection.

We plan to continue work on graph decompositions. We would like to find more efficient solutions to graph problems, by decomposing the input graph into subgraphs with a known structure, solving these subproblems and combining the solutions.

Shortest Paths

The shortest path problem is maybe the most fundamental problem in network optimization. Its theoretical and practical importance has been widely recognized and an extensive list of applications can be found in the recent book by Ahuja, Magnanti and Orlin [1]. The problem is to find paths of minimum weight between vertices in an n-vertex, m-edge directed graph (digraph) G with real edge weights. There are two main versions of the problem: the *single-source or shortest path tree* problem, which asks for shortest paths from a specific vertex to all other vertices in G, and the *all-pairs shortest paths* (apsp) problem, which asks for such paths between every pair of vertices in G. Note that an apsp algorithm which outputs apsp information in the standard form (i.e. either in a table, or as n shortest path trees), requires $\Omega(n^2)$ time and space. Both versions of the problem have been intensively studied, especially in sequential computation.

The best results for the apsp case are due to Frederickson [15, 16] and are based on the sequential hammock decomposition (mentioned previously in the "Graph Decompositions") and on a special encoding of apsp information which uses a local table (called a compact routing table) for apsp inside a hammock and a global table for apsp between hammocks. This encoding avoids the $\Omega(n^2)$ lower bound needed for the time and space if the output has to be in the standard form. In [15], an $O(n + \tilde{\gamma}^2 \log \tilde{\gamma})$ time algorithm for the apsp problem in a sparse digraph is given. If the input digraph is planar, then the above bound reduces to $O(n + \tilde{\gamma}^2)$ [16]. In both cases, the space used is $O(n + \tilde{\gamma}^2)$.

In parallel computation, the best previous result for apsp is due to Han, Pan and Reif [18], and requires $O(n^3)$ work, even for the case where G is a non-planar sparse digraph. The result is based on the matrix powering method. If G is planar, then the best previous result was given by Cohen [12] and runs in $O(\log^5 n)$ time using $O(n^2)$ work on a CREW PRAM.

We have worked towards closing the gap between the time performed by the best sequential algorithm for the apsp problem and the work performed by a parallel algorithm. Based on the hammock-on-ears decomposition (described previously in the "Graph Decompositions"), as well as on other techniques, we have achieved the following results [22]. For the case of sparse digraphs, we have given a CREW PRAM algorithm for the apsp problem, which runs in $O(\log^2 n)$ time using $O(n \log^2 n + \tilde{\gamma}^3)$ work. If the input digraph is planar, then our algorithm runs in $O(\log^2 n + \log^5 \tilde{\gamma})$ time using $O(n \log^2 n + \tilde{\gamma}^2)$ work. The space used in both cases is $O(n + \tilde{\gamma}^2)$ and the encoding of apsp information is the same as in [15, 16]. Note that our algorithms perform very well when $\tilde{\gamma} = o(n)$ (i.e., in all cases where the graph has nice topological properties).

The shortest paths problem appears to have a special irregularity, when the input digraph G contains negative edge weights. In this case, a shortest path between two vertices v and u exists, iff no path from v to u contains a cycle of total negative weight. Thus, finding negative cycles in digraphs is fundamental for the shortest paths problem. In [21], we have given an algorithm for detecting, and outputting if it exists, a negative cycle in planar and sparse digraphs. The resource bounds of the algorithm are the same as those for the apsp problem described above. In the special case where G is outerplanar, we gave an $O(\log n \log^* n)$-time, $O(n)$-work CREW PRAM algorithm.

Another way of avoiding the $\Omega(n^2)$ lower bound for the apsp problem is to make a preprocessing of the input digraph G so that subsequent shortest path or distance *queries*, between any pair of vertices, can be efficiently answered. Moreover, it is of particular interest to be able to update the data structures created during the preprocessing, after the change of an edge weight, in appropriately short time (i.e., without recomputing everything from scratch). We refer to this problem as the *dynamic shortest paths* problem. We have recently given [14] a parallel CREW PRAM algorithm for the dynamic shortest paths problem in planar digraphs, which is also based on the hammock-on-ears decomposition. A distance query is answered in $O(\log n + \log^2 \tilde{\gamma})$ time using $O(\log n + \tilde{\gamma})$ work, after a preprocessing of the planar digraph in $O(\log n \log^* n + \log^5 \tilde{\gamma})$ time and $O(n \log n \log^* n + \tilde{\gamma}^{1.5})$ work. The space used is $O(n + \tilde{\gamma}^{1.5})$. If the shortest path is also required, then it can be output in the same time as the distance query and in additional work proportional to the number of the edges of the path. We can update the data structures set up by our preprocessing algorithm, after an edge weight modification, in $O(\log n + \log^3 \tilde{\gamma})$ time using $O(\log n + \tilde{\gamma}^{1.5})$ work. We are not aware of any previous parallel algorithm for the problem. If the input digraph is outerplanar, then the preprocessing can be done in $O(\log n)$ time and $O(n \log n)$ work. The bounds for the query and the update operations can be derived from the above, by setting $\tilde{\gamma} = 1$.

Another important application of our parallel graph decomposition result, concerns the following problem, known as the *quickest path* problem. Assume that we are given a network N represented as a digraph, where the vertices represent transmitters/receivers without data memories and the edges represent communication channels. Each edge e has a capacity $c(e)$ and a lead (or delay) time $l(e)$ associated with it. The transmission time to send σ units of data from a given source s to a destination t using path p is $T(\sigma, p) = l(p) + \frac{\sigma}{c(p)}$, where $l(p)$ is the sum of the lead times of the edges in p, and $c(p)$ is the minimum capacity of the edges in p. The quickest path problem is to find a path of minimum transmission time to transmit the σ units of data from s to t. Note that this problem is quite different from the shortest path problem, since a subpath of a quickest path is not necessarily a quickest path itself. We have recently given the first efficient parallel and dynamic algorithms for the quickest paths problem on sparse networks [19]. The results are partially based on our previously mentioned algorithms for the apsp and the dynamic shortest paths problem.

We plan to work more in the above directions. Work in progress aims at finding faster and more efficient algorithms for the dynamic shortest paths problem on sparse digraphs.

Parallel Network Flow

Important classes of graphs and networks are those of *bounded tree-width*. While the precise definition of the tree-width of a graph is technical and omitted here, the important intuition is that graphs of small tree-width resemble trees in many ways (although not necessarily from a superficial inspection). Many important computational problems, although hard for general input graphs, are simple for input graphs that are trees. The challenge therefore is to exploit the tree-like structure of graphs of small tree-width in a computational context in order to obtain algorithms for such graphs that are (almost) as efficient as the corresponding algorithms for trees. We have shown how to do this for the fundamental *maximum-flow* problem.

The best algorithms for computing maximum flows in general networks with n vertices and m edges have running times around $\Theta(nm)$. For sparse networks, thus, the running time is quadratic in the input size. In contrast, we showed in [17] that maximum flows in n-vertex networks of bounded treewidth can be computed in $O(n)$ sequential time and in $O(\log n)$ parallel time on an $O(n/\log n)$-processor EREW PRAM. The latter result holds only if structural information about the input graph in the form of a so-called *tree decomposition* is provided as part of the input. Ongoing work aims at removing this condition by showing how to compute a tree decomposition of the input graph (almost) as efficiently as we can solve the maximum-flow problem.

Prefix Graphs

The *range product problem* is, for a given set S equipped with an associative operator $\circ$, to preprocess a sequence $a_1, \ldots, a_n$ of elements from S so as to enable efficient subsequent processing of queries of the form: Given a pair (s,t) of integers with $1 \leq s \leq t \leq n$, return $a_s \circ a_{s+1} \circ \cdots \circ a_t$. The generic range-product problem and special cases thereof, usually with $\circ$ computing the maximum of its arguments according to some linear order on S, have been extensively studied, often without the problem under investigation being recognized as a range-product problem. We show in [11] that a large number of previous sequential and parallel algorithms for such problems can be unified and simplified by means of *prefix graphs*, graphs that are closely related to the fundamental Ackermann's function. In more detail, our method applies to sequential range queries [2], addition on unbounded-fanin circuits [10], parallel segmented broadcasting [9, 25], parallel linear-range merging [8], parallel range maxima of c-bounded sequences [9], and parallel randomized range maxima [7]. While no new results were obtained, the unified approach allowed us, in 13 pages, to reprove results whose original derivation spans nearly 100 journal pages.

Implicit Representation of Graphs

A fundamental data structuring question in the design of efficient algorithms is how to represent a graph in memory using as little space as possible, so that given any two vertices we can test if they are adjacent in $O(1)$ time [20, 27]. The well-known adjacency matrix representation permits adjacency queries in $O(1)$ time, but it requires $\Theta(n^2)$ space, even in the case where the input graph is sparse (i.e. it has a linear number of edges). Another characterization of sparse graphs is given by the *arboricity*. The arboricity of a graph G is defined as $\max_H\{m_H/(n_H-1)\}$, where H is an n_H-vertex, m_H-edge subgraph of G. (For example, planar graphs have arboricity 3.) Now sparse graphs are the graphs of bounded arboricity. It follows by a theorem of Nash-Williams [20] that a graph with arboricity c can be implicitly represented in memory using only $(c+1)n$ space. In such a case, G is said to have an *optimal implicit representation*. The known parallel algorithms for finding an optimal implicit representation of a sparse graph G were based on involved techniques for matroid union and intersection [24]. Moreover, they run in $O(\log^3 n)$ time using $O(n^{4.5})$ processors on a randomized CREW PRAM. The only deterministic parallel algorithm known, was for the case of planar graphs and runs in $O(\log n \log\log n)$ time using $O(n)$ work on a CRCW PRAM.

In [3], we have presented a very simple and optimal parallel algorithm, which runs in $O(\log n)$ time and $O(n)$ work on a CRCW PRAM, for computing an optimal implicit representation of a sparse graph. Moreover, since computing the exact value of the arboricity seems to be hard [24, 27], we have given an efficient deterministic parallel algorithm for computing a

2-approximate value for arboricity. Surprisingly enough, we have also shown [3] that using this approximate value, we can still obtain an optimal implicit representation of a sparse graph.

We plan to continue work in this direction. More precisely, we would like to investigate the case where the input graph may change dynamically.

Degree Sequence Problems

An important problem in graph algorithms is to compute a (simple) graph satisfying given degree constraints. An integer sequence d is called a *degree sequence* if there exists a graph G such that the degrees of its vertices are equal to the components of the sequence d. The graph G is said to be a *realization* of the sequence d.

Given an integer sequence d, there are two problems of interest: the *decision problem* is to test if d is realizable; the *search problem* is to compute a realization of d.

In the context of parallel computation the distinction between search problems and decision problems is far more important than in sequential computation because we are interested in algorithms that run in sublinear time. And in fact there are many cases where a decision problem is easy or even trivial to solve in parallel, but the corresponding search problem is challenging (see [26] for details). Degree sequence problems are one such important case. The decision problem can be solved by verifying certain linear inequalities (see e.g. [6]) and the verification can be done easily and efficiently in parallel. The parallel complexity of the search problem, on the other hand, has been open so far.

There has been recent interest in parallel algorithms for the search problem. It is shown in [13] that a special case of this problem is in NC. We completely solved the search problem by presenting an efficient deterministic parallel algorithm to compute a realization G of a given sequence d [5]. Ours is the first NC algorithm for this problem and the algorithm runs in $O(\log n)$ time using $O(n+m)$ CRCW PRAM processors, where n and m denote the number of vertices and edges in G.

We continued our study of degree sequence problems by considering connectivity requirements. A graph G is k-edge-connected (resp. k-vertex-connected) if G can not be disconnected by deleting k or fewer edges (resp. vertices). Given d and an integer k, we studied the following problems. (i) Compute a k-edge-connected realization of d. (ii) Compute a k-vertex-connected realization of d. We presented the first parallel algorithms for these problems [4]. Our results are as follows, where n and m denote the number of vertices and edges in the realization: (a) a randomized parallel algorithm for problem (i) that runs in $O(k \text{ polylog}(n))$ time using $O(n^{4.5}m)$ CRCW PRAM processors, and a deterministic parallel algorithm for the same problem that runs in $O(k \text{ polylog}(n))$ time using a polynomial number of processors (the polynomial has a very high degree); (b) an efficient deterministic parallel algorithm for problem (ii) when $k = 2$; the algorithm runs in $O(\log n)$ time using $O(n+m)$ CRCW PRAM processors.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows.* Prentice-Hall, 1993.

[2] N. Alon and B. Schieber. Optimal preprocessing for answering on-line product queries. Technical Report 71/87, Tel-Aviv University, 1987.

• [3] S. Arikati, A. Maheshwari, and C. Zaroliagis. Saving bits made easy. In *Proc. 6th Canadian Conference on Computational Geometry (CCCG'94)*, pages 140–146, August 1994. Also Tech. Rep. MPI-I-94-148, 1994.

• [4] S.R. Arikati. On the parallel complexity of the degree sequence problems. Technical Report MPI-I-94-162, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1994.

• [5] S.R. Arikati and A. Maheshwari. Realizing degree sequences in parallel. In *Proc. 5th ISAAC'94, Lecture Notes in Comp. Sci. (LNCS) 834:261–269, Springer-Verlag*, 1994. To appear in *SIAM Journal Discrete Math.*

[6] C. Berge. *Graphs and Hypergraphs.* North-Holland, Amsterdam, 1973.

[7] O. Berkman, Y. Matias, and U. Vishkin. Randomized range-maxima in nearly-constant parallel time. *Comput. Complexity*, 2:350–373, 1992.

[8] O. Berkman and U. Vishkin. On parallel integer merging. *Inform. and Computation*, 106:266–285, 1993.

[9] O. Berkman and U. Vishkin. Recursive star-tree parallel data structure. *SIAM J. Comput.*, 22:221–242, 1993.

[10] A. K. Chandra, S. Fortune, and R. Lipton. Unbounded fan-in circuits and associative functions. *J. Comput. Syst. Sci.*, 30:222–234, 1985.

• [11] S. Chaudhuri and T. Hagerup. Prefix graphs and their applications. In *Proc. 20th International Workshop on Graph-Theoretic Concepts in Computer Science(WG 94)*, Springer Lecture Notes in Computer Science, to appear.

[12] E. Cohen. Efficient parallel shortest-paths in digraphs with a separator decomposition. In *Proc. 5th ACM Symp. on Parallel Algorithms and Architectures (SPAA'93)*, pages 57–67, July 1993.

[13] A. Dessmark, A. Lingas, and O. Garrido. On the parallel complexity of maximum f-matching and the degree sequence problem. In *Proc. MFCS'94, Lecture Notes in Comp. Sci. (LNCS) 841: 316–325, Springer-Verlag*, 1994.

• [14] H. Djidjev, G. Pantziou, and C. Zaroliagis. On-line and dynamic algorithms for shortest path problems. In *Proc. 12th Symp. on Theoretical Aspects of Computer Science (STACS'95), to appear.* LNCS, Springer-Verlag, March 1995. Also Tech. Rep. MPI-I-94-114, 1994.

[15] G. Frederickson. Using cellular graph embeddings in solving all pairs shortest path problems. In *Proc. 30th Annual IEEE Symp. on Foundations of Comp. Science (FOCS'89)*, pages 448–453, October 1989.

[16] G. Frederickson. Planar graph decomposition and all pairs shortest paths. *J. ACM*, 38(1):162–204, January 1991.

• [17] T. Hagerup, J. Katajainen, N. Nishimura, and P. Ragde. Characterizations of k-terminal flow networks and computing network flows in partial k-trees. In *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1995.

[18] Y. Han, V. Pan, and J. Reif. Efficient parallel algorithms for computing all pair shortest paths in directed graphs. In *Proc. 4th ACM Symp. on Parallel Algorithms and Architectures (SPAA' 92)*, pages 353–362, July 1992.

• [19] D. Kagaris, G. Pantziou, S. Tragoudas, and C. Zaroliagis. Quickest paths: Parallelization and dynamization. In *Proc. 28th Hawaii Int'l Conference on System Sciences (HICCS-28), to appear*, 1995.

[20] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. In *Proc. 20th ACM Symp. on Theory of Computing (STOC'88)*, pages 334–343, May 1988.

• [21] D. Kavvadias, G. Pantziou, P. Spirakis, and C. Zaroliagis. Efficient sequential and parallel algorithms for the negative cycle problem. In *Proc. 5th Int'l Symp. on Algorithms and Computation (ISAAC'94)*, pages 270–278. LNCS 834, Springer-Verlag, August 1994.

• [22] D. Kavvadias, G. Pantziou, P. Spirakis, and C. Zaroliagis. Hammock-on-ears decomposition: A technique for the efficient parallel solution of shortest paths and other problems. In *Proc. 19th Symp. on Mathematical Foundations of Comp. Science (MFCS'94)*, pages 462–472. LNCS 841, Springer-Verlag, August 1994. Also Tech. Rep. MPI-I-94-131, 1994.

[23] Y. Maon, B. Schieber, and U. Vishkin. Parallel ear decomposition search (EDS) and st-numbering in graphs. *Theoretical Computer Science*, 47:277–298, 1986.

[24] H. Narayanan, H. Saran, and V.V. Vazirani. Randomized parallel algorithms for matroid union and intersection, with applications to arborescences, and edge-disjoint spanning trees. In *Proc. 3rd ACM-SIAM Symposium on Discrete Algorithms (SODA'92)*, 1992.

[25] P. Ragde. The parallel simplicity of compaction and chaining. *J. Alg.*, 14:371–380, 1993.

[26] E. Upfal, R.M. Karp, and A. Wigderson. The complexity of parallel search. Technical report, IBM Research Report, RJ 5434 (55563), N.Y., 1986.

[27] J. van Leeuwen. Graph algorithms. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol.A*, chapter 10, pages 525–631. Elsevier, 1990.

3.1.3 Lock-Free Data Structures

Investigator: Greg Barnes

When a data structure is shared between several concurrently executing processes, the consistency of the data structure has traditionally been ensured through means of exclusive access. However, the relative speed of concurrently executing processes may vary dramatically over time for a variety of reasons, i.e., a process may experience arbitrarily long delays with respect to other processes, such that granting exclusive access to a process entails the risk of delaying all processes for an indefinite length of time — they cannot proceed until a single slow process relinquishes its lock on the shared data structure. Data structures that do not suffer from this problem are called *lock-free*.

Herlihy [2] described a general method for deriving a lock-free data structure from an arbitrary data structure, but his approach adds very significantly to the time needed for operations on the data structure: An operation that would take time T in a sequential setting takes time proportional to $T+C$, where C is the time needed to make a logical copy of the data structure; C can be much larger than T. In [1] we present an alternative technique that provides the semantics of exclusive access without relying on mutual exclusion. Using this technique, we devise the *caching method*, a general method of implementing lock-free data structures that is provably better than Herlihy's method for many well-known data structures. The caching method implements an operation of sequential cost T to run in $O(T \log T)$ time.

References

• [1] Greg Barnes. A method for implementing lock-free shared data structures. In *Proc. 5th ACM Symposium on Parallel Algorithms and Architectures*, 1993.

[2] M. Herlihy. A methodology for implementing highly concurrent data objects. In *Proc. 2nd Annual ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming*, pages 197–206, 1993.

3.1.4 Adaptive Dynamic Load Balancing

Investigators: Torben Hagerup and Thomas Lauer

Because of the increasing commercial availability of parallel computers with a significant number of processors, a study of dynamic load balancing was initiated in cooperation with Siemens AG (division ZFE BT SE 42). The goal is to find dynamic load-balancing algorithms that perform well for large systems (asymptotic results), and for smaller systems (commercially available sizes) as well. This is advantageous because you can change the size of your computer and benefit directly from the greater number of processors without changing your load balancing algorithm.

Let us first explain what we mean by load balancing: in order to solve a problem efficiently on a parallel computer we have to distribute the work over the available processors. We call this task *load distribution*, because the *load* (work) must be *distributed* over the processors. We will define *load balancing* as a special kind of load distribution and introduce it below.

If the load pattern is *known in advance*, we can perform load distribution using a static function mapping the *load packages* to the processors; for examples, see [1, 9]. But often the load pattern is data-dependent and not known in advance. So we have to look for *dynamic* load distribution schemes, which distribute the load as it arises. The overall strategy in most of the algorithms concerned with dynamic load balancing is nearly the same: when a load package is generated in a processor, it is sent to another randomly chosen processor, which consumes the package, i.e., carries out the associated task. If the number of packages is sufficiently large ($n > p \log p$, where n is the number of packages and p is the number of processors), one can show that with high probability the processors will have nearly the same number of packages. Some refinements of this strategy with corresponding analysis can be found in [2] and [4].

But this strategy has two disadvantages: all load packages must be sent to another processor, and load packages generated in the same processor with high probability will be consumed in different processors. The first problem is a disadvantage because we have to route many packets even if it is not necessary (for example: if all processors generate nearly the same number of load packages). The second problem is a disadvantage because often a *parent problem* creates many related *child problems* that can be solved more efficiently if they are located on the same processor.

From these disadvantages the idea of load *balancing* arises (we call this strategy load *balancing* to distinguish it from the former idea of simple load *distribution*). In a load balancing algorithm a processor normally works on its own problems and only if its load diverges heavily from the average over all processors it tries to *balance* its load with another processor.

A first attempt at load balancing was made in [10]. That paper gives an algorithm where a processor initiates a balancing step depending on the size of the local load, and it is shown that the expected load on each processor varies only by a constant factor from the average load. Motivated by the practical experiments in [8, 5] and [7] Lüling and Monien proposed another dynamic load balancing scheme in [6]. Here a processor balances its load with another processor if its load has grown/shrunk by more than a constant factor since its last balancing step. In the analysis they also prove that the expected load on each processor varies only by a constant factor from the average load. Moreover some experiments and numerical calculation lead to the suggestion that the variance is also small.

Our goal is to derive bounds showing that not only the expected load but also the actual load of each processor is within a small range around the average load. To do this we extended the algorithm in [6] and gave a high-probability analysis. In 1993 we could already achieve our goal if we allow periodical *counting waves*. These are periodically started computations over the whole network that compute a quantity related to the average load.

During the past year we extended the algorithm further to guarantee that waves are only executed when they are necessary. This means that the number of waves depends on the behavior of the load pattern. Dramatical load changes lead to many counting waves, while a quieter load pattern results in a small number of waves.

In the literature an algorithm capable of changing dynamically according to the behavior of the system is said to be *adaptive*.

We call the method that we use to start waves depending on the load-pattern *distributed counting* and think that this method and analysis may have applications in other fields of parallel and distributed computing.

All results can be found in [3].

References

[1] S. Bhatt and I. Ipsen. How to embed trees in hypercubes. Technical Report YALEU/DCS/RR-443 1985, Yale University, 1985.

[2] R. Karp and Y. Zhang. A randomized parallel branch-and-bound procedure. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 290–300, 1988.

• [3] T. Lauer. *Adaptive dynamische Lastbalancierung.* PhD thesis, Universität des Saarlandes, to appear.

[4] T. Leighton, M. Newman, A.G. Ranade, and E. Schwabe. Dynamic tree embeddings in butterflies and hypercubes. In *Proc. of the 1st ACM Symp. on Parallel Algorithms and Architectures*, pages 224–234, 1989.

[5] R. Lüling and B. Monien. Load balancing for distributed branch & bound algorithms. In *Proc. of Int. Parallel Processing Symposium*, pages 543–549, 1992.

[6] R. Lüling and B. Monien. A dynamic load balancing algorithm with provable good performance. In *Proc. of the 5th ACM Symp. on Parallel Algorithms and Architectures*, pages 164–172, 1993.

[7] R. Lüling, B. Monien, M. Räcke, and S. Tschöke. Efficient parallelization of a branch & bound algorithm for the symmetric traveling salesman problem. In *Proc. of the European Workshop on Parallel Computing*, 1992.

[8] R. Lüling, B. Monien, and F. Ramme. Load balancing in large networks: A comparative study. In *Proc. of the 3rd IEEE Symposium on Parallel and Distributed Processing*, 1991.

[9] B. Monien and I. H. Sudborough. Embedding one interconnection network in another. *Computing Supp.*, 7:257–282, 1990.

[10] L. Rudolph, M. Slivkin-Allalouf, and E. Upfal. A simple load balancing scheme for task allocation in parallel machines. In *Proc. of the 1991 ACM Symp. on Parallel Algorithms and Architectures*, pages 237–245, 1991.

3.1.5 Basic Problems on Meshes

Investigators: Heiko Schröder and Jop F. Sibeyn

Realistic parallel computers have a distributed memory and communicate through an interconnection network. Two fundamental problems on such parallel computers are communication and sorting. In communication typically every processing unit, PU, has a certain number of packets to send, k, and to receive, l. The problem is to route the given set of packets as efficiently as possible to their destinations. This we call the k-l routing problem. In a sorting problem all packets have a certain key, and the processors are indexed in a fixed way. The packets must be rearranged such that they appear in the right order with respect to the indexing.

We consider the case of the $n \times n$ *mesh*, in which n^2 PUs are connected by a regular two-dimensional grid of bidirectional communication links. Every PU may send and receive up to four packets in every step. The natural generalization is a d-dimensional $n \times \cdots \times n$ mesh. Most results mentioned hereafter can be extended to this case.

The most elementary problem is the 1-1 routing problem, also called permutation routing [1, 11, 3]. We achieved considerable improvements over previous algorithms.

In [5] we consider a general approach for routing and sorting on meshes. For k-k sorting we achieve the optimal $k \cdot n/2 + o(k \cdot n)$ steps, with a deterministic algorithm. For the 1-1 sorting problem, we are the first to give a deterministic algorithm requiring only $2 \cdot n + o(n)$ steps. Though of great theoretical importance, these algorithms still have considerable drawbacks: the indexing of the mesh must be quite unnatural, and the 1-1 sorting algorithm is fairly complicated. Furthermore, for practical values of n the performance is rather poor. Therefore, we design in [8] algorithms which perform well for all n, and sort the packets with respect to more natural indexing schemes. We also show how 1-1 sorting can be performed in $2 \cdot n + o(n)$ steps by a much simpler algorithm than in [5].

Even more general than k-k routing is the arbitrary k-l routing problem. For k and l approximately equal, one cannot do much better than to perform a $\max\{k, l\}$-$\max\{k, l\}$ routing. But, when k and l are different, $k = o(l)$ or $l = o(k)$, the routing time can be reduced to $\sqrt{k \cdot l} \cdot n/2 + o(\sqrt{k \cdot l})$ steps [13]. Such more general routing patterns appear for example, in PRAM simulations.

For the very simple case that we have a one-dimensional mesh consisting of n linearly connected PUs, routing and sorting are trivial. For the case that there is an additional connection between the first and the last PU, it is not immediate how a factor two should be gained. This problem is analyzed in [9].

In practice it is often considered undesirable to have to store packets in a PU while a connection over which they have to travel is occupied by another packet. In the so-called hot-potato routing algorithms, packets are always on the move, until they eventually reach their destinations. Only recently it has been discovered that $\mathcal{O}(n)$ steps are sufficient for a permutation routing. In [2] the standing result is considerably improved: it is shown that $3^1\!/\!_2 \cdot n$ steps are sufficient.

There are several other problems that are almost as important as routing and sorting. One of them is list ranking. In many parallel algorithms, particularly algorithms on graphs, one needs to determine the ranks of nodes which are contained in a set of linked lists. In parallel this is a hard problem, because it is not obvious how to break the symmetry. On a parallel computer communicating through an interconnection network we have the additional problem that list ranking is very non-local and appears to inherently require a lot of communication. In [10] we give improved algorithms for list ranking on meshes.

Deterministic PRAM Simulation

Parallel Random Access Machines, PRAMs, are popular as a programming and thinking model, because, by abstracting away the communication issue, they allow us to concentrate on the inherent parallelism of a problem. If we would like to actually perform such a computation, we would need some algorithm which simulates the PRAM on a distributed memory machine.

In [7] we consider the deterministic simulation of a PRAM on a two-dimensional mesh.

Limiting Technology RAM Computation

If processors continue to speed up, and memories continue to grow, the time will come when the basic RAM assumption, that every memory cell can be addressed in unit time, breaks down. In [12] we give a general approach that might be useful in that case.

Reconfigurable Meshes

Technology may allow to have switches at every interconnection, such that several connections can be shorted to form a bus. On a bus one PU may send a packet and any other connected PU may pick the information up. On such a reconfigurable mesh, many problems that are hard to solve efficiently on a normal mesh can now be solved incredibly fast. In [4] we analyze improved k-k routing algorithms for this model of computation.

Layout of the Petersen Graph

Simple architectures with constant degree have clear advantages over more complex architectures. Nonetheless, it is at least of theoretical importance to look for the 'best' architecture. The goodness of a network depends, among other things, on its diameter, its degree, and its bisection width. In these respects the Petersen-graph based family of networks performs very well. Once this has been noticed, it becomes interesting whether this network has a reasonable layout in VLSI [6].

References

- [1] Bogdan S. Chlebus, M. Kaufmann, and Jop F. Sibeyn. Deterministic permutation routing on meshes. In *Proc. 5th Symposium on Parallel and Distributed Processing*, pages 814–821. IEEE, 1993.

- [2] Michael Kaufmann, Harald Lauer, and Heiko Schröder. Fast deterministic hot-potato routing on processor arrays. In *Proc. Annual International Symposium on Algorithms and Computation*, pages 333–341. Springer-Verlag, 1994.

- [3] Michael Kaufmann, Uli Meyer, and Jop F. Sibeyn. Towards practical permutation routing on meshes. In *Proc. 6th Symposium on Parallel and Distributed Processing*, pages 664–671. IEEE, 1994. Also Tech. Rep. MPI-I-94-153, 1994.

- [4] Michael Kaufmann, Heiko Schröder, and Jop F. Sibeyn. Asymptotically optimal and practical routing on the reconfigurable mesh. Accepted in *Parallel Processing Letters*, 1995.

- [5] Michael Kaufmann, Jop F. Sibeyn, and Torsten Suel. Derandomizing algorithms for routing and sorting on meshes. In *Proc. 5th Symposium on Discrete Algorithms*, pages 669–679. ACM-SIAM, 1994.

- [6] Sabine Öhring, Jop F. Sibeyn, and Ondrej Sýkora. Optimal VLSI-layout for the efficient Petersen based interconnection network family. In *Proc. 6th International Conference Parallel and Distributed Computing and Systems*, pages 121–124. IASTED, 1994. Also Tech. Rep. TR-II-SAS-06/94/22, Institute for Informatics, Slovac Academy of Sciences, 1994.

- [7] Andrea Pietracaprina, Geppino Pucci, and Jop F. Sibeyn. Constructive deterministic PRAM simulation on a mesh-connected computer. In *Proc. 6th Symp. on Parallel Algorithms and Architectures*, pages 248–256. ACM, 1994.

- [8] Jop F. Sibeyn. Desnakification of mesh sorting algorithms. In *Proc. 2nd European Symposium on Algorithms, LNCS 855*, pages 337–390. Springer-Verlag, 1994. Also Tech. Rep. MPI-I-94-102, 1994.
- [9] Jop F. Sibeyn. Deterministic sorting on circular arrays. In *Proc. 8th International Parallel Processing Symposium*, pages 406–410. IEEE, 1994.
- [10] Jop F. Sibeyn. Independent sets and list ranking on meshes. In *Proc. Computing Science in the Netherlands*, pages 271–280, Amsterdam, Netherlands, 1994. SION.
- [11] Jop F. Sibeyn, Bogdan S. Chlebus, and Michael Kaufmann. Shorter queues for permutation routing on meshes. In *Proc. 19th Symposium on the Mathematical Foundations of Computer Science, LNCS 841*, pages 597–607. Springer-Verlag, 1994.
- [12] Jop F. Sibeyn and Tim Harris. Exploiting locality in LT-RAM computation. In *Proc. 4th Scandinavian Workshop on Algorithm Theory*, pages 338–349. Springer-Verlag, 1994.
- [13] Jop F. Sibeyn and Michael Kaufmann. Deterministic 1-k routing on meshes. In *Proc. 11th Symposium on Theoretical Aspects of Computer Science, LNCS 775*, pages 237–248. Springer-Verlag, 1994. Also Tech. Rep. MPI-I-93-163, 1993.

3.1.6 Locality in Parallel Algorithms

Investigator: Pierre Kelsen

We consider a model of parallel computation in which the processors are nodes in a graph with the edges representing direct communication links. We call this graph the "communication graph". We assume that all processors are synchronized. In one time step each processor can do some local computation and exchange messages with each of its neighbors in the communication graph. Thus, after t time steps each processor has collected information about the nodes at distance at most t from itself. In order to allow nontrivial problems to be solved we need to have some symmetry-breaking rule. We follow the standard approach of assigning unique labels in the range $\{1, \ldots, n\}$ to the n processors (see e.g. [6]). We shall refer to this model of computation as the "distributed model".

We are interested in the complexity of the $\Delta + 1$-coloring problem in this model, i.e., the problem of coloring the vertices of a graph with maximum degree Δ with $\Delta + 1$ colors so that no two adjacent vertices receive the same color. The $\Delta + 1$ coloring problem is closely related to the maximal independent set (MIS) problem, i.e., the problem of computing a set of nodes that does not contain two adjacent nodes and that is not properly contained in another set with this property. In fact there is a well-known NC reduction from $\Delta + 1$-coloring to MIS (see [8]). This reduction cannot, however, be done efficiently in the distributed model. We give a brief survey of results regarding the complexity of these problems in the distributed model.

If the degree of the communication graph is small, then a technique based on deterministic coin tossing [3] can be used to obtain fast algorithms for several problems including $\Delta + 1$-coloring [4] and MIS. An approach to extend this technique to arbitrary graphs makes use of so-called cluster decompositions: the idea is to decompose the communication graph into connected components of small diameter (clusters) so that the graph induced by the clusters has small chromatic number [2]. Using this technique Awerbuch et al. [2] derived the first sublinear time distributed algorithms for the $\Delta + 1$-coloring problem and the MIS problem. By giving faster algorithms for computing a cluster decomposition Panconesi and Srinivasan [10] obtained somewhat faster algorithms for these problems. The resulting algorithms run in $n^{O(1/\sqrt{\log n})}$ time which is still significantly slower than polylogarithmic time.

In view of these rather weak upper bounds it is natural to investigate lower bounds for these problems. Linial [6] proved a $\Omega(\log^* n)$ lower bound on the complexity of $\Delta + 1$ coloring and MIS on a ring. This bound is tight because of earlier results of Cole and Vishkin [3]. (Linial also gives a $O(\log^* n)$ time distributed algorithm for coloring a graph with $O(\Delta^2)$ colors.) Naor and Stockmeyer [9] investigate the class of problems that can be solved in constant time on graphs of bounded degree. Despite these efforts no nontrivial lower bound for $\Delta + 1$-coloring is known for graphs of arbitrary degree.

In [5] we study the complexity of $\Delta + 1$-coloring in the distributed model for two reasons. The first obvious reason for doing this is the rather large gap between lower and upper bounds for this problem for arbitrary graphs. Another reason is the potential for gaining insight into an outstanding open question in this area, namely the question whether derandomization is possible in the distributed model. Indeed all current derandomization techniques require central computation to be carried out in order to zero in on a good sample point. Such an approach would result in distributed algorithms running in time at least proportional to the diameter of the network. On the other hand fast and simple randomized algorithms are known for problems such as $\Delta + 1$-coloring [8] and MIS [7, 1]. Having a derandomization technique that preserves locality would be of eminent interest since it would imply efficient distributed solutions for these problems and many other problems.

The approach we propose in [5] to study the $\Delta + 1$-coloring problem presents a sharp departure from earlier methods based on cluster decomposition. Our method is based on studying the structure of neighborhood graphs. Informally a neighborhood graph represents the structure of the neighborhoods of a fixed radius in the graph. The concept of neighborhood graph was first introduced by Linial [6] in connection with deriving lower bounds on the complexity of graph-theoretic problems in the distributed model.

To describe the results of our work [5], we first remark that the $\Delta + 1$-coloring problem is nontrivial even if we assume that all processors know the graph and that only the labeling of the processors is unknown (e.g., chosen by an adversary). Indeed the best upper bound for this seemingly simpler problem is the same as that for the general problem. We prove in [5] that the problem of coloring a known graph with unknown labeling with $\Delta + 1$ colors can be solved in time $O((\log n)^5)$ in the distributed model with messages of length $O(\log n)$. It is worth pointing out that all known lower bounds (including those of Linial [6]) hold for this restricted problem. Therefore one consequence of our result is that current lower bound techniques are not strong enough to show that $\Delta + 1$ coloring cannot be done in polylog time in the distributed model (for arbitrary graphs).

We extend the previous result to an infinite class of graphs that includes vertex transitive graphs. More precisely, we prove the existence of an algorithm that colors every graph in this class with $\Delta + 1$ colors in time $O((\log n)^5)$ in the distributed model. For this result we assume that messages exchanged in one time step may be of length $O(n\Delta)$.

Both of these results are nonconstructive, i.e., we give existence proofs for these algorithms. These proofs are based on the analysis of a randomized algorithm for $\Delta + 1$ coloring. This analysis is rather difficult because of the complex dependencies of various random variables that are involved. The main tool that we use are martingale inequalities. We believe that the use of martingale inequalities in this context is somewhat unusual and of independent interest.

References

[1] N. Alon, L. Babai, and A. Itai. A fast randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7:567–583, 1986.

[2] B. Awerbuch, A.V. Goldberg, M. Luby, and S.A. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 364–369, 1989.

[3] R. Cole and U. Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inform. and Control*, 70:32–56, 1986.

[4] A.V. Goldberg, S. Plotkin, and G.E. Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM J. Disc. Math.*, 1:434–446, 1989.

• [5] P. Kelsen. Some results on distributed $\Delta + 1$-coloring. Manuscript, 1994.

[6] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comp.*, 21:193–201, 1992.

[7] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comp.*, 15:1036–1053, 1986.

[8] M. Luby. Removing randomness in parallel computation without a processor penalty. *JCSS*, 47:250–286, 1993.

[9] M. Naor and S. Stockmeyer. What can be computed locally ? In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 184–193, 1993.

[10] A. Panconesi and A. Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 581–592, 1992.

3.1.7 Parallel Algorithms for Molecular Dynamics Simulation of Synthetic Polymers

Investigators: Hans-Peter Lenhof and Christine Rüb

The study of molecular motion by molecular dynamics (MD) simulation is an important tool for testing and developing hypotheses about chemical and physical processes. In an MD-simulation the motions of the atoms of a molecular system are simulated using Newton's equation of motions. Given the atomic positions at time t, the interaction forces as well as the atomic positions and velocities of all atoms at time $t + \Delta$ are computed by integrating Newton's equation numerically. The time step Δ of a simulation is typically of the order of 10^{-15} seconds, i.e. every iteration step of the simulation represents 10^{-15} seconds in reality. To study the dynamics of macro molecules it is necessary to carry out the simulation for a longer period of time, i.e., many steps have to be executed. (If we want to simulate the behaviour of a macro molecule for, say, 10^{-8} s, the number of executed steps will be 10^7.) One way to increase the number of time steps that can be simulated is to execute each iteration in parallel.

Over the last ten years many parallel algorithms for MD-simulation of biopolymers (most of them for proteins) have been developed and implemented. Since proteins fold up to very compact configurations (they "collapse"), these parallel algorithms place the protein into a small box. This box is decomposed into cubes that are then distributed among the processors. Each processor computes in each iteration step the interaction forces and new positions of all atoms that are contained in its cubes. There are two causes for communication between the processors: (a) atoms that are contained in cubes of different processors interact with each other; (b) because the atoms move over time, they will sometimes have to be sent to a different processor. Since proteins have a relatively fixed and compact configuration, this method leads to algorithms with modest communication requirements where the work is distributed reasonably well among the processors.

This method has also been used for MD-simulations of synthetic polymers. However, in this case this technique is not very efficient because synthetic polymers show a completely different behaviour from biopolymers: The trajectory of a synthetic polymer is more or less a three dimensional random walk, i.e. very often the atoms have to be moved to different boxes (and processors). Synthetic polymers do not build compact coils, but their long chains build "long loose" coils, i.e., the volume of any box that contains the molecule will be much larger than the volume of the molecule. Since the dynamics of synthetic polymers can be studied by simulating a single macro molecule, most of the cells used above will be empty. This makes it much more difficult to achieve a good load balance.

Our aim is to develop and implement efficient parallel algorithms for MD-simulation of synthetic polymers that make use of the special properties and behaviour of these polymers. These algorithms should run on various parallel machines. We do this in collaboration with Bernd Jung from the Max-Planck-Institute for Polymer Research in Mainz.

One method is the following. We cut the polymer chain into pieces and each piece gets assigned to one processor. This processor is responsible for all atoms in its piece. To carry out the simulation, firstly communication between processors responsible for adjacent subchains is necessary. If subchains are at a certain time near enough to effect each other, also the processors assigned to these have to communicate. This will, in general, be only the case for very few of the subchains.

We have implemented a first version of this method in C [1] that considers only interactions along the chain. This version runs and has been tested on Intel's iPSC/860 and Thinking Machine's CM5. (In Fig. 1 we list some runnig times and speedups for the iPSC/860.) In the next step we want to include the interactions between atoms that are not neighbours in the chain but are located near enough to effect each other.

	300 C-atoms		1000 C-atoms		3000 C-atoms	
# proc.	time in s	speedup	time in s	speedup	time in s	speedup
1	1419	100	4688	100	14048*	100
2	751	95	2409	97	—	—
4	423	83	1337	88	3911	90
8	247	72	721	81	2029	86
16	147	60	400	73	1118	78

Fig. 1. Running times and speedups for 10000 iterations (i.e. 10^{-11} s of real time) on the iPSC/860. Speedups are listed as percentage of the maximum possible (e.g. 60 % of 16 if a molecule with 16 C-atoms is simulated using 16 processors). The marked (*) running time is an estimation since the space requirement for this size of input was too large.

Since January 1995 this project is supported by the DFG in the Schwerpunktprogramm "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen".

References

• [1] P.S. Müller. Parallele Molekulardynamiksimulation. Diplomarbeit, in preparation, Saarbrücken, 1995.

3.2 Computational Geometry

3.2.1 Proximity Problems

Investigators: Sunil Arya, Dave Mount, Christian Schwarz and Michiel Smid

The closest-pair problem

Given a set S of n points in $\mathbb{R}^d$, where d is a fixed constant, the (static) closest pair problem is to find the minimum distance among all pairs of points in S. The dynamic closest pair problem is to maintain the closest pair in the set S as S is modified by insertions and deletions of points. While the static closest problem has been thoroughly studied for a long time, the dynamic problem has attracted much interest recently. A survey on static and dynamic closest pair algorithms appears in the PhD thesis of Christian Schwarz [11], which was completed in August 1993 at this institute.

Returning to the group after a one-year postdoctoral stay at the International Computer Science Institute (ICSI) in Berkeley (USA) in October 1994, Christian Schwarz completed a study on *implementation and application of dynamic closest pair algorithms* [12], started during his stay at ICSI.

The paper discusses the implementation of two recently developed randomized algorithms, both described in [11]. First, there is a randomized incremental algorithm [10] with an expected update time of $O(1)$ for random insertions and deletions of points. The second, more complicated algorithm [9] supports *arbitrary* insertions and deletions in expected time $O(\log n)$. The fundamental data structure in both algorithms is a grid. The stated complexities are valid under the assumption that hashing is used to store points in a grid, with constant access and update time per point. Using balanced trees instead of hashing would increase the running times by a factor of $\log n$.

An important application of dynamic closest pair algorithms is *hierarchical clustering*, which is of interest in statistics. Another application is particle simulation: in each simulation step, we want to know the impact of a point on the other points. One method that is often used to reduce the amount of computation is to cluster the points hierarchically and then compare each point with the clusters in which it is contained rather than with any other point.

While a number of static closest pair algorithms have been implemented, especially for $d = 2$, the only dynamic algorithm that has been implemented is the one that stores the nearest neighbor for each point in the set S and the smallest of these values, which is the closest pair. When a point is inserted into or deleted from S, this information is maintained in time $O(n)$, since a point can only be nearest neighbor to a constant number of points in S.

To do hierarchical clustering with a dynamic closest pair structure, each cluster is represented by a point, e.g. by the mass center of the points that form the cluster. Repeatedly, the two closest clusters are merged, which is implemented by replacing the two points forming the closest pair by a new point, e.g. their weighted average. Hierarchical clustering based on the simple linear-time update algorithm takes time $O(n^2)$. Using our randomized closest pair algorithm for arbitrary updates improves the clustering time to $O(n \log n)$.

As already mentioned, the fundamental structure in both algorithms is a grid. More precisely, the important concept is that of *neighborhood in a grid*. Implementing the algorithms closely to the original description revealed that in practice, the average number of points per grid box is very small. Thus, the searches needed to *access* the neighborhood boxes dominated over the time needed to process the points actually found in this area. Using larger grids would rebalance this relation, but the algorithm for arbitrary updates actually needs the neighborhood concept with the specified grid sizes for its correctness. To alleviate this problem, the concept of a *virtual grid* is introduced. It decouples the logical mesh size from the internal (larger) mesh size which is used to actually store the points. Experiments were conducted to determine good values for internal grid sizes for the use in both closest pair algorithms.

The algorithms were implemented in `C++` using LEDA. Due to the importance of grids in the design of geometric algorithms, the implementation contributes a dedicated data type "grid" that can be used like other LEDA types.

As a first benchmark, the algorithms were tested for point sets of several dimensions under a random update model. Experiments showed that introducing virtual grids drastically decreased the running times of both randomized algorithms. The simplicity of the randomized incremental algorithm over the algorithm for arbitrary updates showed significantly in the running times.

Hierarchical clustering was used as the second benchmark. Having the application to particle simulation in mind, three-dimensional point sets were examined under various distributions. The randomized incremental algorithm which was superior in the first experiment cannot be applied here: hierarchical clustering is a worst-case application of a dynamic closest pair algorithm in the sense that at each update step, the closest pair is modified. The complexity of the randomized incremental algorithm is directly related to the number of changes of the closest pair, however, and as a result, using it for hierarchical clustering would take $\Omega(n^2)$ time.

Therefore, the randomized algorithm for arbitrary updates was compared with the simple linear-time update algorithm. Using virtual grids, the randomized algorithm beats the simple algorithm for problems exceeding roughly 1000 points.

Nearest neighbor searching

The nearest neighbor problem is: given a set of n points in d-dimensional space, $S \subset \mathbb{R}^d$, and given a query point $q \in \mathbb{R}^d$, find the point of S that minimizes the distance to q. Answering nearest neighbor queries is among the most important problems in computational geometry because of its numerous applications to areas such as data compression, pattern recognition, statistics, and learning theory.

The problem of preprocessing a set of n points S so that nearest neighbor queries can be answered efficiently has been extensively studied. Nearest neighbor searching can be performed quite efficiently in relatively low dimensions. However, as the dimension d increases, either the space or time complexities increase dramatically. For $d \geq 3$, there is no known algorithm for nearest neighbor searching that achieves both nearly linear space and polylogarithmic query time in the worst case.

The difficulty of finding an algorithm with the above performance characteristics for nearest neighbor queries suggests seeking weakened formulations of the problem. One formulation is that of computing *approximate nearest neighbors*. Given any $\epsilon > 0$, a $(1+\epsilon)$-*nearest neighbor* of q is a point $p \in S$ such that, for all $p' \in S$

$$\frac{dist(p,q)}{dist(p',q)} \leq 1+\epsilon.$$

Arya and Mount [2] showed that given a point set S and any $\epsilon > 0$, the point set can be preprocessed by a randomized algorithm running in $O(n^2)$ expected time and $O(n \log n)$ space, so that approximate nearest neighbor queries can be answered by a randomized algorithm that runs in $O(\log^3 n)$ expected time.

In [4, 5], Arya *et al.* improve this result in a number of ways. They present an algorithm that preprocesses a set of n points in $\mathbb{R}^d$ in $O(n \log n)$ time, and produces a data structure of space $O(n)$ such that for any query point q and any $\epsilon > 0$, approximate nearest neighbor queries can be answered in $O(\log n)$ time. This improves the results of Arya and Mount significantly in the following respects.

- Space and query time are asymptotically optimal (for fixed d and ϵ) in the algebraic decision tree model of computation.
- The preprocessing is independent of ϵ, so that one data structure can answer queries for all degrees of precision.
- All algorithms are deterministic, rather than randomized, but the code is still quite simple.
- It is possible to insert and delete data points efficiently.
- Constant factors depending exponentially on dimension have been eliminated from the preprocessing time and space. (Exponential constant factors still remain in the query time.)

Because this problem is of considerable practical interest, the importance of the last item cannot be overstated. When dealing with large point sets ($n \geq 10,000$) in moderately large dimensions (say $d \geq 12$), constant factors in space that are on the order of 2^d or $(1/\epsilon)^d$ (even with $O(n)$ space) are too large for practical implementation.

The algorithm for computing nearest neighbors extends to actually enumerating points in approximately increasing distance from the query point. In particular Arya *et al.* show that, after the same preprocessing, given any point q, $\epsilon > 0$, and k, the $(1+\epsilon)$-approximate k nearest neighbors can be computed in $O(k \log n)$ time. They also show that the data structure can be extended to support dynamic updates for point insertion and deletion in $O(\log n)$ time, assuming a model of computation in which bitwise exclusive-or, integer division and integer logarithms can be computed in constant time. Their method is based on a standard technique called *box-decomposition.*

Boundary effects in nearest neighbor searching

Most average-case analyses of nearest neighbor searching algorithms are made under the simplifying assumption that the dimension d is fixed and the number of data points n is so large that *boundary effects* can be ignored. This greatly simplifies the analysis because for any query point (assuming it is chosen from the same distribution as the data points), the statistical distribution of the data points surrounding it can be assumed to be essentially independent of the location of the query point.

However, there are many important applications where the number of data points n and dimension d are related. One such application is *vector quantization*, a technique used in the compression of speech and images. Samples taken from a signal are blocked into vectors of length d (typically after applying some smoothing transforms). Based on a training set of vectors, a set of code-vectors is first precomputed. The technique then encodes each new vector by the index of its nearest neighbor among the code-vectors. The rate r of a vector quantizer is the number of bits used to encode a sample, and it is related to n, the number of code-vectors, by $n = 2^{rd}$. For the common case of $r = 1$, it follows that $n = 2^d$.

For applications in which d and n are related, the theoretical analyses may significantly overestimate the running time of the algorithm. Intuitively, the reason is that when a query point lies close to the periphery of the point set, a significant amount of the space that would otherwise need to be searched for the nearest neighbor may be pruned away. Because exponential constant factors in dimension are one of the main obstacles to extending nearest neighbor searching to much higher dimensions (where many more important applications reside), it is of important practical interest to accurately understand the nature of these factors.

In [3], Arya, Mount and Narayan provide a theoretical explanation of the phenomenon of these *boundary effects* in nearest neighbor searching. They analyze the *bucketing* and *k-d* tree algorithms, taking into account the effects of the boundary. They assume that points are uniformly distributed in a d-dimensional unit hypercube, and that distances are measured using the L_∞ metric. Their main result is that given 2^d points in d dimensions, the expected number of cells visited by the bucketing algorithm grows as 1.566^d. This is significantly smaller than the growth rate of 2^d predicted by previous analyses which ignore boundary effects.

Euclidean spanners

Let $G = (S, E)$ be a weighted graph, and let $d_G(u, v)$ be the length of a shortest path between vertices u and v in G. Let $t > 1$ be any constant. A subgraph G' is a *t-spanner* for G if, for all pairs of vertices u and v, $d_{G'}(u, v) \leq t \cdot d_G(u, v)$. When S is a set of n points in $\mathbb{R}^d$, G is the complete graph, and the length of edge (u, v) is the Euclidean distance between these points, then we call G a complete Euclidean graph and G' a *Euclidean t-spanner*.

Spanners are important geometrical structures, since they provide a mechanism for approximating the complete Euclidean graph in a much more economical form. Of course, a spanner should have a small number of edges (ideally $O(n)$). It is known already for some time how to construct a Euclidean t-spanner having $O(n)$ edges in $O(n \log n)$ time, which is optimal in the algebraic computation tree model. For many applications, it is important that the spanner be endowed with other properties. These include the following:

Low weight: The total sum of the edge lengths in the spanner should be as small as possible. The best that can be hoped for is some constant times the weight of the minimum spanning tree, $O(w(MST))$.

Bounded degree: The number of edges incident to any vertex should be bounded.

Small spanner diameter: The *spanner diameter* (or simply *diameter*) is defined as the smallest integer D such that for any pair u and v of vertices there is a t-spanner path between u and v containing at most D edges. For spanners of bounded degree the best that can be hoped for is logarithmic diameter. In some applications even smaller diameters may be desirable, but this comes at the expense of increasing degree.

A natural analogy can be made between spanners and a transportation network of roads connecting a large number of locations. Low weight means that the amount of concrete needed to build the roads is small, bounded degree means that no location in the network has more than a bounded number of roads incident to it, and small diameter means that it is possible to describe any spanner path concisely.

Previous work on spanners has focused on achieving one property or the other. However, a transportation network which achieves small diameter by massively increasing total weight is of little practical value. This suggests the important question of whether there exist spanners that simultaneously achieve *some or all of these properties*.

Arya and Smid [8] considered the problem of constructing a t-spanner that has simultaneously bounded degree and low weight. Such a spanner can easily be constructed by a greedy algorithm that considers all pairs of points in increasing order of their distances. Clearly, implementing the algorithm in a straightforward way leads to a running time that is at least quadratic. Aray and Smid show how this greedy algorithm can be implemented so that the running time is bounded by $O(n \log^d n)$. Each vertex in the resulting spanner has a degree that is bounded by a constant, and the total weight of the spanner is proportional to the weight of a minimum spanning tree of the points. We note that previously, the best known algorithm for constructing a bounded degree spanner—without any constraints on the total weight—had quadratic running time.

In [6, 7], Arya, Mount and Smid consider spanners of small diameter. First, they give a very simple randomized algorithm for constructing a t-spanner with diameter $O(\log n)$. This algorithm generalizes Skip Lists—more precisely the *analysis* of this data structure—in a non-trivial way. The algorithm has running time $O(n \log^{d-1} n)$. It turns out that this spanner can be maintained under insertion and deletions of points in the model of random updates. Previously, no spanners were known that can be maintained under updates. Arya, Mount and Smid also give a deterministic algorithm for constructing a t-spanner having diameter $O(\log n)$. This algorithm is based on so-called well-separated pair decompositions and its running time is bounded by $O(n \log n)$, which is optimal.

In Arya *et al.* [1], a number of new constructions for spanners are presented. In almost all cases these constructions are provably *optimal* from the perspectives of computation time, space, and performance on the properties listed above. The problem is complicated by the fact that there are obvious tradeoffs between these properties. (For example, reducing diameter requires the creation of long edges, which in turn increases total weight, or may increase the number of edges needed in the spanner.) For this reason, we consider all possible combinations of these properties.

The results of [1] arise from a number of improved techniques in spanner constructions, but one deserves particular mention. An important data structure used in the construction of spanners is the well-separated pair decomposition, introduced by Callahan and Kosaraju. This structure represents the $O(n^2)$ pairs of points using only $O(n)$ pairs of geometrically "well-separated" pairs of subsets of points. Arya *et al.* [1] give a novel method of further decomposing a well-separated pair decomposition into a constant number of hierarchically organized sets of well-separated pairs. Using this decomposition, a class of spanners can be viewed as being the union of a constant number of trees. Moreover, each of the $O(n^2)$ spanner paths arises as the unique path between two leaves in one of these trees. The fact that the $O(n^2)$ spanner paths can be partitioned among a constant number of trees is a rather remarkable fact in itself, and suggests a great deal about special structure of these graphs.

Among the results in [1] is the construction of a t-spanner of diameter $O(\alpha(n))$ with only $O(n)$ edges, where $\alpha(n)$ is the inverse of the Ackermann function, a very slowly growing function. (Of course, this spanner does not have bounded degree.) This is remarkable, because at first sight one may think that $\log n$ is a lower bound on the diameter of spanners with a linear number of edges.

The main result in [1] is an algorithm for constructing a t-spanner that has *simultaneously* bounded degree, small diameter, and low weight. This algorithm has running time $O(n \log n)$, which is optimal. Hence, this paper basically answers all questions regarding spanners.

References

• [1] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: short, thin, and lanky. In *Proc. 27th Annual ACM Symposium on the Theory of Computing (STOC'95), to appear*, 1995.

[2] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 271–280, 1993.

• [3] S. Arya, D. M. Mount, and O. Narayan. Accounting for boundary effects in nearest neighbor searching. In *Proc. 11th Ann. ACM Sympos. Comput. Geometry, to appear*, 1995. Also Tech. Rep. MPI-I-94-159.

• [4] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 573–582, January 1994.

• [5] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. Manuscript (revised), January 1995.

• [6] S. Arya, D. M. Mount, and M. Smid. Dynamic algorithms for geometric spanners of small diameter: randomized solutions. Technical Report MPI-I-94-156, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1994.

• [7] S. Arya, D. M. Mount, and M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 703–712, 1994.

• [8] S. Arya and M. Smid. Efficient construction of a bounded degree spanner with low weight. In *Proc. 2nd Annu. European Sympos. Algorithms (ESA)*, volume 855 of *Lecture Notes in Computer Science*, pages 48–59, 1994.

[9] M. J. Golin, R. Raman, C. Schwarz, and M. Smid. Randomized data structures for the dynamic closest-pair problem. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 301–310, 1993.

[10] M. J. Golin, R. Raman, C. Schwarz, and M. Smid. Simple randomized algorithms for closest pair problems. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 246–251, 1993.

• [11] C. Schwarz. *Data structures and algorithms for the dynamic closest pair problem.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, December 1993.

• [12] C. Schwarz. Dynamic closest pair algorithms: implementation and application. Technical Report MPI-I-95-1-003, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1995.

3.2.2 Range Searching

Investigators: Sunil Arya and Dave Mount

The range searching problem is among the fundamental problems in computational geometry. A set P of n data points is given in d-dimensional real space, $\mathbb{R}^d$, and a space of possible *ranges* is considered (e.g. d-dimensional rectangles, spheres, halfspaces, or simplices). The goal is to preprocess the points so that, given any query range Q, the points in $P \cap Q$ can be counted or reported efficiently. More generally, one may assume that the points have been assigned weights, and the problem is to compute the accumulated weight of the points in $P \cap Q$, $weight(P \cap Q)$, under some commutative semigroup.

There is a rich literature on this problem. For many applications, the number of data points is sufficiently large that one is limited to using only linear or roughly linear space in solving the problem. Most research has focused on solving this problem exactly, but lower bounds show that if linear space is assumed, the problem cannot be solved in polylogarithmic time. This suggests that it may be worthwhile considering variations of the problem, which may achieve these better running times. In [1], Arya and Mount consider an approximate version of range searching. Rather than approximating the count, they consider the range to be a *fuzzy* range, so that data points that are "close" to the boundary of the range (relative to the range's diameter) may or may not be included in the count.

More precisely, given a bounded range Q of diameter s, and given $\epsilon > 0$, define Q^- to be the locus of points whose distance from a point exterior to Q is at least $s\epsilon$, and Q^+ to be the locus of points whose distance from a point interior to Q is at most $s\epsilon$. Define a *legal answer* to an ϵ-approximate range query to be $weight(P')$ for any subset P' such that

$$P \cap Q^- \subseteq P' \subseteq P \cap Q^+.$$

This definition allows for two-sided errors, by failing to count points that are barely inside the range, and counting points barely outside the query range.

The result of such an approximate range search is probably interesting only for fat ranges. Overmars [2] defines an object Q to be *k-fat* if for any point p in Q, and any ball B with p as center that does not fully contain Q in its interior, the portion of B covered by Q is at least $1/k$. For ranges that are not k-fat, the diameter of the range is potentially very large compared to the thickness of the range at any point. However, there are many applications of range searching which involve fat ranges.

There are a number of reasons that this formulation of the problem is worth considering. It is well known that what seems to make range queries "hard" to solve are the points that are near the boundary of the range. However, there are many applications where data are imprecise, and ranges themselves are imprecise. For example, the user of a geographic information system that wants to know how many single family dwellings lie within a 60 mile radius of Manhattan, may be quite happy with an answer which is only accurate to within a few miles. Furthermore, the user is free to adjust the value of ϵ to whatever precision is desired, understanding that a tradeoff in running times is involved.

In [1], Arya and Mount show that by allowing approximate ranges, it is possible to achieve significant improvements in running times, both from a theoretical as well as practical perspective. They show that (for fixed dimension) after $O(n \log n)$ preprocessing, and with $O(n)$ space, ϵ-approximate range queries can be answered in time $O(\log n + 1/\epsilon^d)$. Under the assumption that ranges are convex, this can be strengthened to $O(\log n + 1/\epsilon^{d-1})$. Some of the features of their method are

- The data structure and preprocessing time are independent of the space of possible ranges and ϵ. This only assumes that in constant time (depending on dimension) it is possible to determine whether there is a nonempty intersection between the inner and outer ranges (Q^- and Q^+) and a cube.

- Space and preprocessing time are free of exponential factors in dimension. Space is $O(dn)$ and preprocessing time is $O(dn \log n)$.

- The algorithms are quite simple. The data structure is a variant of the well-known quadtree data structure.

- Implementation and experimental results show that even for uniformly distributed points in dimension 2, there is a very significant improvement in the running time if a small approximation error is allowed.

They also present a lower bound of $\Omega(\log n + 1/\epsilon^{d-1})$, for the complexity of answering ϵ-approximate range queries assuming a partition tree approach for cubical range in fixed dimension. Thus their approach is optimal under these assumptions for convex ranges.

References

- [1] S. Arya and D. M. Mount. Approximate range searching. In *Proc. 11th Ann. ACM Sympos. Comput. Geometry, to appear*, 1995.

[2] M. H. Overmars. Point location in fat subdivisions. *Inform. Process. Lett.*, 44:261–265, 1992.

3.2.3 Abstract Voronoi Diagrams

Investigator: Ronald Rasch

Since 1975, when Shamos and Hoey discovered Voronoi diagrams for computer science, Voronoi diagrams have been among the structures most frequently investigated in Computational Geometry. This is motivated by the wide range of applications for which Voronoi diagrams have been proved a powerful tool. In general, the Voronoi diagram is defined for a space M, a finite set S of sites and a distance measure d giving a distance for each pair (x, s) with $x \in M$ and $s \in S$. In this setting the Voronoi diagram partitions M into regions such that each region contains all points of M having the same closest (or furthest) site among the elements of S. Different types of Voronoi diagrams are obtained by varying the space. Frequently, we have $M = \mathbb{R}^2$, sites are points, spheres, or polyhedra, and the distance function is the L_p-norm, a convex distance function, or a weighted distance function.

With the notion of abstract Voronoi diagram, Klein has proposed a unifying approach to closest-point Voronoi diagrams for the important case $M = \mathbb{R}^2$. Klein's approach covers the most important types of closest point Voronoi diagrams in the plane by replacing the notion of distance by the topological concept of bisecting curves.

We have contributed to the mathematical as well as the algorithmic treatment of both closest-point and furthest-point abstract Voronoi diagrams. Our results about the closest-point abstract Voronoi diagrams were mentioned in the previous report.

This year, Rasch completed his Ph.D. Thesis [1]. He shows that the theory of closest-point abstract Voronoi diagram can be generalized to furthest-point diagrams. Furthest-point abstract Voronoi diagrams are shown to have a tree structure and linear complexity. The work includes an $O(n \log n)$ time randomized incremental algorithm for the construction of such a diagram. Note that in this way, we obtain a generic algorithm for the construction of all types of furthest-point Voronoi diagrams that fit in the abstract Voronoi diagram model. Moreover, this algorithm is optimal in the algebraic computation tree model. Examples of sites and distance functions that fall in the abstract model are

1. point sites under the L_1- and L_2-metric,

2. circle sites under the L_2-metric,

3. line segments under the L_2-metric,

4. power diagrams.

It is well known that the closest- and furthest-point Voronoi diagrams, for the case where the sites are planar points and the distance metric is the Euclidean metric, can be computed in linear time provided the sites are in convex position, i.e., all sites lie on their convex hull. Rasch defines the notion of convex position in the abstract Voronoi diagram model, and he shows that for sites in convex position the diagram can be computed in linear time.

References

• [1] R. Rasch. *Abstrakte inverse Voronoidiagramme.* PhD thesis, Dept. Comput. Sci., Univ. Saarlandes, Saarbrücken, Germany, 1994.

3.2.4 Geometric Optimization Problems

Investigators: Michiel Smid and Christian Thiel

A problem from neurosurgery

Geometric optimization problems in low-dimensional spaces have received great attention. Such problems often occur in practical situations. Consider the following example from the field of neurosurgery: A surgeon wants to remove tissue samples from the brain of a patient for diagnosis purposes. This is done by inserting a probe through a small hole in the skullcap of the patient. In order to minimize the exposure to danger, the point of entry has to be chosen in such a way that the trajectory of the probe stays away from certain brain areas. If we model this trajectory as a ray, and the brain areas we want to avoid by weighted points in three-dimensional space, then we want to find a ray R emanating from the position at which we want to remove the tissue sample such that the minimal weighted distance from any of the points to R is maximal.

The Euclidean distance between two points p and q is denoted by $d(p,q)$. If p is a point in $\mathbb{R}^d$, and R is a closed subset of $\mathbb{R}^d$, then the distance between p and R is defined as $d(p,R) := \min\{d(p,q) : q \in R\}$. Finally, we define an anchored ray as a ray that emanates from the origin.

The above mentioned optimization problem is the three-dimensional version of the following problem: Given a set S of n points in $\mathbb{R}^d$, such that each point p of S has a weight $w(p)$, compute an anchored ray R for which $\min_{p\in S} w(p) \cdot d(p,R)$ is maximal.

Smid and Thiel [2] show how to solve the planar version of this problem in $O(n \log n)$ time, which is optimal. Previously, the best known algorithm for this problem had running time $O(n\alpha(n) \log n)$, where $\alpha(n)$ is the inverse of the Ackermann function. Hence, the result of Smid and Thiel is only a modest improvement. The interesting fact about their algorithm, however, is its simplicity. Also, the analysis uses basic concepts from combinatorial geometry, such as lower envelopes and Davenport-Schinzel sequences, in an elegant way.

The three-dimensional version of the anchored ray problem can be solved in roughly quadratic time, again by using lower envelopes. Smid and Thiel show how to use the parametric search technique to improve this to $O(n \log^5 n)$ time, which is a drastic improvement.

The width and roundness of a point set

Another application of geometric optimization problems is in the area of shape analysis. In this field, we want to approximate a point set by a simple geometric figure. As an example, assume we have a large number of mass-manufactured circular profiles. In order to test the quality of such a profile, we take sample points from its surface. The profile is acceptable if the smallest annulus that contains all these sample points has a width that is less than some tolerance factor. (The American National Standards Institute recommends this measure to be used for testing circular profiles.)

Smid and Janardan [1] consider two such approximation problems. Before we formulate them, we need some definitions. A slab is defined as the closed region lying between any two parallel lines in the plane, and an annulus as the closed region lying between any two concentric circles of finite radius in the plane. The width of a slab (resp. annulus) is defined as the distance between its bounding lines (resp. the difference of the radii of its bounding circles).

The two problems considered in [1] are: Given a set S of points in the plane, compute its *width*, which is defined as the minimum width of any slab that contains all points of S, respectively compute its *roundness*, which is defined as the minimum width of any annulus that contains all points of S.

The problem of computing the width of a planar point set was considered already previously. Smid and Janardan give a new characterization of the width. This leads to a new $O(n \log n)$ time algorithm for computing the width. In order to give this new characterization, we need to introduce some notions.

Let $NVD(S)$ (resp. $FVD(S)$) denote the closest-point (resp. furthest-point) Voronoi diagram of S. If e is an unbounded edge of $NVD(S)$ or $FVD(S)$, then $w(e)$ will denote the limit of $d(x, F(x)) - d(x, N(x))$, if x goes to infinity on e. Then the width of S is equal to the minimum value of $w(e)$ over all unbounded edges e of $NVD(S)$. Also, the width of S is equal to the minimum value of $w(e)$ over all unbounded edges e of $FVD(S)$.

The problem of computing the roundness of a planar point set has received considerable attention recently. For $x \in \mathbb{R}^2$, let $N(x)$ (resp. $F(x)$) denote a nearest (resp. furthest) neighbor of x in S. Then the minimum-width annulus centered at x that contains all points of S has width $d(x, F(x)) - d(x, N(x))$, where $d(\cdot, \cdot)$ is the Euclidean distance function. Hence, the problem is to minimize the function $d(x, F(x)) - d(x, N(x))$ over all points x in the plane.

Let G be the planar graph obtained by superimposing these two diagrams. In many places in the literature, it is claimed that there is an optimal annulus having its center at a vertex of G. This fact is used to derive efficient algorithms for computing the roundness. It seems, however, that no proof of the claim appears in the literature. Smid and Janardan show that the claim is in fact wrong. Moreover, they give the correct form and give a *rigorous* proof.

The main difficulty in the proof is that the minimum of the function $d(x, F(x)) - d(x, N(x))$ may not exist at all and instead we need to characterize its infimum. For example, consider a set of points on the Y–axis. For any point x in the plane, the value of $d(x, F(x)) - d(x, N(x))$ is positive. If we let x go to infinity on the positive X-axis, then $d(x, F(x)) - d(x, N(x))$ converges to zero. The reader may argue that this only happens if all points are on a line. This is, however, not the case. In [1], an example of a set of points is given, not all on a line, such that *any* annulus containing all points has width strictly larger than one, whereas the points are contained in a slab of width one. As it turns out, this fact is one of the reasons why the proof of the characterization of the roundness is non-trivial. The characterization is as follows: The roundness of the set S is the minimum of the width of S and

$$\min\{d(v, F(v)) - d(v, N(v)) : v \text{ is a vertex of } G\}.$$

Moreover, the following is shown in [1]: Suppose there is an optimal slab such that one of its bounding lines contains exactly one point of S. Then the roundness of S is strictly smaller than the width of S.

References

- [1] M. Smid and R. Janardan. On the width and roundness of a set of points in the plane. Technical Report MPI-I-94-111, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1994.

- [2] M. Smid and C. Thiel. Computing a largest empty anchored cylinder, and related problems. In preparation, 1995.

3.2.5 Intersection-Detection Problems

Investigators: Michiel Smid and Christian Thiel

Generalized intersection searching problems

Problems arising in diverse areas, such as computer graphics, robotics, VLSI layout design, and databases can often be formulated as *intersection searching problems*. In a generic instance of such a problem, a set S of geometric objects is to be preprocessed into a suitable data structure so that given a query object q we can answer efficiently questions regarding the intersection of q with the objects in S. The problem comes in two versions, depending on whether S remains fixed or changes through insertion and deletion of objects—the *static* version and the *dynamic* version, respectively. In the dynamic version, which arises very often owing to the highly interactive nature of the above-mentioned applications, we wish to perform the updates more efficiently than simply recomputing the data structure from scratch after each update, while simultaneously maintaining fast query response times. Typical examples of (S, q)-combinations include:

1. S consists of points in $\mathbb{R}^d$ and q is a d-range: This is the d-dimensional range searching problem.
2. S consists of d-ranges and q is a point in $\mathbb{R}^d$: This is the d-dimensional point enclosure searching problem.
3. S consists of intervals on the real line and q is an interval: This is the interval intersection searching problem.
4. S consists of horizontal line segments in the plane and q is a vertical line segment: This is the orthogonal segment intersection searching problem.
5. S consists of line segments in the plane and q is a line segment: This is the segment intersection searching problem.

Due to their numerous applications, intersection searching problems have been the subject of much study and efficient algorithms have been devised for many of them.

The efficiency of an intersection searching algorithm is measured by the space used by the data structure, the query time, and, in the dynamic setting, the update time. The space and update time are expressed as a function of n, whereas the query time is expressed as a function of both n and the output size k (i.e., the number of intersected objects) and is typically of the form $O(f(n)+k)$ or $O(f(n)+k \cdot g(n))$, for some functions f and g. Such a query time is called *output-sensitive*.

In many applications, a more general form of intersection searching arises: Here the objects in S come aggregated in disjoint groups and of interest are questions regarding the intersection of q with the groups rather than with the objects. (q intersects a group if and only if it intersects some object in the group.) It will be convenient to associate with each group a different color and imagine that all the objects in the group have that color. Then, in the *generalized reporting* problem, we want to report the distinct colors intersected by q; in the dynamic setting, an object of some (possibly new) color is inserted in S or an object in S is deleted. Note that the generalized problem reduces to the standard one when each color class has cardinality one.

We give two examples where such generalized problems arise: (1) Consider the personnel database of a large company, which contains age and salary information for employees across all departments. Given an age–salary range, we wish to report the departments with employees in that range. Clearly, this is an instance of the generalized 2-dimensional range searching problem. (2) VLSI designs often consist of several layers, where each layer is typically comprised of thousands of iso-oriented rectangles. Often it is necessary to wire certain subsets of these layers using vertical channels called vias. Given a candidate position for a via (a point), the layout designer is faced with the problem of identifying the layers that get electrically connected, i.e., those layers that have at least one rectangle containing the via. This can be solved by assigning each layer a different color and solving an instance of the generalized 2-dimensional point enclosure searching problem.

One approach to solving a generalized problem is to try to take advantage of solutions known for the corresponding standard problem. For instance, we can solve a generalized reporting problem by first determining the objects intersected by q (a standard reporting problem) and then reading off the distinct colors. However, the query time can be very high since q could intersect $\Omega(n)$ objects but only $O(1)$ distinct colors. For a generalized reporting problem, we seek query times that are sensitive to the number, i, of distinct colors intersected, typically of the form $O(f(n)+i)$ or $O(f(n)+i \cdot g(n))$, where f and g are polylogarithmic.

Gupta, Janardan and Smid [2, 4, 5] did an extensive study on these generalized intersection searching problems. In [5], a unified approach is given for solving generalized intersection problems on axes-parallel objects, such as points, iso-oriented rectangles, and horizontal/vertical line segments. One of the ingredients is the use of partially persistent data structures. In [2], the results are generalized to objects that can have an arbitrary orientation. Finally, in [4], a generic technique is given for solving generalized intersection searching problems for curved objects such as circles and circular arcs.

The rectangle enclosure problem

The problem of computing intersections in a set of rectangles has received much attention. There are several variants of the problem depending on the notion of "intersection" that is used. Gupta, Janardan, Smid, and Dasgupta [6] consider the following version of the problem: Given a set $\mathcal{R}$ of n axes-parallel rectangles in the plane, report all pairs (R', R) of rectangles such that R *encloses* R'. This problem finds applications in the computer-aided-design of VLSI circuits.

Let k denote the number of pairs (R', R) of rectangles such that R encloses R'. In 1982, Lee and Preparata [7] showed how the problem can be solved in $O(n \log^2 n + k)$ time. They mention as an open problem whether this can be improved. The algorithm of Lee and Preparata has never been improved.

In [6], an algorithm having a running time of $O(n \log n \log\log n + k \log\log n)$ is given. This algorithm first *normalizes* the coordinates in the sense that each coordinate is an integer between one and n. Then, the problem is solved by means of a divide-and-conquer algorithm. In the merge step of this algorithm, we have to report red-blue enclosures in a set of red and blue rectangles. This problem is solved using a sequence of non-trivial sweep steps. Since all coordinates are from a finite universe, we can use van Emde Boas trees in order to search among them in $O(\log\log n)$ time rather than $O(\log n)$ time. In this way, the merge step of the divide-and-conquer algorithm takes $O((n+k')\log\log n)$ time, where k' is the number of dominance pairs that are reported in this step.

Problems on moving objects

Problems involving geometric objects that are in time-dependent motion arise in diverse applications, such as, for instance, traffic control, robotics, manufacturing, and animation, to name just a few. In such problems, we are given a collection of geometric objects, such as points, line segments, or polyhedra, along with a description of their motion, which is usually specified by a low-degree polynomial in the time parameter t. The objective is to answer questions concerning (i) properties of the objects (e.g., the closest pair) at a given time instant t or in the so-called "steady-state", i.e., at $t = \infty$; or (ii) the combinatorics of the entire motion i.e., from $t = 0$ to $t = \infty$ (e.g., the number of topologically different Euclidean minimum spanning trees determined by a set of moving points); or (iii) the existence of certain properties (e.g., collision) or computing the optimal value of some property (e.g., the smallest diameter) over the entire motion.

The systematic study of such dynamic problems was initiated by Atallah [1]. Examples of problems considered by him include computing the time intervals during which a given point appears on the convex hull of a set of moving points and determining the steady-state closest/farthest pair, and smallest enclosing circle for moving points.

In [3], problems of type (iii) are addressed. Specifically, they consider sets of moving objects such as points, line segments, or axes-parallel hyper-rectangles in $\mathbb{R}^d$ and are interested in questions such as: "Do two objects ever collide?" and "What is the smallest inter-point distance or smallest diameter ever attained?" Of course, these problems can be solved easily in quadratic time, by brute-force. Gupta, Janardan and Smid show that many instances of this problem can be solved in subquadratic time. The strategy for solving these dynamic problems is to reduce the problem at hand to a different problem on a set of static objects. The latter problem is then solved using techniques such as sweeping, orthogonal range searching, halfspace range searching, simplex compositions, and parametric search.

Collision detection for moving polyhedra

The demands on quality, security and higher production capacity in manufacturing increase the need for automation during the phase of product design. To find potential faults in the design as soon as possible one uses simulation programs: these predict the physical properties and reactions of the product and check whether particular prefabricated parts can be easily assembled. For the latter purpose, efficient methods for collision detection are needed. In general, collision detection is an essential prerequisite of simulations of mechanical tools.

Schömer and Thiel [8] consider the problem of designing efficient algorithms for collision detection between two moving objects. They assume the following model: (i) Objects are rigid bodies (polyhedra) in $\mathbb{R}^3$, their surfaces consist of planar faces with straight boundaries; (ii) An object may be moving translationally in an arbitrary direction or it may be rotating around an arbitrary axis. These restrictions are based on the fact that real objects can be easily modelled by polyhedra and every motion can be approximated by a sequence of translations and rotations.

It is not difficult to see that in $\mathbb{R}^3$ a collision between a moving polyhedral object and a stationary obstacle is computable in time $O(n^2)$, where n denotes the complexity of the two objects. In fact, previously, no subquadratic algorithms were known for these problems. Even the special case of two convex polyhedra, one of which is rotating has not been solved up to now. This particular problem was posed as an open question by Jack Snoeyink during the Third Dagstuhl Seminar on Computational Geometry in March 1993: Given two convex polyhedra A,B, and an axis of rotation, compute the smallest angle by which B has to rotate to meet A. Can this be done in sub-quadratic time?

Schömer and Thiel give the first sub-quadratic algorithms, which solve the collision problem between two *general* polyhedra, one of which is moving translationally or rotating around a fixed axis, whereas the other is stationary. They get a running time of $O(n^{8/5+\epsilon})$ for the translational movement and $O(n^{5/3+\epsilon})$ for the rotational movement, where ϵ is an arbitrary small positive constant.

The first collision between two polyhedra can either be a collision between a vertex of one polyhedron and a facet of the other or a collision between two edges. The former case is the simpler one and is treated by simple plane sweep techniques. The latter problem is the harder problem. Schömer and Thiel show how to preprocess the set of stationary segments, such that they can efficiently compute the first segment hit by a moving query segment. They proceed in three steps: In the first step the parametric search technique of Meggido is used to reduce the problem of computing the first intersection during the motion to the problem of computing the total number of intersections during the motion. In the second step, the latter problem is reduced to a combination of halfspace and simplex range searching problems; the key technique here is linearization. In the third step the range searching problems are solved using known techniques of van Kreveld and Matoušek. After that the general technique can be applied to the collision problem of line segments which move translationally or rotate around a fixed axis.

References

[1] M.J. Atallah. Some dynamic computational geometry problems. *Computers and Mathematics with Applications*, 11:1171–1181, 1985.

• [2] P. Gupta, R. Janardan, and M. Smid. Efficient algorithms for generalized intersection searching on non-iso-oriented objects. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 369–378, 1994.

• [3] P. Gupta, R. Janardan, and M. Smid. Fast algorithms for collision and proximity problems involving moving geometric objects. In *Proc. 2nd Annu. European Sympos. Algorithms (ESA)*, volume 855 of *Lecture Notes in Computer Science*, pages 278–289, 1994.

• [4] P. Gupta, R. Janardan, and M. Smid. On intersection searching problems involving curved objects. In *Proc. 4th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 824 of *Lecture Notes in Computer Science*, pages 183–194, 1994.

• [5] P. Gupta, R. Janardan, and M. Smid. Further results on generalized intersection searching problems: counting, reporting, and dynamization. *Journal of Algorithms*, to appear, 1995.

• [6] P. Gupta, R. Janardan, M. Smid, and B. Dasgupta. The rectangle enclosure and point-dominance problems revisited. In *Proc. 11th Ann. ACM Sympos. Comput. Geometry, to appear*, 1995. Also Tech. Rep. MPI-I-94-142.

[7] D.T. Lee and F.P. Preparata. An improved algorithm for the rectangle enclosure problem. *Journal of Algorithms*, 3:218–224, 1982.

• [8] E. Schömer and C. Thiel. Efficient collision detection for moving polyhedra. Technical Report MPI-I-94-147, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1994.

3.2.6 Geometric Constraints

Investigator: Vasilis Capoyleas

The problem of Geometric Constraints can be defined as follows: Given a set of unknown geometric objects (which we call *geometries*) and a set of constraints on them, we want to find the feasible sets of objects [4, 5, 3].

With the geometries restricted to points, this is a classical problem in Mathematics; it is the main topic of Distance Geometry. Its algorithmic aspects, however, have received little attention from the theoretical computer science community so far. It is easy to show that even the simplest version: "Given names for points on a line and distances between some pairs of them, find the points", is NP-hard.

However, new practical applications, motivate closer attention. This problem appears in a very explicit manner in Computer-Aided-Design (CAD) systems. A popular new trend is *parametric* CAD systems, in which the position in space of new geometries is not defined directly, but instead, various constraints are given, which the new geometries must satisfy. The system must deduce the correct positions of the geometries, from the constraints [5].

For example, in systems like Pro/Engineer, Euclid, EMS Intergraph, the user is allowed to sketch a set of lines, points and circles representing a design, with the mouse. The system gets some rough idea of the intended places. Then the user identifies some of the sketched geometries, to which he assigns constraints. These can be fixing a distance, fixing an angle, forcing some combinatorial incidences, or even an equation involving expressions in which the values of distances or angles and so on, appear as atoms. Then the system has to find a solution; a set of geometries that satisfy all the constraints. Note that there might be more degrees of freedom than constraints (under-constraining), or a subset of the geometries may involve more constraints than degrees of freedom (over-constraining). Over-constraining and under-constraining are called *structural deficiencies* [6]. Usually, one checks for structural deficiencies, before trying to find a solution.

As noticed above, in practical CAD systems, the user provides the system with a rough sketch. This sketch can be used to obtain some orientation information. This motivates the study of a different formulation of the Geometric Constraint problem, in which, *orientation information about the desired set of geometries is available.* It is no longer clear if the problem is NP-hard.

Two main research directions are *detecting structural deficiencies in a system of geometric constraints* and *solving a system of geometric constraints, when orientation information is available* [4]. Partial and ad hoc solutions, are used successfully in practice [4, 5], but there are not many rigorous theoretical results. When checking for structural deficiencies of systems of points and distances, one can borrow from the theory of rigidity [2].

The Geometric Constraint problem comes in many different versions, depending on the allowable kinds of geometries and the allowable kinds of constraints. Kinds of geometries used in practice, include points, lines, planes, circles, spheres, etc. Kinds of constraints used in practice, are simple or complicated equations, involving distances or angles. The Geometric Constraint problem becomes more and more complicated, as one allows more kinds of geometries and constraints. For almost all the cases, nothing is known. Every case is treated separately by CAD software developers.

Recently, we were able to show that all versions of the Geometric Constraint problem commonly met in practice, *can be reduced to the most simple one*, which involves only points and distances in two dimensions [1]. In this work, we define a large class of Geometric Constraint problems, in two and three dimensions and show that they are all linearly reducible to some equivalent one, involving only points and distances in two dimensions.

Currently, we are trying to use our reduction techniques to cover the more complicated cases, by extending results known for the case of points and distances.

References

• [1] V. Capoyleas. A reduction theory for geometric constraint problems. In preparation, 1995.

[2] J. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity*, volume 2 of *AMS Graduate Studies in Mathematics*. American Mathematical Society, 1994.

[3] G. Kramer. *Solving Geometric Constraint Systems (A case study in kinematics).* MIT Press, 1992.

[4] J. C. Owen. Algebraic solutions for geometry from dimensional constraints. In *Symposium on Solid Modeling Foundations and CAD/CAM Applications*. ACM Press, 1991.

[5] J. C. Owen. Constraints on simple geometry in two and three dimensions. Technical report, D-Cubed Ltd., 68 Castle Street, Cambridge CB3 OAJ, UK., 1993.

[6] W. Whiteley. Constraining plane geometric configurations in cad: Directions and lengths. preprint, June 1994.

3.3 Data Structures and Combinatorial Algorithms

3.3.1 Partial orders

Investigators: Devdatt Dubhashi, Kurt Mehlhorn, Desh Ranjan and Christian Thiel

In the general problem of searching and sorting ordered data structures we are given a partially–ordered data structure modeled by a poset P, and a *storage function* $f : \mathsf{P} \to \mathbb{R}$ such that $p \leq_{\mathsf{P}} q$ implies $f(p) \leq f(q)$. We may think of the elements of the poset as data locations where real numbers are stored consistent with the poset ordering.

In the *searching* problem for partial orders, we are given a real number α and are asked to locate it in the poset (if it is present) with a minimum number of comparisons of the form $\alpha \leq f(p)$ or $\alpha = f(p)$, for $p \in \mathsf{P}$. By the Central Element Theorem of Linial and Saks [3], it follows that the direct information–theoretic lower bound of $\log N$ (where N is the number of *ideals* in P) can indeed be attained *if* one can locate so-called *central* elements in the poset. Hence for the problem of searching ordered data structures, the search for central elements is fundamental. Linial and Saks went on to prove a deep theorem which asserted that *every* poset in fact possesses such central elements. Their proof was, however, non-constructive and the question of whether such elements can actually be found constructively was left open.

The *sorting* problem for general posets is: given a finite set P and an *unknown* partial order $\leq$ on P, determine the partial order using a minimum number of comparisons of the form $p \leq q$ for $p, q \in P$. The central elements again play a key role. Given a procedure to obtain central elements, the sorting problem for posets with n elements and having a total of N ideals can be solved with $O(n \log N)$ comparisons [3]. In [1], we gave an information-theoretic lower bound of $\Omega(n \log N)$ for this problem.

Faigle *et al.* showed in [2] an intimate connection between the problem of generating central elements and the problem of counting ideals in the poset. The latter problem is known to be P-complete in general [4]. In [1], we investigated if randomisation helps in solving the approximate version of this P-complete problem as it indeed does in some other well known cases. We presented schemes for (efficiently) transforming a randomised generation procedure for central elements (which often exists for some classes of posets) into randomised procedures for approximately counting ideals in the poset and for testing if an arbitrary element is central. In turn, we showed how to use this approximate counting of ideals to bootstrap the original generating procedure into one with a strongly amplified probability of generating a central element.

References

• [1] D. Dubhashi, K. Mehlhorn, D. Ranjan, and C. Thiel. Searching, sorting and randomised algorithms for central elements and ideal counting in posets. In *Proceedings 13th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'93)*, pages 436–443. LNCS 761, Springer-Verlag, 1993. Also Technical Report MPI-I-93-154, Max-Planck-Institut Saarbrücken, 1993.

[2] U. Faigle, L. Lovász, R.Schrader, and G. Turán. Searching in trees, series-parallel and interval orders. *SIAM Journal on Computing*, 15:1075–1084, 1986.

[3] N. Linial and M. Saks. Searching ordered structures. *Journal of Algorithms*, 6:86–103, 1986.

[4] J.S. Provan and M.O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12:777–788, 1983.

3.3.2 Lower Bounds for the Matrix Chain Ordering Problem

Investigator: Phil Bradford

An important problem in combinatorial optimization, with many applications, is to find the cheapest way to multiply a chain of n matrices, where the matrices are pairwise compatible but of varying dimensions. The problem is known as the *matrix chain ordering problem* (MCOP) [1, 3]. Two matrices M_i and M_j, of dimensions $d_i \times d_{i+1}$ and $d_j \times d_{j+1}$ respectively, are compatible iff $d_{i+1} = d_j$. A list of compatible matrices is a matrix *chain*. We take the cost of multiplying a $d_i \times d_j$ matrix by a $d_j \times d_k$ matrix as $d_i d_j d_k$. In the product of any chain of n matrices $M_1 \circ M_2 \circ \cdots \circ M_n$, we must consider the $n+1$ dimensions $d_1, d_2, \cdots, d_{n+1}$. Given a matrix chain and multiplying the matrices in this chain according to all valid parenthesizations gives the same result because matrix multiplication is associative. However, products based on different parenthesizations may lead to different total costs.

The MCOP is the focus of much pedagogy because of its amenability to an elementary dynamic programming solution. Many of the popular books on the design and analysis of algorithms and combinatorial optimization use the MCOP as a central example of the dynamic programming paradigm. We have found at least twelve standard text books containing this example, see for example [1, 3, 8]. Furthermore, the fastest known algorithm for the MCOP takes $O(n \lg n)$ time and is due to Hu and Shing [4, 5].

To our knowledge all algorithms for the MCOP and their analyses explicitly or implicitly assume that an n-matrix input to the MCOP can have an optimal solution that is any of the Catalan number of parenthesizations. These assumptions are made without any other considerations. In [2], we have shown that to have certain optimal parenthesizations whose tree representations are of depth $\Theta(n)$ we must have input matrix dimensions that are exponential in n. In particular, to have certain alternating products output as solutions to the MCOP, we need to have exponential inputs.

The comparison based model assumes that the complete cost of any matrix product is atomic. That is, the costs of different possible matrix products can be compared only with each other and every such comparison is of constant cost. In this model an $\Omega(n \lg n)$ lower bound is given for a very restricted case of the MCOP and our comparison based lower bound seems to be implied by Ramanan's algebraic decision tree lower bound [6, 7].

Finally, we have given a trade-off between the input lower bound and the atomic comparison based lower bound. This trade-off is based on the optimal product tree depth. It shows that hard instances of the comparison based model are easy instances in terms of input complexity and vice versa.

An associative product of the form

$$(M_1 \circ ((M_2 \circ ((M_3 \circ \cdots \circ M_{n-3}) \circ M_{n-2})) \circ M_{n-1})) \circ M_n$$

is an *alternating* product. In [2], we have shown for an n-matrix input to the MCOP to generate an alternating product as a solution, the input matrices must be of exponential size. That is, in some sense alternating products are the "hardest" instances for the input lower bound. Products of the form,

$$(((M_1 \circ M_2) \circ (M_3 \circ M_4)) \circ ((M_5 \circ M_6) \circ (M_7 \circ M_8)))$$

are *full balanced* trees since their binary tree representation is a balanced full tree. Fully balanced trees are the hardest instances for our atomic comparison based model.

Prior to our results, algorithms and their analyses did not account for input size for solving the MCOP. However, finding lower bounds for the MCOP has been the focus of some research.

A. C.-C. Yao's work on decision trees has played an important role in the development of good lower bounds for a variety of problems, in the context of our results, see for example [9]. Building on Yao's work, Ramanan showed that such techniques give a lower bound such that [6, 7]:

> "If we could extend our lower bound technique to bounded degree algebraic decision trees, we would have a tight $\Omega(n \lg n)$ lower bound for the [MCOP]."

Ramanan's lower bound technique works on a problem that seems to be a close relative of the matrix chain ordering problem, although it has not yet clinched an $\Omega(n \lg n)$ lower bound for the MCOP. Ramanan recently mentioned that his lower bound proof requires that the input consist of a polynomial number of bits [7, Page 849]: "So our lower bound proof requires that the input consist of a large number of bits." But, again this is only for a problem that seems to be a close relative of the MCOP.

References

[1] A. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

• [2] P.G. Bradford, V. Choppella, and G.J.E. Rawlins. Lower bounds for the matrix chain ordering problem. To appear in the Proceedings of LATIN '95, 1995.

[3] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* McGraw Hill/MIT Press, 1990.

[4] T.C. Hu and M.T. Shing. Computation of matrix product chains. Part I. *SIAM Journal on Computing*, 11:362–373, 1982.

[5] T.C. Hu and M.T. Shing. Computation of matrix product chains. Part II. *SIAM Journal on Computing*, 13:228–251, 1984.

[6] P. Ramanan. A new lower bound technique and its application: Tight lower bound for a polygon triangularization problem. In *Proceedings 2nd ACM-SIAM Symposium on Discrete Algorithms (SODA'91)*, pages 281–290, 1991.

[7] P. Ramanan. A new lower bound technique and its application: Tight lower bound for a polygon triangularization problem. *SIAM Journal on Computing*, 23:834–851, 1994.

[8] R. Sedgewick. *Algorithms, second edition.* Addison-Wesley, 1989.

[9] A.C. Yao. Lower bounds for algebraic computation trees with integer inputs. *SIAM Journal on Computing*, 20:655–668, 1991.

3.3.3 On-Line Algorithms

Investigator: Susanne Albers

The competitive analysis of on-line algorithms is a new fruitful area of research that has developed in the past eight years. On-line algorithms receive the input data incrementally one by one and must react to each new input portion, not knowing future data. On-line problems typically arise in data structuring, scheduling, exploration or learning, and distributed computing. An on-line algorithm is said to be c-competitive if it computes for any input sequence a solution which is at most a factor of c away from an optimal solution for this input.

Randomized Algorithms for the List Update Problem

The list update problem is a fundamental on-line problem in the area of data structures. It is among the first on-line problems that have been studied with respect to competitiveness. The problem is to maintain a set of items as an unsorted, linear linked list. A list update algorithm must serve a sequence of requests to items in the list. Serving a request to the item at position i in the current list incurs a cost of i. Immediately after a request, the requested item may be moved at no extra cost to any position closer to the front of the list. The goal is to serve the request sequence so that the total cost is as small as possible.

The optimal competitive factor of deterministic on-line algorithms for the list update has been known for some time: Sleator and Tarjan [13] showed that the well-known MOVE-TO-FRONT algorithm, which moves an item to the front of the list each time it is requested, is 2-competitive. Karp and Raghavan observed that no deterministic on-line algorithm for the list update problem can be better than 2-competitive. On the other hand, the optimal competitive factor of randomized on-line algorithms has not been determined yet. Irani [9] has presented a randomized on-line algorithm that is $\frac{31}{16}$-competitive. Reingold *et al.* [12] have given a family of COUNTER and RANDOM RESET algorithms that achieve a competitive ratio of $\sqrt{3} \approx 1.73$. This has been the best upper bound known so far for randomized list update algorithms.

We have developed a new family of randomized on-line algorithms that beat the competitive ratio of $\sqrt{3}$. Our improved algorithms are called TIMESTAMP algorithms and achieve a competitiveness of $\max\{2-p, 1+p(2-p)\}$, for any real number $p \in [0,1]$. Setting $p = (3-\sqrt{5})/2$, we obtain a ϕ-competitive algorithm, where $\phi = (1+\sqrt{5})/2 \approx 1.62$ is the Golden Ratio. TIMESTAMP algorithms coordinate the movements of items using some information on past requests. We can reduce the required information at the expense of increasing the competitive ratio. We have presented a very simple version of the TIMESTAMP algorithms that is 1.68-competitive. The family of TIMESTAMP algorithms also includes a new deterministic 2-competitive on-line algorithm that is different from the MOVE-TO-FRONT rule. The results of this work appeared in [3].

On-line Page Replication

Page replication and migration problems are important on-line problems in distributed data management. The problem is to distribute a set of memory pages in a network of processors, each of which has its local memory, so that a sequence of memory accesses can be processed at low cost. More specifically, the goal is to minimize the cost incurred by communication (when a processor wants to read a page that is not in its local memory) and possible re-allocation of pages. In the page migration problem, only one copy of each page may exist. On the other hand, in the page replication problem, multiple copies are allowed and the problem is to determine which local memories should contain a copy of a given page.

Awerbuch *et al.* [5] have presented a deterministic on-line replication strategy for general graphs that achieves an optimal competitive ratio of $O(\log n)$, where n is the number of processors. However, for many important topologies, this bound is not very expressive. Black and Sleator [7] have proposed an optimal deterministic on-line algorithm for trees and uniform networks which is 2-competitive. A uniform network is a complete graph in which all edges have the same length. Koga [10] has developed a randomized replication algorithm for trees that is 1.71-competitive. For the ring topology, randomized replication algorithm with competitive ratios of $2(2+\sqrt{3})$ and 4 have been proposed [6, 10]. No efficient deterministic algorithm for rings was known so far.

We have developed a number of new deterministic and randomized on-line replication algorithms. We concentrated on network topologies that are important in practice and for which on-line algorithms with a constant competitive factor can be given. More specifically, we have developed an optimal randomized on-line replication algorithm for trees and uniform networks; its competitive factor is $\frac{e}{e-1} \approx 1.58$. Furthermore we have considered on-line replication algorithms for rings and presented general techniques that transform large classes of c-competitive algorithms for trees into $2c$-competitive algorithms for rings. As a result we have obtained a randomized on-line algorithm for rings that is 3.16-competitive. We have also derived two very simple 4-competitive on-line algorithms for rings which are either deterministic or memoryless. Our replication results appeared in [4].

Lookahead in On-line Algorithms

In the research period 1991 – 1993, we had mainly worked on the problem of lookahead in on-line algorithms: What improvement can be achieved in terms of competitiveness if an on-line algorithm sees not only the present request to be served but also some future requests? We had introduced two different models of lookahead and had studied "classical" on-line problems such as paging, list update, the k-server problem and metrical task systems using these models. See the Progress Report 1991 – 1993 [1] for a more detailed description of our results. The question of lookahead is still an interesting topic in the on-line community, cf. [11, 8] for some recent papers. Some of our results were published only recently, e.g. our results on list update with lookahead were presented in [2].

References

[1] Progress report 1991–1993. Technical report, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, November 1993.

• [2] S. Albers. A competitive analysis of the list update problem with lookahead. *Proceedings Mathematical Foundations of Computer Science (MFCS'94)*, 1994.

• [3] S. Albers. Improved randomized on-line algorithms for the list update problem. to appear in *Proceedings 6th ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*, January 1995.

• [4] S. Albers and H. Koga. New on-line algorithms for the page replication problem. In *Proceedings 4th Scandinavian Workshop on Algorithm Theory (SWAT'94)*. LNCS 824, Springer Verlag, 1994.

[5] B. Awerbuch, Y. Bartal, and A. Fiat. Competitive distributed file allocation. In *Proceedings 25th ACM Symposium on Theory of Computing (STOC'93)*, pages 164–173, May 1993.

[6] Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. In *Proceedings 24th ACM Symposium on Theory of Computing (STOC'92*, pages 39–50, May 1992.

[7] D.L. Black and D.D. Sleator. Competitive algorithms for replication and migration problems. Technical Report CMU-CS-89-201, Carnegie Mellon University, 1989.

[8] E.F. Grove. On-line bin packing with lookahead. To appear in *Proceedings 6th ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*, January 1995.

[9] S. Irani. Two results on the list update problem. *Information Processing Letters*, 38:301–306, 1991.

[10] H. Koga. Randomized on-line algorithms for the page replication problem. In *Proceedings 4th Symposium on Algorithms and Complexity*, pages 436–445. LNCS 762, Springer-Verlag, 1993.

[11] E. Koutsoupias and C.H. Papadimitriou. Beyond competitive analysis. In *Proceedings 35th IEEE Symposium on Foundations of Computer Science (FOCS'94)*, pages 394–400, 1994.

[12] N. Reingold, J. Westbrook, and D.D. Sleator. Randomized competitive algorithms for the list update problem. *Algorithmica*, 11 (1):15–32, 1994.

[13] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, 28:202–208, 1985.

3.3.4 String Searching

Investigator: Torben Hagerup

Imagine that we are given a table of size n, each of whose entries is a k-character string. The entries in the table are sorted according to the lexicographical order, as in a dictionary, and we want to determine whether a particular k-character string is present in the table, i.e., we want to look up a word in a dictionary. How many character comparisons between the query string and the table are needed in the worst case?

One possibility is to use standard binary search, each stage of which now compares two full k-character strings. The resulting algorithm uses $O(k \log n)$ comparisons. Another simple approach by Hirschberg [3] achieves $O(k+n)$ comparisons, showing that binary search is not always optimal. On the other hand, when k is small relative to n, $O(k+n)$ is not very impressive. Other bounds of $O(k \log n / \log k)$ and $O(k\sqrt{\log n} + \log n)$ were indicated by Hirschberg and Kosaraju [2, 4]. Still, despite the fundamental nature of the problem, its complexity is not known, and no progress was reported for over a decade.

In [1], we showed that the string-searching problem can be solved with

$$O\left(\frac{k \log\log n}{\log\log\left(4 + \frac{k \log\log n}{\log n}\right)} + k + \log n\right)$$

character comparisons in the worst case, which subsumes all other known upper bounds. Despite the complicated form of this bound, we believe that it expresses the exact complexity of the problem (up to a constant factor), and current research aims to demonstrate this by establishing a matching lower bound.

References

• [1] A. Andersson, T. Hagerup, J. Håstad, and O. Petersson. The complexity of searching a sorted array of strings. In *Proceedings 26th ACM Symposium on Theory of Computing (STOC'94)*, pages 317–325, May 1994.

[2] D.S. Hirschberg. A lower worst-case complexity for searching a dictionary. In *Proceedings 16th Allerton Conference on Communication, Control, and Computing*, pages 50–53, 1978.

[3] D.S. Hirschberg. On the complexity of searching a set of vectors. *SIAM Journal on Computing*, 9:126–129, 1980.

[4] S.R. Kosaraju. On a multidimensional search problem. In *Proceedings 11th ACM Symposium on Theory of Computing (STOC'79)*, pages 67–73, May 1979.

3.3.5 Pattern Matching in Compressed Texts

Investigator: Ramesh Hariharan

Pattern matching in strings is an important problem in computer science as well as in other sciences. The simplest form of the problem is to locate a string p, called pattern, in another string s, occurring as a substring of s. This problem has been extensively studied, and we refer the readers to [1] for an excellent survey. There are several well-known algorithms which solve this problem in time linear in the sum of the lengths of the two strings.

The problem we addressed in [3] is that of locating a pattern p in a string s, which is given in a compressed form. In particular, we considered strings given in compressed forms obtained by adaptive dictionary encoding schemes [2], which are essentially encoding schemes proposed by Ziv and Lempel [7, 8], or their variants.

In these schemes, a string s is compressed by replacing a substring of the string by a reference to an earlier occurrence of the same substring in s. Various restrictions, due to efficiency (mostly space efficiency) concerns, lead to different compression schemes. We study this problem, in its full generality, by allowing a compression scheme in which any duplicate occurrence of a string may be replaced by a reference to its earlier occurrence. The reference is usually a pair of integers: a pointer to the starting position of the earlier occurrence of the substring in s, and the length of the substring. Such a compression scheme was studied and implemented in $O(|s|)$ time by [6].

Let the length of the compressed string χ (representing s) be n, and the length of the pattern (uncompressed, i.e., in its original form) be m. Then, we gave an algorithm which runs in time $O(m^2 + n^{3/2} \cdot \log n \cdot \log\log n \cdot \log(\rho(\chi)) + t)$, where $\rho(\chi)$ is the expansion complexity (as opposed to Lempel and Ziv's compression complexity [4]) of the compressed string χ. The exact definition of expansion complexity is rather technical, but it suffices to say that it is upper bounded by the length of s, the uncompressed form of χ. However, if substrings replaced in the compression refer to substrings which overlap heavily with the replaced substring, then the expansion complexity of χ can be order of magnitude lesser. The number of occurrences of p detected in s is given by the output sensitivity factor t.

Note that, if m^2 is greater than $n^{3/2}$, then the m^2 factor in the above complexity is rather expensive. We gave another algorithm which runs in time $O(m \log m + n \cdot (\sqrt{n} \log n \cdot \log\log n \cdot \log(\rho(\chi)) + \log^3 m \cdot \log\log m) + t)$.

We also considered the complexity of locating p in χ, when χ is obtained by other compression schemes. We showed that for most practical compression schemes, which are special cases of [6], including [7, 8, 5], there is an algorithm which runs in time $O(\min(m^2 + n, m \log m + n \cdot \log^3 m \cdot \log\log m) + t)$.

References

[1] A. Aho. Algorithms for finding patterns in strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A*. Elsevier, 1990.

[2] T. Bell, J. Cleary, and I. Witten. *Text Compression.* Prentice Hall, 1990.

• [3] R. Hariharan and C.S. Jutla. Pattern matching in Ziv-Lempel compressed strings. Manuscript, 1994.

[4] A. Lempel and J. Ziv. On the complexity of finite sequences. *IEEE Transactions on Information Theory*, IT-22:75–81, 1976.

[5] V. Miller and M. Wegman. Variations on a theme by Ziv and Lempel. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Words*, pages 131–140. NATO ASI Series, Vol. F12, Springer-Verlag, 1984.

[6] M. Rodeh, V. Pratt, and S. Even. Linear algorithm for data compression via string matching. *Journal of the ACM*, 28:16–24, 1981.

[7] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23:337–342, 1977.

[8] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, IT-24:530–536, 1978.

3.3.6 Probabilistic Analysis of Algorithms

Investigators: Shiva Chaudhuri, Devdatt Dubhashi and Desh Ranjan

The analysis of many randomized algorithms, for example in dynamic load balancing, in the probabilistic divide-and-conquer paradigm and distributed edge-coloring [5, 6, 7, 8, 9], requires ascertaining the precise nature of the correlation between the random variables arising in the following prototypical "balls-and-bins" experiment. Suppose a certain number of balls are thrown uniformly and independently at random into n bins. Let X_i be the random variable denoting the number of balls in the ith bin, $i \in [n]$, where $[n]$ denotes the set $\{1, 2, ..., n\}$. These variables are clearly not independent and are intuitively negatively related, i.e., when one of the variables is "large", another is likely to be "small". In [3] we made this mathematically precise by proving the following type of correlation inequalities:

- For index sets $I, J \subseteq [n]$ such that $I \cap J = \emptyset$ or $I \cup J = [n]$, and any non–negative integers t_I, t_J,
$$\Pr[\sum_{i \in I} X_i \geq t_I \mid \sum_{j \in J} X_j \geq t_J] \leq \Pr[\sum_{i \in I} X_i \geq t_I].$$
- For any disjoint index sets $I, J \subseteq [n]$, any $I' \subseteq I, J' \subseteq J$ and any non–negative integers $t_i, i \in I$ and $t_j, j \in J$,
$$\Pr[\bigwedge_{i \in I} X_i \geq t_i \mid \bigwedge_{j \in J} X_j \geq t_j] \leq \Pr[\bigwedge_{i \in I'} X_i \geq t_i \mid \bigwedge_{j \in J'} X_j \geq t_j].$$

Although these inequalities are intuitively appealing, establishing them is non–trivial; in particular, direct counting arguments become intractable very fast. We proved the inequalities of the first type by an application of the celebrated FKG Correlation Inequality. The proof for the second uses only elementary methods and hinges on some *monotonicity* properties.

More importantly, we then introduced a general methodology that may be applicable whenever the random variables involved are negatively related. Precisely, we invoked a general notion of *negative association* of random variables and showed that:

- The variables X_i are negatively associated. This yields most of the previous results in a uniform way.

- For a set of negatively associated variables, one can apply the Chernoff-Hoeffding bounds to the sum of these variables. This provides a tool that facilitates analysis of many randomized algorithms, for example, the ones mentioned above.

The analysis in [7, 8, 9] seems to strongly hint in another direction, namely that these negatively related variables are in some sense *stochastically dominated* by a set of *independent* random variables with the same marginals. Thereby, one hopes to salvage tools such as the Chernoff–Hoeffding bound also for analysis involving the dependent set of variables. In [4] we exploded myths of this kind and argued that stochastic majorisation in conjunction with an independent set of variables is actually much less useful a notion than it might have appeared.

However, there are positive results to offset the negative ones just mentioned. The performance attributes of a broad class of randomised algorithms can be described by a recurrence relation of the form $T(x) = a(x) + T(H(x))$, where a is a function and $H(x)$ is a random variable. For example, consider an algorithm that, on an input of size x, performs $a(x)$ work to generate a subproblem of size $H(x)$ and then solves the subproblem recursively. (Here $H(x)$ is a random variable taking values in $[0, x]$ and whose distribution depends on the algorithm; moreover, $E[H(x)] \leq m(x)$ for a fixed function $m(x)$ satisfying $0 \leq m(x) \leq x$.) Hence, $T(x)$ is a random variable whose distribution depends on the distribution of $H(x)$. Karp [5] recognised that the analysis of all such algorithms can be conveniently formulated as tail bounds on the distribution of $T(x)$. He gave the following bounds on the tail of the distribution of $T(x)$:

$$\Pr[T(x) \geq u(x) + wa(x)] \leq (m(x)/x)^w$$

where $u(x) = \sum_{i \geq 0} a(m^{(i)}(x))$ and $m^{(0)}(x) = x$ and $m^{(i+1)}(x) = E[H(m^{(i)}(x))]$. The bounds show that the deviation of $T(x)$ from its "expected" value behaves in a fashion similar to that given by Chernoff bounds for the sum of independent random variables. However, the bounds hold for only those distributions of $H(x)$ such that $m(x)/x$ is nondecreasing, which renders the result inapplicable to a number of probabilistic algorithms [2]. Furthermore, the proof, while ingenious offers no intuition as to why the results hold.

In [1], we give an alternative analysis that uses stochastic dominance to reduce the problem to giving a Chernoff-like bound for unbounded random variables. This analysis also allows us to prove similar results for the case when $m(x)/x$ is decreasing, allowing the results to be applied to a wider class of algorithms.

References

- [1] S. Chaudhuri and D. Dubhashi. (Probabilistic) Recurrence Relations Revisited. In *Proceedings II Latinamerican Symposium on Theoretical Informatics (LATIN '95)*, 1995. to appear.

- [2] D. Dubhashi and A. Panconesi. Near optimal distributed edge colouring. Technical Report MPI-I-94-136, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, July 1994.

- [3] D. Dubhashi and D. Ranjan. Some correlation inequalities for probabilistic analysis of algorithms. Technical Report MPI-I-94-143, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, August 1994.

- [4] D. Dubhashi and D. Ranjan. Stochastic majorisation : exploding some myths. Technical Report MPI-I-94-144, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, August 1994.

[5] R. Karp. Probabilistic recurrence relations. In *Proceedings of the 23th ACM Symposium on Theory of Computing (STOC'91)*, pages 190–197, 1991.

• [6] T. Lauer. *Adaptive dynamische Lastbalancierung.* PhD thesis, Universität des Saarlandes, to appear.

[7] A. Panconesi. *Locality in distributed computing.* PhD thesis, Cornell University, 1993.

[8] A. Panconesi and A. Srinivasan. Fast randomized algorithms for distributed edge coloring. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 251–262, 1992.

[9] A. Srinivasan. *Techniques for probabilistic analysis and randomness-efficient computation.* PhD thesis, Cornell University, 1993.

3.3.7 Enumerating Spanning Trees in Graphs

Investigator: Ramesh Hariharan

Enumerating combinatorial objects is a fundamental problem in computer science. Spanning tree enumeration in directed and undirected graphs is one such classical problem, which arises in the solution of electrical networks [6].

There was much early work on this problem, for example [1, 7, 8]. For undirected graphs, Read and Tarjan [9] gave an algorithm which runs in $O(NE + V + E)$ time, on a graph with N spanning trees, E edges and V vertices. Gabow and Myers [2] refined this approach to obtain an algorithm which runs in $O(NV + V + E)$ time. This algorithm is optimal if all spanning trees have to be explicitly output. On the other hand, if only a computation tree which describes relative changes between spanning trees is desired, then the above algorithm is no longer optimal because the output has just $O(N)$ size. The algorithm of Gabow and Myers takes $O(V)$ time per spanning tree even to generate just the computation tree. Note that if desired, all spanning trees can be explicitly enumerated in $O(NV)$ time using the above computation tree. Kapoor and Ramesh [4, 5] gave an algorithm to generate the computation tree of relative changes between spanning trees in $O(N + V + E)$ time, which is optimal.

The case of directed graphs seems to be harder. Shinoda [8] gave an algorithm which could take exponential time per spanning tree in the worst case. Gabow and Myers [2] gave an algorithm which runs in $O(NE + V + E)$ time. Kapoor and Ramesh [4] gave an algorithm to generate the computation tree of relative changes between spanning trees in $O(NV + V^3)$ time. The question that then remained open was whether this computation tree could be generated in $o(V)$ time per spanning tree.

We [3] answered the above question in the affirmative by showing that the above computation tree can be generated using just $O(\log V)$ time per spanning tree. More precisely, we gave an $O(N \log V + V^2\alpha(V, V) + VE)$ algorithm for generating the computation tree ($\alpha(V, V)$ is the functional inverse of Ackermann's function). It is interesting to note that but for one component in the algorithm, the rest of the algorithm runs in just $O(\alpha(V, V))$ time per spanning tree. The "slow" component, which involves maintaining edges in a certain order, takes $O(\log V)$ time per spanning tree. We believe that this order can be maintained more efficiently but have not been able to give an algorithm to do so. An interesting side-effect of our algorithm is a procedure which, given vertex v of the graph, determines which edges occur in none of the spanning trees rooted at v in $O(V\alpha(V, V) + E)$ time.

References

[1] S.M. Chase. Analysis for algorithms for finding all spanning trees of a graph. Technical Report RC3190, IBM T.J. Watson Research Center, 1970.

[2] H.N. Gabow and E. Myers. Finding all spanning trees of directed and undirected graphs. *SIAM Journal on Computing*, 7, 1978.

- [3] R. Hariharan, S. Kapoor, and V. Kumar. Algorithms for generating all spanning trees of directed graphs. Manuscript, 1994.

[4] S. Kapoor and H. Ramesh. Algorithms for generating all spanning trees of undirected, directed and weighted graphs. In *Workshop on Algorithms and Data Structures (WADS'91)*. LCNS 519, Springer-Verlag, 1991.

[5] S. Kapoor and H. Ramesh. Algorithms for generating all spanning trees of undirected and weighted graphs. *SIAM Journal on Computing*, 24, 1995.

[6] W. Mayeda. *Graph Theory*. John Wiley, NY, 1972.

[7] G.J. Minty. A simple algorithm for listing all trees of a graph. *IEEE Transactions on Circuit Theory*, CT-12:120–128, 1965.

[8] S. Shinoda. Finding all possible directed trees of a directed graph. *Electron Communication, Japan*, 51-A:45–47, 1968.

[9] R.E. Tarjan and R.E. Read. Bounds on backtrack algorithms for listing cycles, paths and spanning trees. *Networks*, 5:237–252, 1975.

3.3.8 Degree Sequence Problems

Investigators: Srinivasa Arikati and Anil Maheshwari

An important problem in graph algorithms is to compute a (simple) graph satisfying the given degree constraints. An integer sequence d is called a *degree sequence* if there exists a graph G such that the degrees of its vertices are equal to the components of the sequence d. The graph G is said to be a *realization* of the sequence d.

Given an integer sequence d of length n, there are two problems of interest: the *decision problem* is to test if d is realizable; the *search problem* is to compute a realization of d. A characterization of degree sequences known as the Erdős-Gallai inequalities [3] results in an $O(n)$-time algorithm for the decision problem. Another characterization called the Havel-Hakimi characterization [3] leads to an efficient algorithm for the search problem and the algorithm can be implemented in $O(n \log\log n)$ time (see, e.g. [2]). Designing an $O(n)$-time algorithm for the search problem has been open so far. We presented an $O(n)$-time algorithm for computing a realization G of a given degree sequence [1]. Observe that G may have $\Omega(n^2)$ edges. Our algorithm computes an implicit representation of G and this representation needs only $O(n)$ space.

References

- [1] S.R. Arikati and A. Maheshwari. An $O(n)$ algorithm for realizing sequences. In *Proceedings 14th Conference on Foundations of Software Technology and Computer Science (FSTTCS'94)*, pages 125–136. LNCS 880, Springer-Verlag, December 1994.

[2] T. Asano. Graphical degree sequence problems with connectivity requirements. In *Proceedings 4th International Symposium on Algorithms and Computation (ISAAC'93)*, pages 38–47. LNCS 762, Springer-Verlag, December 1993.

[3] L. Lovász and M. Plummer. *Matching Theory*. Academic Press, Budapest, Hungary, 1986.

3.3.9 Random Walks on Graphs

Investigator: Greg Barnes

Consider a simple random walk on G, an undirected graph with n vertices and m edges. At each time step, if the walk is at vertex v, it moves to a vertex chosen uniformly at random from the neighbors of v. Random walks have been studied extensively, and have numerous applications in theoretical computer science, including space-efficient algorithms for undirected connectivity [4, 8], derandomization [1], recycling of random bits [12], approximation algorithms [9, 14], efficient constructions in cryptography [11], and self-stabilizing distributed computing [13].

Frequently (see, for example, Karger *et al.* [16] and Nisan *et al.* [17]), we are interested in $E[T(\mathcal{N})]$, the expected time before a simple random walk on an undirected connected graph, G, visits its $\mathcal{N}^{th}$ distinct vertex, $\mathcal{N} \leq n$. The corresponding question for edges is also interesting, and arises in the work of Broder *et al.* [8]: how large is $E[T(\mathcal{M})]$, the expected time before a simple random walk on an undirected connected graph, G, traverses its $\mathcal{M}^{th}$ distinct edge, $\mathcal{M} \leq m$? In [6], we gave upper bounds on $E[T(\mathcal{N})]$ and $E[T(\mathcal{M})]$ for arbitrary graphs. While a great deal was previously known about how quickly a random walk covers the entire graph (see, for example, [2, 4, 7, 15, 18]), little was known about the behavior of a random walk before the vertices are covered. These bounds help fill the gaps in our knowledge of random walks, giving a picture of the rate at which a random walk explores a finite or an infinite graph.

Aleliunas *et al.* [4] showed that the expected time to visit *all* vertices of an arbitrary graph (called the *cover time*) is $O(mn) \leq O(n^3)$. Using this bound, Linial derived a bound for general $\mathcal{N}$ of $E[T(\mathcal{N})] = O(\mathcal{N}^4)$ [16, Lemma 4.1]. Linial [personal communication] conjectured that the cover time bound generalizes to all $\mathcal{N}$, that is, $\forall \mathcal{N} \leq n, E[T(\mathcal{N})] = O(\mathcal{N}^3)$. We proved Linial's conjecture.

Theorem 1 *For any connected graph on n vertices, and for any $\mathcal{N} \leq n$, $E[T(\mathcal{N})] = O(\mathcal{N}^3)$.*

Zuckerman [18] proved an upper bound of $O(mn)$ on the time to traverse all edges in a general graph. We are unaware of any previous nontrivial bounds for $\mathcal{M} < m$. We proved:

Theorem 2 *For any connected graph with m edges, and for any $\mathcal{M} \leq m$, $E[T(\mathcal{M})] = O(\mathcal{M}^2)$.*

Theorem 2 holds even if G is not a simple graph (i.e., if we allow self-loops and parallel edges). Let $E[T(\mathcal{M},\mathcal{N})]$ be the expected time for a simple random walk to either traverse $\mathcal{M}$ distinct edges or visit $\mathcal{N}$ distinct vertices (whichever comes first). Then the following theorem implies both the above theorems, by considering $E[T(\mathcal{N}^2,\mathcal{N})]$ and $E[T(\mathcal{M},\mathcal{M})]$, respectively.

Theorem 3 *For any connected graph with m edges and n vertices, and for any $\mathcal{M}$ and $\mathcal{N}$ such that $\mathcal{M} \leq m$ or $\mathcal{N} \leq n$, $E[T(\mathcal{M},\mathcal{N})] = O(\mathcal{M}\mathcal{N})$.*

In the above three theorems, the graph G need not be finite. If G is a graph with infinitely many vertices (each vertex of finite degree), then we can consider only the finite portion of G that is within distance $\mathcal{N}$ (or $\mathcal{M}$) from the starting vertex of the random walk, and the proofs remain unchanged. For finite graphs, the following theorem serves to complete the picture of the rate at which vertices (or edges) are discovered. It provides better bounds than Theorems 1 and 2 when the number of vertices to be discovered is larger than $\sqrt{m}$ or the number of edges to be discovered is larger than n.

Theorem 4 *For any simple connected graph on n vertices and m edges, for any $\mathcal{N} \leq n$, $E[T(\mathcal{N})] = O(m\mathcal{N})$, and for any $\mathcal{M} \leq m$, $E[T(\mathcal{M})] = O(n\mathcal{M})$.*

Our theorems are the best possible in the sense that there exist graphs for which the bounds are tight up to constant factors (e.g., the n-cycle for Theorem 2). However, these bounds can be refined if additional information regarding the structure of G is given. The work of Kahn *et al.* [15] indicates that d_{min}, the minimum degree of the vertices in the graph G, is a useful parameter to consider. They showed that the expected cover time of any connected graph is $O(mn/d_{min})$, implying a cover time of $O(n^2)$ for regular graphs. This inverse dependency on d_{min} applies also to short random walks. Preliminary results in this direction (tight up to a logarithmic factor) were presented in [5]. The superfluous logarithmic factor in these results was subsequently removed by Feige [10], building upon proof techniques that were developed by Aldous [3]. Aldous is writing a textbook giving a systematic account of random walks on graphs and reversible Markov chains. The current draft [3] contains results similar to ours in the regular graph setting.

While the short term behavior of random walks is worth studying in its own right, short random walks also have immediate applications in many areas of computer science. Our results, of course, cannot be applied to all such areas. For example, much stronger results are already known about the properties of short random walks on the special class of graphs known as *expanders* (see, for example, Ajtai *et al.* [1], and Jerrum and Sinclair [14]). One might hope our results would dramatically improve the algorithms of Karger *et al.* [16] and Nisan *et al.* [17] for undirected connectivity. As mentioned above, both require an estimate of $E[T(\mathcal{N})]$ (and both used the estimate $E[T(\mathcal{N})] = O(\mathcal{N}^4)$). Unfortunately, substituting our bound only improves the constants for the algorithms, since the running times of both depend on the *logarithm* of $E[T(\mathcal{N})]$, not $E[T(\mathcal{N})]$.

Our results may yield significant improvements for other randomized algorithms. In particular, consider randomized time-space tradeoffs for undirected $\mathcal{S}$-$\mathcal{T}$ connectivity (USTCON), as studied by Broder *et al.* [8]. One key property of Broder *et al.*'s algorithm is that a short random walk from a given edge traverses many edges. Improved bounds on $E[T(\mathcal{M})]$, then, would seem to provide an improvement to their tradeoff. Partial results in this direction were presented in [5], and further improvements are presented by Feige [10].

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi. Deterministic simulation in LOGSPACE. In *Proceedings 19th ACM Symposium on Theory of Computing (STOC'87)*, pages 132–140, May 1987.

[2] D.J. Aldous. Lower bounds for covering times for reversible Markov chains and random walks on graphs. *Journal of Theoretical Probability*, 2(1):91–100, 1989.

[3] D.J. Aldous. Reversible Markov chains and random walks on graphs. Draft, 1993.

[4] R. Aleliunas, R.M. Karp, R.J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proceedings 20th Symposium on Foundations of Computer Science (FOCS'79)*, pages 218–223, October 1979.

• [5] G. Barnes and U. Feige. Short random walks on graphs. In *Proceedings 25th ACM Symposium on Theory of Computing (STOC'93)*, pages 728–737, May 1993.

• [6] G. Barnes and U. Feige. Short random walks on graphs. Technical Report MPI-I-94-121, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, April 1994.

[7] A.Z. Broder and A.R. Karlin. Bounds on the cover time. *Journal of Theoretical Probability*, 2(1):101–120, 1989.

[8] A.Z. Broder, A.R. Karlin, P. Raghavan, and E. Upfal. Trading space for time in undirected s-t connectivity. In *Proceedings 21th ACM Symposium on Theory of Computing (STOC'89)*, pages 543–549, May 1989.

[9] M. Dyer, A. Frieze, and R. Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. In *Proceedings 21 ACM Symposium on Theory of Computing (STOC'89)*, pages 375–381, May 1989.

[10] U. Feige. A randomized time-space tradeoff of $\tilde{O}(m\hat{R})$ for USTCON. In *Proceedings 34th Symposium on Foundations of Computer Science (FOCS'93)*, pages 238–246, November 1993.

[11] O. Goldreich, R. Impagliazzo, L. Levin, R. Venkatesan, and D. Zuckerman. Security preserving amplification of hardness. In *Proceedings 31st Symposium on Foundations of Computer Science (FOCS'90)*, pages 318–326, October 1990.

[12] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings 30th Symposium on Foundations of Computer Science (FOCS'89)*, pages 248–253, October 1989.

[13] A. Israeli and M. Jalfon. Token management schemes and random walks yield self-stabilizing mutual exclusion. In *Proceedings 9th ACM Symposium on Principles of Distributed Computing (PODC'90)*, pages 119–131, 1990.

[14] M. Jerrum and A. Sinclair. Conductance and the rapid mixing property for Markov chains: The approximation of the permanent resolved. In *Proceedings 20th ACM Symposium on Theory of Computing (STOC'88)*, pages 235–244, May 1988.

[15] J.D. Kahn, N. Linial, N. Nisan, and M.E. Saks. On the cover time of random walks on graphs. *Journal of Theoretical Probability*, 2(1):121–128, 1989.

[16] D.R. Karger, N. Nisan, and M. Parnas. Fast connected components algorithms for the EREW PRAM. In *Proceedings ACM Symposium on Parallel Algorithms and Architectures (SPAA'92)*, pages 373–381, June 1992.

[17] N. Nisan, E. Szemerédi, and A. Wigderson. Undirected connectivity in $O(\log^{1.5} n)$ space. In *Proceedings 33rd Symposium on Foundations of Computer Science (FOCS'92)*, pages 24–29, October 1992.

[18] D.I. Zuckerman. On the time to traverse all edges of a graph. *Information Processing Letters*, 38(6):335–337, 1991.

3.3.10 Algorithms for Sparse Graphs and Networks

Investigators: Srinivasa Arikati, Shiva Chaudhuri, Torben Hagerup, Anil Maheshwari and Christos Zaroliagis

Many of the graphs or networks used in various applications have a particular sparse structure. Roughly speaking, this means that the graph or the network has a number of edges proportional to the number of vertices. Typical examples of sparse graphs, which are widely used in algorithms design, include trees, outerplanar graphs, series-parallel graphs, planar graphs, graphs of small genus and graphs of bounded treewidth.

Recently, a major part of our work has focussed on the efficient solution of combinatorial problems on sparse graphs and networks. More precisely, we have concentrated our effort on two important subclasses: (i) on the well-known class of planar graphs; and (ii) on the class of graphs having bounded treewidth. The precise definition of this latter class is technical and omitted here, but the important intuition is that graphs of small treewidth resemble trees in many ways (although not necessarily from a superficial inspection). The challenge therefore with the above classes is to exploit the planarity or the tree-like structure of the graph in a computational context, in order to obtain algorithms which are more efficient than the ones for general graphs (in both cases), or that are (almost) as efficient as the corresponding algorithms for trees (in the case of graphs with bounded treewidth). We have shown how to do this for some fundamental problems, including (among others) shortest paths and maximum flow. (The results for the latter problem are discussed in subsection 3.1.2.)

Shortest Paths

Finding shortest paths in directed graphs (digraphs) is perhaps the most fundamental (and definitely the simplest) problem in network optimization. Given an n-vertex, m-edge digraph G with real edge weights, the shortest paths problem asks for finding paths of minimum weight between vertices in G. There are two main versions of the problem. In the *single-source* or *shortest path tree* problem, we seek for shortest paths from a specific vertex to all other vertices in G. In the *all-pairs shortest paths* (apsp) problem, we seek for shortest paths between every pair of vertices in G. Due to its fundamental importance and wide applicability (see e.g. [1] for an extensive list of applications), both versions of the shortest paths problem have been extensively studied.

An apsp algorithm which outputs apsp information in the standard form (i.e., either in a table, or as n shortest path trees), requires $\Omega(n^2)$ time and space. A more efficient approach (which avoids this lower bound) is to preprocess the digraph so that subsequently, queries can be efficiently answered. A *query* specifies two vertices and a *shortest path query* asks for a minimum weight path between them, while a *distance query* only asks for the weight of such a path. Moreover, it is important that the data structures, created during preprocessing, can be updated efficiently (without recomputing everything from scratch) to reflect any change in edge weights. We refer to this problem as the *dynamic shortest paths* problem.

Our group has worked towards more efficient solutions of the apsp problem and its dynamic version. More precisely, we have considered two important subclasses of sparse digraphs, namely planar digraphs and digraphs of bounded treewidth.

The best previous result for the dynamic shortest paths problem in planar digraphs was due to Feuerstein and Spaccamela [7]. Their algorithm makes an $O(n \log n)$ preprocessing of the input digraph G and then answers a distance or a shortest path query in $O(n)$ time. The data structures set up during the preprocessing can be updated in $O(\log^3 n)$ time, after the modification of an edge weight. Based on the hammock decomposition technique (see subsection 3.1.2) which decomposes a planar digraph into $\tilde{\gamma}$ outerplanar graphs called hammocks ($1 \leq \tilde{\gamma} < n$), as well as on other techniques, we have given an algorithm for the dynamic shortest path problem [5]. A distance query is answered in $O(\log n + \tilde{\gamma})$ time and a shortest path one in $O(L)$ additional time (where L is the number of edges in the required path), after an $O(n + \tilde{\gamma} \log \tilde{\gamma})$ preprocessing of G. We can update the data structures set up during the preprocessing in $O(\log n + \log^3 \tilde{\gamma})$ time. Note that our algorithm performs very well when $\tilde{\gamma} = o(n)$ (i.e., in all cases where G has nice topological properties). In the case where G is outerplanar ($\tilde{\gamma} = 1$), the distance query and update time are logarithmic, and the preprocessing time is linear. Our data structures can also answer shortest path tree queries efficiently and our results can be extended to hold in digraphs with genus $o(n)$.

When the preprocessing for the apsp problem is restricted to be $O(n)$, efficient algorithms were known only for outerplanar digraphs [8] and graphs of treewidth 2 [3]; they answered distance queries in $O(\alpha(n))$[1] and $O(\log n)$ time respectively. We have very recently [4] given algorithms for the apsp problem that depend on the treewidth of the input digraph. When the treewidth is a constant, our algorithms can answer distance queries in $O(\alpha(n))$ time after $O(n)$ preprocessing. This improves upon the previous results since the class of constant treewidth graphs includes both the above classes of graphs. In the same paper [4], we also gave an algorithm for the dynamic shortest path problem for the class of digraphs of constant treewidth. A distance query is answered in $O(\alpha(n))$ time after $O(n)$ preprocessing. The algorithm updates the data structures, after a change in an edge weight, in time $O(n^{\beta})$, for any arbitrarily small constant $0 < \beta < 1$. (Shortest path queries are answered in time proportional to the number of the edges of the path.)

If the input digraph G has negative edge weights, then some of the shortest paths may not be defined due to cycles of negative weight [1], called *negative cycles*. The negative cycle problem is therefore fundamental to finding shortest paths in G. The problem is closely related to finding a shortest path tree, since most algorithms either construct a shortest path tree rooted at a given vertex, or find a negative cycle. But constructing a shortest path tree is often easier when the digraph has non-negative edge weights. In this latter case, the best algorithm for general digraphs takes $O(m + n \log n)$ time [9] to construct the shortest path tree. If the digraph has negative real edge weights, then one needs $O(nm)$ time to either construct a shortest path tree, or find a negative weight cycle (Bellman-Ford method [1]). The same happens in the case of planar digraphs. There is an $O(n)$ time algorithm for constructing a shortest path tree in a digraph with non-negative real edge weights [12], while a shortest path tree or a negative cycle can be found in $O(n^{1.5} \log n)$ time if the digraph has both positive and negative real edge weights [13].

We have worked towards closing the gap between the complexity of computing a shortest path tree in a digraph with non-negative real edge weights, and that of finding a negative cycle in a digraph with both positive and negative real edge weights. For the case of planar digraphs, we gave in [11] an algorithm which finds a negative cycle in $O(n + \tilde{\gamma}^{1.5} \log \tilde{\gamma})$ time and is based on the previously mentioned decomposition of the input planar digraph into $\tilde{\gamma}$ outerplanar graphs, called hammocks. (In the case that there is no negative cycle, the algorithm can be easily modified to output the shortest path tree as well.) For the case of digraphs with constant treewidth, we have very recently given [4] an $O(n)$ time algorithm for either constructing a shortest path tree or finding a negative cycle. To the best of our knowledge, this is the most general class of graphs for which the complexity of computing a shortest path tree matches that of finding a negative cycle.

We plan to work more along the above directions, especially on finding faster solutions for the dynamic shortest paths problem in planar digraphs.

Compact Representation of Sparse Graphs

A fundamental data structuring question in the design of efficient algorithms, is how to represent a graph in memory using as little space as possible, so that given any two vertices we can test if they are adjacent in $O(1)$ time [10, 15]. The well-known adjacency matrix representation permits adjacency queries in $O(1)$ time, but it requires $\Theta(n^2)$ space even for the case when the input graph is sparse (i.e., it has a linear number of edges). Another characterization of sparse graphs is given by the *arboricity*. The arboricity of a graph G is defined as $\max_H\{m_H/(n_H - 1)\}$, where H is an n_H-vertex, m_H-edge subgraph of G. (For example, planar graphs have arboricity 3.) Now sparse graphs are the graphs of bounded arboricity. It

[1] $\alpha(n)$ is the inverse of Ackermann's function and is a very slowly growing function.

follows by a theorem of Nash-Williams [10], that a graph with arboricity c can be compactly represented in memory using only $(c+1)n$ space. In such a case, G is said to have an *optimal compact or implicit representation.* The known algorithms for obtaining an optimal implicit representation of a sparse graph ran in $O(n^4)$ time and were based on involved techniques such as Edmonds' results on matroid partitioning [6]. Also, the results of Schnyder [14], imply an $O(n)$ time algorithm for the optimal implicit representation of planar graphs.

We achieved the goal of computing an optimal implicit representation of a sparse graph, by giving a very simple and optimal $O(n)$ time algorithm for this problem [2]. Furthermore, since computing the exact value of the arboricity seems to be hard [15], we have given an efficient algorithm for computing a 2-approximate value for arboricity. Surprisingly enough, we have also shown [2] that using this approximate value, we can still obtain an optimal implicit representation of a sparse graph.

We plan to continue work in this direction. In particular, we would like to investigate the case where the input graph may change dynamically.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows.* Prentice-Hall, 1993.

• [2] S. Arikati, A. Maheshwari, and C. Zaroliagis. Saving bits made easy. In *Proceedings 6th Canadian Conference on Computational Geometry (CCCG'94)*, pages 140–146, August 1994. Also Technical Report MPI-I-94-148, Max-Planck-Institut für Informatik, 1994.

[3] H. Bodlaender. Dynamic algorithms for graphs with treewidth 2. In *Proceedings 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'93)*, pages 112–124. LNCS 790, Springer-Verlag, 1994.

• [4] S. Chaudhuri and C. Zaroliagis. Shortest path queries in digraphs of small treewidth. In *Proc. 22nd Int'l Colloquium on Automata, Languages and Programming (ICALP'95), to appear.* LNCS, Springer-Verlag, 1995.

• [5] H. Djidjev, G. Pantziou, and C. Zaroliagis. On-line and dynamic algorithms for shortest path problems. In *Proc. 12th Symp. on Theoretical Aspects of Computer Science (STACS' 95), to appear.* LNCS, Springer-Verlag, 1995. Also Tech. Rep. MPI-I-94-114, 1994.

[6] J. Edmonds. Minimum partition of a matroid into independent sets. *Research of the NBS*, 69B:67–72, 1965.

[7] E. Feuerstein and A.M. Spaccamela. Dynamic algorithms for shortest paths in planar graphs. *Theoretical Computer Science*, 116:359–371, 1993.

[8] G. Frederickson. Using cellular graph embeddings in solving all pairs shortest path problems. accepted in *Journal of Algorithms*, 1994.

[9] M. Fredman and R. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.

[10] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. In *Proceedings 20th ACM Symposium on Theory of Computing (STOC'88)*, pages 334–343, May 1988.

• [11] D. Kavvadias, G. Pantziou, P. Spirakis, and C. Zaroliagis. Efficient sequential and parallel algorithms for the negative cycle problem. In *Proceedings 5th International Symposium on Algorithms and Computation (ISAAC'94)*, pages 270–278. LNCS 834, Springer-Verlag, August 1994.

[12] P. Klein, S. Rao, M. Rauch, and S. Subramanian. Faster shortest-path algorithms for planar graphs. In *Proceedings 26th ACM Symposium on Theory of Computing (STOC'94)*, pages 27–37, May 1994.

[13] K. Mehlhorn and B. Schmidt. A single source shortest path algorithm for graphs with separators. In *Proceedings Fundamentals of Computation Theory (FCT'83)*, pages 302–309. LNCS 158, Springer-Verlag, 1983.

[14] W. Schnyder. Embedding planar graphs on the grid. In *Proceedings 1st ACM-SIAM Symposium on Discrete Algorithms (SODA'90)*, January 1990.

[15] J. van Leeuwen. Graph algorithms. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A*, chapter 10, pages 525–631. Elsevier, 1990.

3.3.11 Approximation Algorithms for NP-hard Problems

Investigators: Naveen Garg, Ramesh Hariharan and Sanjeev Mahajan

Since most real world combinatorial optimization problems are NP-hard and it is unlikely that they could be solvable in polynomial time, researchers in the past have attempted to find polynomial time algorithms that guarantee a solution with value that is within a multiplicative factor of α of the value of the optimal solution. The factor α is variously referred to as the *performance guarantee* or the *approximation bound* of the algorithm and an algorithm with performance guarantee α is called an *α-approximation* algorithm.

One approach to NP-hard problems involving cuts in graphs has been to define a multicommodity flow problem such that the maximum flow is a good lower bound on the value of the minimum cut. A fundamental problem in this setting is regarding the *feasibility of flow* – Given a multicommodity flow problem, in which each commodity has an associated demand (the amount of that commodity which we wish to ship), one often needs to know if there is a *feasible flow*, i.e., a flow that satisfies the demands and obeys the capacity constraints. For a feasible flow to exist it is necessary that the capacity of any cut exceed the sum of the demands that are separated by the cut. A natural question to ask is how large a "safety margin" is needed, i.e., by what factor should the capacity of a cut exceed the total demand separated by the cut, in order to ensure existence of a feasible flow.

The optimization version of the feasibility problem for multicommodity flow is to find the maximum f such that a fraction f of each demand can be routed concurrently. The fraction f is called the *maximum concurrent flow*; the problem is feasible if $f \geq 1$. Define the *sparsest cut* to be the cut that minimizes the ratio of the total capacity of the cut to the sum of the demands separated by the cut. It is easy to see that f cannot exceed the sparsest cut ratio and in [5] it was proved that f is at least as large as the sparsest cut ratio times $\Omega(\frac{1}{\log k})$. This approximate min-max relation implies that an $O(\log k)$ safety margin is sufficient for flow routability. The proof of this theorem also yields a randomised $O(\log k)$-approximation algorithm for the sparsest cut in the graph (computing the sparsest cut is NP-hard). In [2] we provided an alternate proof of this fact. Our proof is conceptually similar to that of [5] but has the advantage that it yields a deterministic $O(\log k)$-approximation algorithm for the sparsest cut in a graph.

Leighton and Rao [4] showed how an α-approximation algorithm for the sparsest cut can be used to obtain a separator of weight at most $O(\alpha \log n)$ times the weight of the best bisection. These ideas combined with the algorithms in [1] produce polylogarithmic approximations for a host of VLSI related problems.

Another approach to NP-hard problems involves semidefinite programming. In [3], Goemans and Williamson use semidefinite programming to design randomized algorithms in order to give approximation algorithms for NP-hard problems such as Max-Cut, Max-Dicut and Max-SAT. These algorithms achieve the best known approximation ratios. They then give a general method of derandomizing these algorithms. In [6], we first point out a flaw in their derandomization scheme and then show how to correctly derandomize their randomized algorithms.

References

[1] S.N. Bhatt and F.T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, 1984.

• [2] N. Garg. A deterministic $O(\log k)$-approximation algorithm for the sparsest cut. Manuscript, 1995.

[3] M. Goemans and D. Williamson. 0.878 approximation algorithms for max cut and max sat. In *Proc. 26th ACM Symposium on Theory of Computing (STOC'94)*, pages 422–431, May 1994.

[4] F.T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with application to approximation algorithms. In *Proceedings 29th IEEE Symposium on Foundations of Computer Science (FOCS'88)*, pages 422–431, 1988.

[5] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. In *Proceedings 35th IEEE Symposium on Foundations of Computer Science (FOCS'94)*, pages 577–591, 1994.

• [6] S. Mahajan and R. Hariharan. Correctly derandomizing the 0.878 approximation algorithm for max cut. Manuscript, 1995.

3.3.12 Computation of Exact Ground States of Ising Spin Glasses

Investigator: Petra Mutzel

The computation of exact ground states of Ising spin glasses is one of the challenging problems in Statistical Physics. The problem for the 2-dimensional standard spin glass model with periodic boundary conditions and zero magnetic field can be formulated as a max-cut problem on toroidal grid graphs. The magnetic field can be modeled as a supernode added to the toroidal grid graph, where the supernode is adjacent to all the grid nodes [3]. When there is no external magnetic field and all nearest neighbor interactions are $+J$ or $-J$, the problem is solvable in polynomial time [1]. If we have general (e.g. normally distributed) interactions and no field, the complexity status is open. The problem gets NP-hard as soon as a non-zero magnetic field occurs [1]. We focused our research on two different approaches.

One idea was to try to extend the combinatorial max-cut algorithm on planar graphs to toroidal grid graphs. For a planar graph G, a maximum cut can be found by solving a minimum-weight perfect matching problem in a special graph constructed from G [10, 11]. The max-cut problem on toroidal graphs can be transformed into a similar minimum matching problem with side constraints. These side-constraints are certain evenness conditions on the rows and columns of the grid and can be tested easily.

Unfortunately, we have not yet succeeded in solving this restricted matching problem. Since the interest in computing the exact ground states is only statistical, we proposed to solve the unrestricted minimum-weight perfect matching problem. If the evenness conditions are satisfied (this can be checked easily in $O(k+l)$ time for a $k \times l$ grid), the maximum cut is found. Otherwise a different instance is generated.

Our experimental results confirm our conjecture that about one quarter of all runs are successful for instances with normally distributed nearest neighbor interactions, and that this sample is statistically meaningful [7].

Our second approach was to use a branch and cut algorithm in order to compute exact ground states of Ising spin glasses with and without a magnetic field. The facial structure of the polyhedra associated with all cuts of a given complete graph has already been studied by many researchers [8, 5]. However, there are only a few papers investigating the polyhedral structure for general graphs [4].

We investigated the facial structure of the polytope associated with all cuts occurring in the toroidal grid graph $G_{k\times l}$. In addition to the trivial inequalities and the cycle inequalities, special subdivisions of K_5 are facet-defining inequalities for the associated cut polyhedra. These subdivisions of K_5 which are obtained by using the node-splitting operation given in [4] are embeddable on $G_{k\times l}$ and can be zero-lifted to yield facets for the cut polytope associated with the toroidal grid graph $G_{k\times l}$.

Using the branch and cut paradigm is, up to now, the only way to compute exact ground states of 2-dimensional spin glasses of reasonable sizes. Previous branch and cut algorithms have been able to solve 2-dimensional spin glass instances with periodic boundary conditions and general (normally distributed) interactions for instances of sizes up to 30×30 [9], and one 35×35 instance [2].

In our newest computational experiments we have been able to solve instances with and without magnetic field of sizes up to 100×100 in a reasonable amount of time [6].

References

[1] F. Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical Gen.*, 36:3241–3253, 1982.

[2] F. Barahona. Ground-state magnetization of Ising spin glasses. *Physical Review B*, 49:12864, 1994.

[3] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36:493–513, 1988.

[4] F. Barahona and A.R. Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.

[5] C. De Simone. *The Max Cut Problem.* PhD thesis, Rutgers University, 1991.

• [6] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi. Exact ground states of Ising spin glasses: New experimental results with a branch and cut algorithm. to appear in the *Journal of Statistical Physics*, 1995.

• [7] C. De Simone, M. Jünger, P. Mutzel, and G. Rinaldi. An approach to a combinatorial algorithm for max cut on toroidal grid graphs. Extended Abstract, 1994.

[8] M. Deza and M. Laurent. Facets for the cut cone I/II. *Mathematical Programming*, 56:121–160, 161–181, 1992.

[9] M. Grötschel, M. Jünger, and G. Reinelt. Calculating exact ground states of spin glasses: A polyhedral approach. In J.L. van Hemmen and J. Morgenstern, editors, *Proceedings of the Heidelberg Colloquium on Glassy Dynamics, Lecture Notes in Physics 275*, pages 325–353. Springer-Verlag, 1987.

[10] F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4:221–225, 1975.

[11] G.I. Orlova and Y.G. Dorfman. Finding the maximal cut in a graph. *Engrg. Cybernetics*, 10:502–506, 1972.

3.4 Complexity Theory

3.4.1 Directed s-t Connectivity

Investigator: Greg Barnes

The s-t connectivity problem is a fundamental one in computational complexity theory [17]. The s-t connectivity problem for *directed* graphs (STCON) is the prototypical complete problem for nondeterministic logarithmic space [14]. Both STCON and the corresponding problem for undirected graphs, USTCON, are DLOG-hard — any problem solvable deterministically in logarithmic space can be reduced to either problem [11, 14]. Understanding the complexity of s-t connectivity is, therefore, a key to understanding the relationship between deterministic and nondeterministic space bounded complexity classes. For example, showing that there is no deterministic logarithmic space algorithm for directed connectivity would separate the classes DSPACE($\log n$) and NSPACE($\log n$), while devising such an algorithm would prove that DSPACE($f(n)$) = NSPACE($f(n)$) for any constructible $f(n) = \Omega(\log(n))$ [14]. Unfortunately, determining the complexity of STCON remains a difficult open problem. A fruitful intermediate step is to explore *time-space tradeoffs* for STCON; that is, the *simultaneous* time and space requirements of algorithms for directed connectivity.

Proving lower bounds on the time or space requirements of STCON for a general model of computation, such as a Turing machine, is beyond the reach of current techniques. Thus, it is natural to consider a *structured* model [5] whose basic operations are based on the structure of the graph, as opposed to being based on the bits in the graph's encoding. A natural structured model for the problem of s-t connectivity is the "jumping automaton for graphs", or *JAG*, introduced by Cook and Rackoff [8]. A JAG moves a set of pebbles on the graph. There are two basic operations — moving a pebble along a directed edge in the graph, and jumping a pebble from its current location to the vertex occupied by another pebble. Although the JAG model is structured, it is not weak. In particular, it is general enough that most known deterministic algorithms for graph connectivity can be implemented on it. Poon [12] introduced the more powerful node-named JAG (*NNJAG*), an extension of the JAG model where the computation is allowed to depend on the names of the nodes on which the pebbles are located.

Cook and Rackoff [8] proved a lower bound of $\Omega\left(\log^2 n/\log\log n\right)$ on the space required for a JAG to compute directed s-t connectivity (STCON). Berman and Simon [4] extended this result to randomized JAGs, and Poon [12] extended it to a probabilistic version of the NNJAG. Tompa [16] showed lower bounds on the product of the time and space needed when using certain natural approaches to solve STCON. Many time-space lower bounds have been proved for *undirected* s-t connectivity on various weak versions of the JAG model [3, 7, 8]. Edmonds was the first to prove a time-space lower bound for USTCON on the unrestricted JAG model [9].

The standard algorithms for s-t connectivity, breadth- and depth-first search, run in optimal time $\Theta(m+n)$ and use $\Theta(n \log n)$ space. At the other extreme, Savitch's Theorem [14] provides a small space ($\Theta(\log^2 n)$) algorithm that requires time exponential in its space bound (i.e., time $n^{\Theta(\log n)}$). Barnes *et al.* [1] showed the first sublinear space, polynomial time algorithm for STCON. All of these algorithms can be implemented on the standard JAG [8, 13]. Using the NNJAG's ability to access the names of the nodes in the graph, Poon [12] showed how to implement Immerman's and Szelepcsényi's nondeterministic $O(\log n)$-space algorithm for directed *s-t* nonconnectivity [10, 15] on a nondeterministic NNJAG. It is not clear that this algorithm can be implemented on a standard nondeterministic JAG.

In [2], we could prove lower bounds of $ST = \Omega(n^2/\log n)$ and $S^{1/2}T = \Omega(mn^{1/2})$ for STCON on the JAG model, and of $S^{1/3}T = \Omega(m^{2/3}n^{2/3})$ on the more powerful Node-Named JAG model, where S is the space and T the time used by the JAG, This last bound is proved on probabilistic NNJAGs by transforming the machine into a structured branching program, and following the framework introduced by Borodin *et al.* [6]. These lower bounds approach the known upper bound of $T = O(m)$ when $S = \Theta(n \log n)$, and are the first time-space tradeoff on JAGs with an unrestricted number of jumping pebbles.

References

- [1] G. Barnes, J.F. Buss, W.L. Ruzzo, and B. Schieber. A sublinear space, polynomial time algorithm for directed *s-t* connectivity. In *Proceedings 7th Structure in Complexity Theory*, pages 27–33, June 1992. Submitted for publication.

- [2] G. Barnes and J.A. Edmonds. Time-space lower bounds for directed *s-t* connectivity on JAG models. Technical Report MPI-I-94-119, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, April 1994.

[3] P.W. Beame, A. Borodin, P. Raghavan, W.L. Ruzzo, and M. Tompa. Time-space tradeoffs for undirected graph connectivity. In *Proceedings 31st Symposium on Foundations of Computer Science (FOCS'90)*, pages 429–438, October 1990. Full version: University of Washington Department of Computer Science and Engineering Technical Report 93-02-01, 44 pages; submitted for publication.

[4] P. Berman and J. Simon. Lower bounds on graph threading by probabilistic machines. In *Proceedings 24th Symposium on Foundations of Computer Science (FOCS'83)*, pages 304–311, November 1983.

[5] A. Borodin. Structured *vs.* general models in computational complexity. *L'Enseignement Mathématique*, XXVIII(3-4):171–190, 1982.

[6] A. Borodin, M.J. Fischer, D.G. Kirkpatrick, N.A. Lynch, and M. Tompa. A time-space tradeoff for sorting on non-oblivious machines. *Journal of Computer and System Sciences*, 22(3):351–364, 1981.

[7] A. Borodin, W.L. Ruzzo, and M. Tompa. Lower bounds on the length of universal traversal sequences. *Journal of Computer and System Sciences*, 45(2):180–203, 1992.

[8] S.A. Cook and C.W. Rackoff. Space lower bounds for maze threadability on restricted machines. *SIAM Journal on Computing*, 9(3):636–652, 1980.

[9] J.A. Edmonds. Time-space trade-offs for undirected *ST*-connectivity on a JAG. In *Proceedings 25th ACM Symposium on Theory of Computing (STOC'93)*, pages 718–727, May 1993.

[10] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, 1988.

[11] H.R. Lewis and C.H. Papadimitriou. Symmetric space-bounded computation. *Theoretical Computer Science*, 19(2):161–187, 1982.

[12] C.K. Poon. Space bounds for graph connectivity problems on node-named JAGs and node-oriented JAGs. In *Proceedings 34th Symposium on Foundations of Computer Science (FOCS'93)*, November 1993.

[13] C.K. Poon. A sublinear space, polynomial time algorithm for directed *st*-connectivity on the JAG model. Manuscript, 1993.

[14] W.J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[15] R. Szelepcsényi. The method of forcing for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[16] M. Tompa. Two familiar transitive closure algorithms which admit no polynomial time, sublinear space implementations. *SIAM Journal on Computing*, 11(1):130–137, 1982.

[17] A. Wigderson. The complexity of graph connectivity. In *Proceedings 17th Symposium on Mathematical Foundations of Computer Science (MFCS'92)*, pages 112–132. LNCS 629, Springer-Verlag, 1992.

3.4.2 The Additive Fragment of Linear Logic and $\mathcal{NC}^1$

Investigator: Phil Bradford

Since Girard's introduction of linear logic in 1987, there have been a number of papers about the computational complexity of derivability in fragments of linear logic. Lincoln *et al.* [6] showed that provability in the multiplicative-additive fragment of linear logic is $\mathcal{P}$SPACE-complete. In [1] we have shown that derivability in the additive fragment of linear logic is $\mathcal{NC}^1$-complete under D-logtime (hence $\mathcal{AC}^0$) reductions. From here on, "$\mathcal{NC}^1$" always means D-logtime uniform $\mathcal{NC}^1$ or equivalently log-time bounded alternating Turing machines; see [7]. Our work builds on Buss's result that the Boolean sentence value problem is $\mathcal{NC}^1$-complete.

Our result may be interpreted in a number of ways, since it connects two different enterprises. In the literature on the complexity of fragments of linear logic, our results can be seen in contrast with Kanovich's results showing the multiplicative fragment of linear logic is $\mathcal{NP}$-complete [5]. Further, Lincoln *et al.* showed that the additive and multiplicative fragments *together* capture $\mathcal{P}$SPACE.

Our work shows how the additive fragment of linear logic embodies the notion of alternation in both $\mathcal{NC}$ and $\mathcal{P}$SPACE. (That is, the additive fragment of linear logic embodies a classical notion of parallelism.) In fact, the connection is more pervasive than just the results of this paper. Consider Lincoln *et al.*'s proof that the multiplicative-additive fragment is $\mathcal{P}$SPACE-complete [6]. The proof actually shows that the degree of alternation of the additive connectives corresponds to the levels of the polynomial hierarchy.

The class $\mathcal{NC}^1$ includes problems with fast parallel solutions. It is in $\mathcal{P}$TIME, and many interesting problems have been shown to be in $\mathcal{NC}^1$, including parsing context-free grammars, multiplication, and sorting [4, 8]. Buss [2, 3] and Buss *et al.* [4] showed that the Boolean sentence value problem is $\mathcal{NC}^1$-complete. In the roughest terms, these papers provided an algorithm for the Boolean sentence value problem which uses a polynomial number of independent processors, each doing a very small (logarithmic) amount of work. Therefore these results show that a simple and eminently important logical question, that of evaluating Boolean sentences, is in $\mathcal{NC}^1$. Our work may be regarded as an application of Buss's techniques to proof theory: we show that the derivability problem for sequents in the additive fragment of linear logic is also $\mathcal{NC}^1$-complete. To do this, we generalize Buss's pebbling games from single trees to *pairs of trees*. We hope that our generalization of these pebbling games suggests other extensions of the game-theoretic perspective.

References

• [1] Phillip G. Braford, Jean-Yves Marion, and Lawrence S. Moss. The additive fragment of linear logic is $\mathcal{NC}^1$-complete. In *To appear in the Proceedings of the International Conference in Logic and Computational Complexity*, 1995.

[2] S. R. Buss. The boolean formula value problem is in ALOGTIME. In *Symposium on the Theory of Computing (STOC)*, pages 123–131, 1987.

[3] S. R. Buss. Algorithms for boolean formula evaluation and for tree contraction. In P. Clote and J. Krajicek, editors, *Arithmetic, Proof Theory, and Computational Complexity*, pages 96–115. Oxford Logic Guides, 1993.

[4] S. R. Buss, S. Cook, A. Gupta, and V. Ramachandran. An optimal parallel algorithm for formula evaluation. *SIAM J. on Comp.*, 21:755–780, 1992.

[5] M. I. Kanovich. Horn programming in linear logic is $\mathcal{NP}$-complete. In 7^{th} *Logic in Computer Science Conference (LICS)*, pages 200–210, 1992.

[6] P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. *Annals Pure Appl. Logic*, 56:239–311, 1990.

[7] W. Ruzzo. On uniform circuit complexity. *J. of Computer and System Sci.*, 22:365–383, 1981.

[8] J. E. Savage. *The Complexity of Computing.* Wiley, 1976.

3.4.3 Lower Bounds on Decision Trees

Investigator: Rudolf Fleischer

Among other algebraic complexity measures, the algebraic decision tree and algebraic computation tree models have turned out to be very useful in proving lower bounds for elementary combinatorial or geometric problems like maximum finding, set equality, set disjointness and sorting (see [1] for more examples) or even more complicated problems like convex polygon inclusion [10] and motion planning [8].

The algebraic decision tree model is an abstraction of "real" algorithms where only comparisons between input variables or functions of input variables are counted whereas all other time-consuming operations like data-management, function evaluation or other control structures have zero cost. For complex problems, this simplification can make the problem considerably easier; for example, the knapsack problem which is known to be $\mathcal{NP}$-complete has a polynomial solution in the decision tree model [7]. Therefore, lower bounds in the decision tree model can only be tight for quite simple problems.

A *decision problem* is a partition of $\mathbb{R}^n$ into sets $S_1, S_2, \cdots, S_q$. A *decision tree* T for a decision problem is a binary tree in which internal nodes are labeled by predicates defined on $\mathbb{R}^n$, outgoing edges of an internal node are labeled by *true* or *false*, and leaves are labeled by one of the S_i. The evaluation of T on input $x \in \mathbb{R}^n$ starts at the root and then proceeds downwards by evaluating the predicate at an internal node and taking the appropriate of the two outgoing edges. Finally, a leaf with label S_x is reached. S_x is the result of the computation, the path x followed is the *computation path* of x, and T is correct if $x \in S_x$ for all x. The worst-case running time of T is the length of the longest computation path in T.

T is called an *algebraic decision tree* if all functions evaluated at internal nodes are defined by polynomials. The most restricted algebraic decision trees are *comparison trees* where only comparisons between two input variables are allowed [5]. *Linear decision trees* where linear functions of the input variables can be used [2, 5, 11] are more powerful. Products of linear functions were used in [13] and arbitrary polynomials of bounded degree in [1] and [14]. Finally, arbitrary analytic functions were allowed in [6] and [9]. Of course, this classification of algebraic decision trees is not exhaustive and many other restrictions on the functions can be found in the literature.

We assume that the decision problem is a membership problem, i.e. we want to decide whether an input $x \in \mathbb{R}^n$ is in a set $S \subseteq \mathbb{R}^n$ (we call S the *target set*), but the lower bounds mentioned below can easily be transformed into similar bounds for arbitrary decision problems.

Whereas many different lower bound techniques for restricted decision trees are known [1, 2, 5, 10, 14], the only known results about analytic decision trees are due to Rabin [9] and Jaromczyk [6]. Rabin proved the fundamental Theorem that any analytic decision tree for S must have depth $|H|$ if S is defined by a set H of independent linear inequalities. Jaromczyk generalized Rabin's Theorem to sets S defined by arbitrary polynomial inequalities.

In [4] we gave an alternative proof of Rabin's Theorem by using a new lower bound technique which we had developed to solve an open question raised by Yao. In [14], Yao showed that median tests are not really more powerful than simple comparisons between the input variables when computing the largest k elements of n given numbers. He asked whether this can be generalized to functions which are arbitrary products of linear functions (the median test can be written as the product of two linear functions).

We showed that it can be generalized. Our proof technique is based on a dimension argument and works only for sets S defined by linear inequalities. Let $rank(S)$ be the maximal dimension of a linear subspace contained in the closure of S. We show that, for any computation path p in the decision tree, the closure of the set of inputs x which have computation path p always contains a linear subspace of dimension $n - length(p)$. Hence $length(p) \geq n - rank(S)$.

It seems to be the first time that a dimension argument is used to derive good lower bounds for nonlinear decision trees.

We would like to point out that we learned after the publication of our results that a similar proof of Rabin's Theorem had been obtained earlier by Pardo and Tomás [12], and that one can find even more general approaches in the mathematics literature [3].

References

[1] M. Ben-Or. Lower bounds for algebraic computation trees. In *Proc. 15th Symp. on Theory of Computing (STOC'83)*, pages 80–86, 1983.

[2] A. Björner, L. Lovász, and A.C. Yao. Linear decision trees : Volume estimates and topological bounds. In *Proc. 24th Symp. on Theory of Computing (STOC'92)*, pages 170–177, 1992.

[3] J. Bochnak, M. Coste, and M.F. Roy. *Géométrie Algébrique Réelle, Ergebnisse der Mathematik und ihrer Grenzgebiete, Folge 3, Bd. 12*. Springer Verlag, 1987.

• [4] R. Fleischer. Decision trees : Old and new results. In *Proc. 25th Symp. on Theory of Computing (STOC'93)*, pages 468–477, May 1993. Also Technical Report MPI-I-92-125.

[5] F. Fussenegger and H.N. Gabow. A counting approach to lower bounds for selection problems. *J. of the ACM*, 26:227–238, 1979.

[6] J.W. Jaromczyk. Lower bounds for problems defined by polynomial inequalities. In *Symp. on Foundations of Computing Theory (FCT'81)*, pages 165–172. LNCS 117, Springer Verlag, 1981.

[7] F. Meyer auf der Heide. A polynomial linear search algorithm for the n-dimensional knapsack problem. *J. of the ACM*, 31(3):668–667, 1984.

[8] C. O'Dúnlaing. A tight lower bound for he complexity of path-planning for a disc. *Information Processing Letters*, 28(4):165–170, 1988.

[9] M. Rabin. Proving simultaneous positivity of linear forms. *J. on Computer System Sciences*, 6:639–650, 1972.

[10] P. Ramanan. Obtaining lower bounds using artificial components. *Information Processing Letters*, 24(4):243–246, 1987.

[11] M. Snir. Comparisons between linear functions can help. *J. on Theoretical Computer Science*, 19:321–330, 1982.

[12] R. Tomás and M. Pardo. Rabin's width of a complete proof and the width of a semialgebraic set. In *European Conference on Computer Algebra*, pages 133–139. LNCS 378, Springer Verlag, June 1987.

[13] A.C. Yao. On selecting the k largest with median tests. *Algorithmica*, 4(2):293–300, 1989.

[14] A.C. Yao. Algebraic decision trees and euler characteristics. In *Proc. 33th Conf. on the Foundations of Computer Science (FOCS'92)*, pages 268–277, 1992.

3.4.4 Circuit Complexity

Investigator: Shiva Chaudhuri

A fundamental goal of circuit complexity theory is to obtain bounds on the resources required to solve various problems. In spite of the importance of the problem, obtaining bounds on the resources required to solve most natural problems remains an open problem. The best size lower bound for a natural problem, PARITY, is $4n - 4$. In order to understand the nature of various circuit resources, various restrictions of the model have been studied, with some success. Bounded depth circuits, in particular, have been closely studied. The class AC^0 is the set of functions computable by uniform polynomial size circuits of constant depth. The class LC^0 is the set of functions computable by uniform linear size circuits of constant depth [6, 7]. While LC^0 appears to be a severely restricted class, it turns out to contain surprisingly complex functions. It was not even known whether LC^0 is properly contained in AC^0 [6].

In [2], we answer this question in the affirmative. An 1/4-*approximate selector* is any function whose value is 0 if the number of 1's in the input is less than $n/4$, 1 if the number of 1's is more than $3n/4$ and can be either 0 or 1 otherwise. Such a function provides a rough estimate of the number of 1's and is extremely useful in parallel computation [6, 3]. Recently, Ajtai gave a uniform construction of an AC^0 circuit to compute such a function [1]. Our main result gives a tight, superlinear lower bound on the number of gates in any constant depth circuit that computes such a function. Thus, no approximate selector function is in LC^0. This shows that LC^0 is strictly contained in AC^0.

We actually prove a size-depth tradeoff for a general class of functions. The *robustness* of a function is essentially the maximum number of input bits that can always be revealed to an adversary without revealing the value of the function. For example, the robustness of PARITY is $n - 1$, of MAJORITY, $\lfloor n/2 \rfloor$. A λ-approximate selector is any function that satisfies: $f : \{0,1\}^n \longrightarrow \{0,1\}$ such that for $x \in \{0,1\}^n$, $s(x) < (1/2 - \lambda)n \Rightarrow f(x) = 0$ and $s(x) < (1/2 + \lambda)n \Rightarrow f(x) = 1$, where $s(x)$ is defined to be the number of 1's in x. The robustness of a λ-approximate selector is at least $n/2 - \lambda n$. We show that if a circuit of depth k with S gates computes a function of robustness R, then $S = \Omega(R^{1+\epsilon(k)})$, where $\epsilon(k) = 1/2^{2k+1}$. This implies that a linear size circuit that computes a 1/4-approximate selector has depth

$\Omega(\log\log n)$. The lower bound is optimal upto the value of $\epsilon(k)$. We give uniform constructions of circuits of depth $O(k)$, computing a 1/4-approximate selector, with $O(n^{1+\delta(k)})$ gates where $\delta(k) = 1/2^k$, which imply a linear size circuit of depth $O(\log\log n)$.

While the number of gates has been extensively studied as a resource in circuit complexity, much less attention has been given to wires and none to the trade-off between gates and wires. We study this tradeoff, via small threshold functions. The threshold function T_k^n assumes the value 1 iff at least k of its input bits have value 1.

It follows from the lower bounds in [4], that T_k^n is in AC^0 iff $(\log k)/\log\log n = \lfloor d \rfloor$ for some constant d. It also follows that a circuit of depth d for T_k^n must have at least $\exp 2(\Omega(k^{1/d}))$ gates. In [5] depth d circuits for T_k^n are given which use $\exp 2(O(k \log\log n)^{1/d})$ gates and $O(n(k\log n)^2)$ wires. The lower bound of [4] shows that the number of gates is close to optimal, but gives no information about the number of wires. In particular, can we achieve a circuit that has a linear number of wires? We give a partial answer to this question, by showing that any circuit computing T_k^n which has at most $n/4$ gates must have $kn/2$ wires. This bound holds for circuits of any depth, and is one of very few bounds for unbounded depth circuits. Thus, if k is not a constant, it is not possible to simultaneously achieve a sublinear number of gates and a linear number of wires. This bound is the best possible for unrestricted depth circuits, since T_k^n can be computed by a circuit of depth $O(\log n)$ with $O(n)$ gates and wires.

We conjecture a bound of the form $\Omega(nf(k))$ on the number of wires in a constant depth circuit computing T_k^n, where $f(k) \longrightarrow \infty$. We are working on this problem and have been able to prove some partial results in this direction.

References

[1] M. Ajtai. Approximate counting with uniform constant depth circuits. In *DIMACS Series in Disc. Math. and Theoret. Comp. Sci., American Math. Society*, pages 1–20, 1993.

• [2] S. Chaudhuri. $LC^0 \subset AC^0$: Polynomial is strictly better than linear. Submitted, 1994.

[3] T. Goldberg and U. Zwick. Optimal deterministic approximate parallel prefix sums and their applications. Submitted, 1994.

[4] J. Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings 18th Symp. Theory of Computation*, pages 6–20, 1986.

[5] J. Håstad, I. Wegener, N. Wurm, and S-Z. Yi. Optimal depth, very small size circuits for symmetric functions in AC^0. *Information and Computation*, 108:200–211, 1994.

[6] I. Newman, P. Ragde, and A. Wigderson. Perfect hashing, graph entropy and circuit complexity. In *Proceedings 5th Ann. Conf. on Structure in Complexity Theory*, pages 91–99, 1990.

[7] P. Ragde and A. Wigderson. Linear-size constant-depth polylog-threshold circuits. *Information Processing Letters*, 39:143–146, 1991.

3.5 The LEDA Platform

The LEDA project, started in 1988, has already been described in the previous Progress Report (November 1993). A more detailed description can be either found in that report, or via World Wide Web in `http://www.mpi-sb.mpg.de/LEDA/leda.html`. Here we only give a short overview of LEDA and describe its new feature concerning correct implementations of some basic geometric algorithms by handling all degeneracies and avoiding the dangers of inexact arithmetic.

3.5.1 Overview

Investigators: Kurt Mehlhorn, Stefan Näher and Christian Uhrig

Combinatorial and geometric computing rely heavily on data types like stacks, queues, dictionaries, sorted sequences, priority queues, graphs, points, segments, etc. LEDA (Library of Efficient Data types and Algorithms) is a library that provides such data types and algorithms. Some of its main features are:

- LEDA provides most of the data types and algorithms described in text books in the area and makes them easy to use for non-experts.
- LEDA provides very efficient implementations for each of the data types. It provides a mechanism which allows the user to choose among different implementations for the same data type.
- LEDA contains a comfortable data type graph that supports the implementation of graph algorithms in a form close to the typical text book representation.

A significant part of LEDA is now so mature that several hundred sites (including universities and industrial software companies all over the world) have made it the basis of algorithms and serious software development. For instance, there is an intensive cooperation with Siemens AG [5, 6]. Moreover, it is no longer the mission of a research institute to maintain it. Kurt Mehlhorn, Stefan Näher, and Christian Uhrig have founded LEDA GmbH which (after a license agreement with the Max-Planck-Gesellschaft) plans to distribute, maintain and further develop LEDA. Also, a verification group at the DFKI (Deutsches Forschungszentrum für Künstliche Intelligenz) plans to verify several parts of the library.

The main concepts of LEDA are described in [9]. The user manual [11] lists the specifications of all data types and algorithms contained in the current version 3.1 of the library and gives many example programs.

Kurt Mehlhorn and Stefan Näher are writing a book about LEDA that will contain a tutorial and provide many details about the design and the implementation of LEDA.

Current and future work (besides what is described in the following sections) includes the incorporation of new data structures [12] and of combinatorial [1, 2, 10, 14] as well as geometric algorithms [3, 4, 7, 8, 13].

References

- [1] J. Dorchain. Implementierung eines Algorithmus zum Berechnen der Konvexen Hülle. *Diploma Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken*, 1995.
- [2] P. Hilpert. Implementierung von Heuristiken zum Traveling Salesman Problem. *Diploma Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken*, 1995.
- [3] K. Jung. Konvexe Polyeder. *Diploma Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken*, 1995.
- [4] P. Ksinsik. Ein approximativer Bewegungsplanungsalgorithmus zum Bewegen eines Rechteckes basierend auf Voronoi-Diagrammen, deren Distanzfunktionen durch Orientierungen des Rechtecks gegeben sind. *Diploma Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken*, 1995.

[5] U. Lauther. A fast planning tool for routing and scheduling of cargo trains. In *ALCOM Workshop*, 1993.

[6] U. Lauther. Routing and scheduling of trains, a prototype based on LEDA. In *Workshop on Optimization in Production and Transportation*, 1994.

• [7] S. Leinenbach. Eine effiziente Implementierung des Datentyps Polyeder. *Diploma Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken*, 1995.

• [8] S. Lutter. Ein approximativer Bewegungsplanungsalgorithmus zum Bewegen zweier gelenkverbundener Rechtecke. *Diploma Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken*, 1995.

• [9] K. Mehlhorn and S. Näher. LEDA: A platform for combinatorial and geometric computing. *Communications of the ACM*, 38(1):96–102, 1995.

• [10] U. Meyer. Deterministische PRAM-Simulation auf Gittern. *Diploma Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken*, 1995.

• [11] S. Näher. LEDA Manual, Version 3.1. Technical Report MPI-I-95-1-002, Max-Planck-Institut für Informatik, Saarbrücken, January 1995.

• [12] M. Paul. Augmented tree data structures based on skiplists and randomized search trees. *Ph.D. Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken*, 1995.

• [13] K.D. Rottmann. Bewegen eines Polygons. *Diploma Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken*, 1995.

• [14] T. Ziegler. Implementierung eines Algorithmus für das Maximum Weighted Matching Problem. *Diploma Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken*, 1995.

3.5.2 Exact Geometric Computations

Investigators: Christoph Burnikel, Kurt Mehlhorn, Stefan Näher, Stefan Schirra and Christian Uhrig

In computational geometry almost all papers assume exact real arithmetic. Implementors of geometric algorithms then often simply replace the exact real arithmetic of this model by fixed precision arithmetic, thereby making theoretically correct algorithms incorrect. Two approaches have been taken to remedy this situation. The first approach is redesigning geometric algorithms for fixed precision arithmetic. The redesign is difficult and can inherently lead to inexact results. In many cases the algorithms differ significantly from their exact arithmetic counterparts and that is probably the biggest disadvantage of the inexact arithmetic approach: it does not allow us to directly use the vast body of available geometric algorithms. The other approach advocates the use of exact real arithmetic. In principle, this approach is trivial. After all, it is well known how to compute exactly with integers, rationals, and even algebraic numbers. So the question really is: Can exact arithmetic be provided in an efficient and convenient way?

We considered two basic geometric problems in more detail, line segment intersection and Voronoi diagram of line segments and points. For the well-known plane sweep algorithm by Bentley and Ottmann [2] for computing the intersection of a set of line segments, exact rational arithmetic is sufficient if the coordinates of all input points determining the segments are **doubles**. Our implementation [5] in LEDA uses the so-called *floating point filters* [4] to reduce the overhead of exact computation. Based on an error analysis potentially unreliable computations are filtered out. Only if the computation with doubles does not guarantee exactness, e.g. in a sign computation, the computation is done (again) using exact rational arithmetic. Our implementation is about three times slower than the straightforward unreliable implementation using computation with **double**s, but it is correct!

In Voronoi diagram computations an important test is the incircle test. A vertex of a Voronoi diagram is the center of a circle touching three sites. The incircle test asks for the position (inside, on, or outside) of a fourth site with respect to a Voronoi circle. Assume for concreteness that the three defining sites are the origin p and lines l_1 and l_2 and that the fourth site is a line l_3. Assume further that the equation of line l_i is $a_i x + b_i y + c = 0$ with a_i, b_i, and c_i being $2k$-bit integers. The Voronoi vertex v has coordinates (x_v, y_v) where

$$x_v = \frac{a_1 c_2 + a_2 c_1 \pm \sqrt{2c_1 c_2 (a_1 a_2 - b_1 b_2 + \sqrt{(a_1^2 + b_1^2)(a_2^2 + b_2^2)})}}{\sqrt{(a_1^2 + b_1^2)(a_2^2 + b_2^2)} - a_1 a_2 - b_1 b_2}$$

and

$$y_v = \frac{b_1 c_2 + b_2 c_1 \mp \operatorname{sign}(a_1 b_2 + a_2 b_1)\sqrt{2c_1 c_2 (b_1 b_2 - a_1 a_2 + \sqrt{(a_1^2 + b_1^2)(a_2^2 + b_2^2)})}}{\sqrt{(a_1^2 + b_1^2)(a_2^2 + b_2^2)} - a_1 a_2 - b_1 b_2}.$$

The incircle test involves a comparison between the distance between v and p and the distance between v and l_3 and is therefore tantamount to comparing $x_v^2 + y_v^2$ and $(b_3 x_v + b_3 y_v + c_3)/(a_3^2 + b_3^2)$. We show that precision $48k$ suffices for this test, which is two orders of magnitude better than the bound obtained by applying general root separation bounds from computer algebra. The tests can be evaluated by repeated squaring and exact arbitrary precision integer arithmetic or by a floating point computation that guarantees the required precision. The latter approach can be improved by *lazy evaluation*, i.e. increasing the precision in steps until sufficient precision is reached. Our experiments with incircle tests generated by an unreliable Voronoi diagram algorithm have shown that both approaches take roughly the same amount of time. We report on our results in [3]. Our experiments have also shown that the full $48k$ precision was rarely needed in the examples. On average, precision $6k$ seems to suffice.

In both examples exact computation is provided by the user herself. The code looks still quite different from the code produced by implementors who simply replace exact real computation by floating point arithmetic (especially if repeated squaring is used), although from an algorithmic point of view it is essentially the same. There is clearly a need to package the exact computation. Therefore we added new number types to LEDA, that facilitate exact computation for many geometric problems. The most general one is the data type **real**. Every integer is a **real** and **real**s are closed under the operations addition, subtraction, multiplication, division and squareroot. All comparison operators $\{>, \geq, <, \leq, =\}$ are *exact*. In order to determine the sign of a real number x the data type first computes a *separation bound* q such that $|x| \leq q$ implies $x = 0$ and then computes an approximation of x of sufficient precision to decide the sign of x. The user may assist the data type by providing a separation bound q. The data type also allows evaluation of real expressions with arbitrary precision. The following is (part of) the LEDA manual page for **real**s.

real	$x + y$	addition
real	$x - y$	subtraction
real	$x * y$	multiplication
real	x / y	division
real	$- x$	negation
real	sqrt(**real** x)	squareroot operation
int	x.sign()	returns -1 if (the exact value of) x is < 0, 1 if $x > 0$, 0 if x is $= 0$.
int	x.sign(**int** k)	as above. *Precondition:* if $\lvert x\rvert \leq 2^{-k}$ then $x = 0$.
void	x.improve(**int** k)	(re-)computes the approximation of x such that its absolute error is bounded by 2^{-k}.
void	x.compute(**int** k)	(re-)computes the approximation of x; each numerical operation is carried out with k binary places.
void	x.compute_up_to(**int** k)	(re-)computes the approximation of x such that the relative error is bounded by 2^{-k}.
bool	$x < y$	returns true if x is smaller than y
bool	$x \ldots y$	further exact comparisons

The data type **real** provides exact computation in a convenient way. In an implementation of a geometric algorithm in C++, **real**s can be used like **double**s.For example, the following C++ procedure realizes the incircle test discussed above.

int incircle(**real** a_1, **real** b_1, **real** c_1, **real** a_2, **real** b_2, **real** c_2, **real** a_3, **real** b_3, **real** c_3)
{
  real $N = (a_1 * a_1 + b_1 * b_1) * (a_2 * a_2 + b_2 * b_2)$;
  real $C = a_1 * a_2 - b_1 * b_2$;
  real $D = sqrt(N) - (a_1 * a_2 + b_1 * b_2)$;
  real $S = -(a_1 * b_2 + a_2 * b_1)$;
  real $x_v = (a_1 * c_2 + a_2 * c_1 + sqrt(2 * c_1 * c_2 * (sqrt(N) + C))/D$;
  real $y_v = (b_1 * c_2 + b_2 * c_1 + S.\mathrm{sign}() * sqrt(2 * c_1 * c_2 * (sqrt(N) - C))/D$;
  real $E = (a_3 * x_v + b_3 * y_v + c_3) * (a_3 * x_v + b_3 * y_v + c_3) - (a_3 * a_3 + b_3 * b_3) * (x_v * x_v + y_v * y_v)$;
  return E.sign();
}

If a_i, b_i, c_i are known to be $2k$-bit integers, the sign computation E.sign() can be replaced by $E.\mathrm{sign}(48 * k)$, see [3]. Without this assistance the sign computation could be very slow.

The implementation of **real**s is based on the LEDA data types **integer** and **bigfloat** which are arbitrary precision integers and floating point numbers with exponents and mantissa of arbitrary length. Similar to [1, 8] a **real** is represented by the expression which defines it. Furthermore it has a **double** approximation $\hat{x}$ together with a relative error bound ϵ_x. It also uses floating-point filter and lazy evaluation: When the double test is inconclusive the quality of the approximation is repeatedly doubled (using calls of improve) until a decision is possible. A decision is possible if either the absolute value of the approximation is less than its absolute error or the separation bound is reached. The latter bound is either provided by the user or computed as in [7, 8].

Recent experiments have shown that the code obtained by replacing **double**s by **real**s in the straightforward implementation of the line segment intersection algorithm is about eight times slower than the original code.

References

[1] M.O. Benouamer, P. Jaillon, D. Michelucci, and J-M. Moreau. A "lazy" solution to imprecision in computational geometry. In *5th Canadian Conf. on Computational Geometry*, pages 73–78, 1993.

[2] J.L. Bentley and T.A. Ottmann. Algorithms for reporting and counting geometric intersections. In *IEEE Trans. Comput.*, volume C-28, pages 643–647, 1979.

• [3] C. Burnikel, K. Mehlhorn, and S. Schirra. How to compute the Voronoi diagram of line segments: Theoretical and experimental results. In *Algorithms - ESA'94*. LNCS, Springer-Verlag, September 1994.

[4] S. Fortune and C. van Wyk. Efficient exact arithmetic for computational geometry. *Proc. of the 9th Symp. on Computational Geometry*, pages 163–171, 1993.

• [5] K. Mehlhorn and S. Näher. Implementation of a sweep line algorithm for the straight line segment intersection problem. Technical Report MPI-I-94-160, Max-Planck-Institut für Informatik, Saarbrücken, October 1994.

• [6] K. Mehlhorn and S. Näher. The implementation of geometric algorithms. In *13th World Computer Congress IFIP94*, volume 1, pages 223–231. Elsevier Science B.V. North-Holland, Amsterdam, 1994.

[7] M. Mignotte. *Mathematics for Computer Algebra*. Springer Verlag, 1992.

[8] Ch.K. Yap and T. Dube. The exact computation paradigm. In *Computing in Euclidean Geometry*. World Scientific Press. to appear, 2nd edition.

3.5.3 Trapezoidal decomposition induced by a set of line segments

Investigator: Thomas Schilz

A fundamental problem in computational geometry is to compute the trapezoidal decomposition induced by a set of line segments. The problem is defined as follows: From each segment's endpoint and from each point of intersection between two segments raise a vertical *attachment* upwards and downwards to the first segment reached, in both directions. This way the plane is decomposed into trapezoidal regions.

The problem has been intensively investigated in the past and there exist deterministic [1] as well as randomized [3, 2] optimal algorithms with (expected) time complexity $O(n \log n + k)$ where n is the number of segments and k is the number of intersections.

The cited algorithms all assume that their input is non degenerate, which means that there are no overlapping segments, no more than two segments intersecting in a common point and no two endpoints or intersection points sharing a common x coordinate. Usually the authors point out that a degenerate input may be slightly perturbed such that the degeneracies disappear. But if we consider the case of several line segments intersecting in a common point then it is desirable to obtain a running time of $O(n \log n + i)$ where i is the number of intersection points not the number of pairs of intersecting segments. Perturbing the input makes this impossible.

In [4] we present an implementation, in C++ using LEDA, of the algorithm in [3] (p. 84ff) that handles degeneracies explicitly and show how the desired running time can easily be achieved in this way.

References

[1] Bernard Chazelle and Herbert Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. Technical Report UIUCDCS–R–88–1419, Department of Computer Science, University of Illinois at Urbana–Champaign, March 1988.

[2] Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.

[3] Ketan Mulmuley. *Computational geometry: an introduction through randomized algorithms.* Prentice Hall, 1994.

• [4] T. Schilz. Implementation of a randomized incremental algorithm for computing the trapezoidal decomposition induced by a set of line segments in the plane (to appear). Technical report, Max-Planck-Institut für Informatik, Saarbrücken, Germany, March 1995.

4 Visitors

Since December 1993, the following researchers (39 in total) visited or are visiting our group.

Alberts, David	01.03.94 - 31.07.94	FU Berlin
Anhalt, Christopher	03.06.94	Universität Karlsruhe
Prof. Chen, Danny	01.06.94 - 31.07.94	University of Notre Dame, Notre Dame, Indiana, USA
Prof. Cheng, Siu-Wing	01.07.94 - 22.07.94	The Hong Kong University of Science and T., Hong Kong
Dr. Chlebus, Bogdan	01.08.94 - 30.09.94	Warsaw University, Warsaw, Poland
Prof. Cho, Hwan Gue	17.01.94 - 16.01.95	Pusan National University, Pusan, Korea
Dr. Cucher, Felipe	26.04.94	Universität Pompen Fabra, Barcelona, Spain
Prof. Drysdale, Scott	27.02.94 - 04.03.94	Dartmouth College, Hanover, New Hampshire, USA
Dr. Golin, Mordecai	20.06.94 - 03.07.94	The Hong Kong University of Science and T., Hong Kong
Dr. Golin, Mordecai	15.07.94 - 29.07.94	The Hong Kong University of Science and T., Hong Kong
Gupta, Prosenjit	08.07.94 - 30.09.94	University of Minnesota Minneapolis, Minnesota, USA
Prof. Hartmanis, J.	01.10.93 - 31.05.94	Cornell University, Ithaca, New York, USA
Huson, Daniel	24.08.94	Universität Bielefeld
Dr. Jansen, Klaus	29.06.94	Universität Trier
Prof. Jünger, M.	04.02.94	Universität Köln
Prof. Kapoor, Sanjiv	10.06.94 - 31.07.94	Indian Institute of Technology, New Delhi, Indien
Prof. Katoh, Naoki	29.09.94 - 01.10.94	Kobe Univ. of Commerce Kobe, Japan
Prof. Klein, Rolf	13.09.94	Fernuniversität Hagen
Koga, Hisashi	23.06.94 - 05.07.94	The University of Tokyo, Tokyo, Japan
Prof. Krithivasan, K.	01.06.94 - 30.06.94	Indian Institute of Technology, Madras, Indien
Dr. La Poutré, Han	10.01.94 - 14.01.94	Utrecht University, Utrecht , The Netherlands
Prof. Lingas, Andrzej	28.06.94 - 30.06.94	Lund University, Lund, Sweden

Dr. Lüling, Reinhard	31.05.94 - 01.06.94	Universität Paderborn
Prof. Mount, Dave	20.06.94 - 15.12.94	University of Maryland, Maryland, USA
Dr. Mutzel, Petra	30.05.94 - 01.06.94	Universität Köln
Öhring, Sabine	15.06.94 - 17.06.94	University of North Texas, Denton, Texas, USA
Dr. Palios, Leonidas	12.09.94 - 22.09.94	The Geometric Center, Minneapolis, Minnesota, USA
Dr. Panconesi, A.	11.01.94 - 18.01.94	CWI Amsterdam, Amsterdam, The Netherlands
Papatriantafilou, Marina	01.01.95 - 31.12.95	Computer Technology Institute, Univ. of Patras, Greece
Dr. Raman, Rajeev	24.10.94 - 23.11.94	King's College London, London, United Kingdom
Dr. Ranjan, Desh	15.05.94 - 15.08.94	New Mexico State University Las Cruces, New Mexico, USA
Dr. Rauch, Monika	04.07.94 - 05.07.94	Cornell University, Ithaca, USA
Ruffing, Andreas	15.06.94	MPI für Physik, München
Prof. Salowe, Jeffrey	15.06.94 - 31.07.94	University of Virginia, Charlottesville, Virginia, USA
Prof. Schröder, Heiko	01.03.94 - 20.11.94	University of Newcastle, Newcastle, Australia
Prof. Spirakis, Paul	13.04.94	Computer Technology Institute, Univ. of Patras, Greece
Dr. Suel, Thorsten	13.12.94 - 20.12.94	NEC Research Institute, Princeton, New Jersey, USA
Prof. Sugihara, K.	08.03.94	University of Tokyo Tokyo, Japan
Prof. Zelikovsky, A.	21.09.94 - 30.11.94	Institute of Mathematics Kishinev, Moldova

5 Journal and Conference Activities

5.1 Editorial positions

Kurt Mehlhorn has been an editor of *Algorithmica* (since 1985), *Computational Geometry: Theory and Applications* (since 1990), *Information and Computation* (since 1985), *Int'l Journal of Computational Geometry* (since 1990), *Discrete and Computational Geometry* (since 1988), and *SIAM Journal on Computing* (since 1988).

5.2 Conference positions

Torben Hagerup is/was a program committee member of the:

- 6th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 1994), Cape May, New Jersey, USA.
- 4th Scandinavian Workshop on Algorithm Theory (SWAT 1994), Århus, Denmark.
- 10th International Conference on Fundamentals of Computation Theory (FCT 1995), Dresden, Germany.
- 7th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 1995), Santa Barbara, California, USA.
- 2nd Workshop on Parallel Algorithms for Irregularly Structured Problems (Irregular 1995), Lyon, France.

Kurt Mehlhorn was the chairman of the program committee of the 10th Annual ACM Symposium on Computational Geometry, 1994.

5.3 Organization of Workshops

Rudolf Fleischer organized the "22. Workshop über Komplexitätstheorie, Datenstrukturen und effiziente Algorithmen" (22nd Workshop on Complexity Theory, Data Structures and Efficient Algorithms) which took place in the Max-Planck-Institut für Informatik on February 8, 1994. This workshop takes place three times a year, always in Germany. Twenty talks were given and the workshop was attended by (slightly more than) fifty people. Abstracts of the talks can be found in the Technical Report MPI-I-94-104.

Kurt Mehlhorn, with Jan van Leeuwen (Utrecht) and Thomas Reps (Madison), organized the Dagstuhl-Seminar on "Incremental Computation and Dynamic Algorithms" held in Schloß Dagstuhl (near Saarbrücken) on May 2-6, 1994. The purpose of the Seminar was to bring two research communities together that have common interests, but which (to date) have had relatively limited contact; namely, theoretical computer scientists working in the area of "dynamic algorithms", and programming-language and systems researchers working in the area of "incremental computing". The Seminar provided the opportunity to present the state-of-the-art in the relevant fields at a high level and to let the two research communities benefit from each other's insights. Twenty-seven talks were given and the Seminar was attended by thirty-one participants from Europe and the United States. Abstracts of the talks can be found in the Dagstuhl-Seminar-Report No. 88.

Kurt Mehlhorn, with Robert Tarjan (Princeton), organized the Workshop on "Efficient Algorithms" held in the Mathematisches Forschungsinstitut Oberwolfach (near Freiburg) on August 7-13, 1994. The goal of the Workshop was to stimulate research in (almost) all areas of algorithms' design. Particular emphasis was given on the efficient solution of large problems (with respect to the size of the input) and on the design of simple, easy to implement, algorithms. Twenty-nine talks were given and the Workshop was attended by thirty-one participants from Europe and the United States. Abstracts of the talks can be found in the Tagungsbericht 34/1994 of the Mathematisches Forschungsinstitut Oberwolfach.

6 Teaching Activities

The group contributes intensively to the Computer Science Curriculum. The core courses "Praxis des Programmierens", "Datenstrukturen und Algorithmen" and "Optimierung" are always taught by members of the group. In addition we teach some specialized courses.

Summer Semester 1994

LECTURES:
Praxis des Programmierens (K. Mehlhorn, S. Schirra)
Algorithmen und Komplexität II (S. Albers, M. Smid)
Randomisierte Algorithmen (T. Hagerup)
String Matching (H.P. Lenhof)
Parallele Algorithmen für Netzwerke (C. Rüb, J. Sibeyn)
SEMINAR:
Datenstrukturen und Netzwerkalgorithmen (R. Fleischer, K. Mehlhorn)
PROJECT CLASS:
Softwarekonstruktion (S. Näher, C. Uhrig)

Winter Semester 1994/95

LECTURES:
Praxis des Programmierens (S. Schirra)
Datenstrukturen und Algorithmen (K. Mehlhorn, C. Schwarz)
On-line Algorithmen (R. Fleischer)
Computational Biology (H.P. Lenhof)
Graph- und Netzwerkalgorithmen (S. Chaudhuri, C. Rüb)
PROJECT CLASS:
Effiziente Algorithmen unter LEDA (K. Mehlhorn, C. Uhrig)

We are also teaching the course "C-Blockkurs" which is a compact (lasting two weeks) introduction into the C programming language. Furthermore there is a course called "Ausgewählte Kapitel aus Effiziente Algorithmen" ("Selected Topics in Algorithms"). It is organized by our post-docs and intented for Ph.D. students. (See section 8.)

Class Notes

For some of the courses there are class notes prepared by the lecturer. Michiel Smid wrote class notes [6] on selected topics in data structures. They cover skip lists, the union-find problem, range trees and the post-office problem, and maintaining order in a list. In his class notes on interactive proof systems [4], Sanjeev Saluja covers recent developments which led to results on the hardness of approximability of some **NP**-hard problems like MaxClique. The class notes start with a review of the early result **IP=PSPACE** [3, 5]. Next probabilistically checkable proofs (**PCP**) are introduced. Then the intermediate result of Arora and Safra [2]: $\mathbf{NP} \subseteq \mathbf{PCP}(log(n), polylog(n))$ is proved and it is also shown that the above result can be strengthened further. Then it is proved: $\mathbf{NP} \subseteq \mathbf{PCP}(poly, 1)$; which is another important building block of the final result. Finally it is shown how strengthened version of the Arora-Safra result and the above result can be combined to obtain the final result : $\mathbf{NP} \subseteq \mathbf{PCP}(log(n), 1)$ [1].

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 14–23, 1992.

[2] S. Arora and S. Safra. Probabilistic checking of proofs; a new characterization of NP. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 2–13, 1992.

[3] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 2–10, 1990.

• [4] J. Radhakrishnan and S. Saluja. Interactive proof system. Technical report, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1995.

[5] A. Shamir. IP = PSPACE. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 11–15, 1990.

• [6] M. Smid. Lecture notes: selected topics in data structures. Technical Report MPI-I-94-155, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1994.

7 Dissertations and Habilitations

7.1 Dissertations

Completed:

Klär G.: Verdrahtungsprobleme auf planaren Graphen. June 1994.
Rasch R.: Furthest Site Abstract Voronoi Diagrams. September 1994.

Ongoing:

Bast H.: Efficient Parallel Algorithms for Fundamental Problems on the CRCW PRAM. December 1997[2].
Burnikel Ch.: Precision and Degeneracy in Geometric Computations. March 1995[2].
Lauer Th.: Dynamische Lastbalancierung. February 1995[2].
Paul M.: Augmented Tree Data Structures Based on Skiplists and Randomized Search Trees. February 1995[2].
Priebe V.: Analysis of randomized combinatorial algorithms. June 1995[2].
Reinert K.: Algorithmen zur Sequenzanalyse. December 1997[2].
Schilz T.: Verteilte Algorithmen für den Design–Rule–Check von VLSI–Layouts. June 1995[2].

[2] Expected completion date.

Schwarzenecker E.: Ein NP–vollständiges Problem aus der Kartographie. February 1995[2].
Thiel Ch.: Schnitt von Polyedern in höheren Dimensionen. March 1995[2].

7.2 Habilitations

Completed:

Näher St.: Das Minimum Cost Flow-Problem. June 1994.
Smid M.: Nachbarschaftsprobleme in der algorithmischen Geometrie. February 1995.

8 Organization of our Group

The group meets two to four times a week at 1.30 pm.

On Monday and Wednesday (1.30 - 2.15) we have our noon seminar. It lasts about 45 minutes and is reserved for presentations of new results and ongoing research. We also ask our guests to give presentations in the noon seminar.

On Tuesday and Thursday (1.30 - 3.00) we run the "Selected Topics in Algorithms" course. This course is reserved for two to four week intensive treatments of subjects of current interest. Topics treated from December 1993 until today were:

Reconfigurable Networks (H. Schröder)
The Blum-Shub-Smale Model of Computation over the Reals (F. Cucker)
Majorization and its applications (S. Arikati)
Approximate max-flow min-cut theorems for Multicommodity Flows (N. Garg)
Spanners: Approximating the complete Euclidean graph (S. Arya, D. Mount, M. Smid)
Pattern Matching (R. Hariharan)
Hammock Decomposition of Graphs (C. Zaroliagis)
Semi-definite Programming (S. Mahajan)
Interactive Proof Systems (S. Saluja)
Topics in Probabilistic Methods (P. Kelsen).

We have elected an executive committee (K. Mehlhorn, T. Hagerup, V. Priebe, S. Arya, A. Eßer, C. Schmitz) which makes the day to day decisions concerning the group.

9 Cooperations

At an informal level we cooperate with researchers from the other research group of the Max-Planck-Institut für Informatik and from the Computer Science Department of the university. With Ganzinger's group we cooperate on two problems: specification of abstract data types and identification of partial orders. In the computer science department our main contacts are Prof. Hotz (on VLSI–design), Prof. Paul (on parallel algorithms), Prof. Buchmann (on computer algebra), and Prof. Wilhelm (on parallel programming languages).

At an institutional level we are involved in five research projects:

- SFB 124 VLSI–Entwurfsmethoden und Parallelität,
- ESPRIT Basic Research Action No. 7141 (ALCOM II),
- EC Cooperative Action IC-1000–project ALTEC,
- EC-project HCM,
- GIF research project: Arrangements in Computational Geometry.

9.1 SFB 124 VLSI–Entwurfsmethoden und Parallelität

The Sonderforschungsbereich (SFB) 124 is a special research effort, sponsored by the DFG (Deutsche Forschungsgemeinschaft). The project is directed towards the practical aspects of parallel systems: design, understanding, programming, and optimized use.

The project was initiated in 1983, and has been prolonged recently until December 1997. Probably this is going to be the last three year period of the project.

Partners and group leaders of the project are:

Universität Kaiserslautern (Professors: Härder, Nehmer)
Universität des Saarlandes (Professors: Buchmann, Hotz, Loeckx, Mattern, Molitor, Paul, Thiele, Wilhelm, Zimmermann)
Max-Planck-Institut für Informatik (Prof. Mehlhorn)

9.2 ALCOM

ALCOM (Algorithms and Complexity) is an Esprit basic research action. It involves 12 partners in nine EU countries. The purpose of ALCOM is to foster algorithms research within Europe and to stimulate cooperation among research institutes from the EU. ALCOM has been very successful in reaching these goals. It has given the European algorithms community an identity and its own conference ESA (European Symposium on Algorithms), and it has led to close collaboration within it.

The project started in 1989 as ALCOM I. The actual ALCOM II project runs from 24-7-1992 until 23-7-1995. Presently we are planning for a further three year period on a slightly different basis, but it is not sure that this will be granted.

Kurt Mehlhorn and Christoph Storb coordinate the action. Torben Hagerup is our local contact person.

Partners and group leaders of the action are:

Århus University, Århus, Denmark (Prof. E. M. Schmidt)
Universitat Politècnica de Catalunya, Barcelona, Spain (Prof. J. Diaz)
Freie Universität, Berlin, Germany (Prof. E. Welzl)
University of Dublin, Dublin, Irish Republic (Prof. C. O'Dúnlaing)
EHESS, Paris, France (Prof. P. Rosenstiehl)
INRIA–Paris, Rocquencourt, France (Dr. P. Flajolet)
Universität-GH Paderborn, Paderborn, Germany (Prof. B. Monien)
Computer Technology Institute, Patras, Greece (Prof. P. Spirakis)
Universitá di Roma, Roma, Italy (Prof. G. Ausiello)
INRIA–Sophia-Antipolis, Sophia-Antipolis, France (Dr. J. Boissonnat)
University of Utrecht, Utrecht, Netherlands (Prof. J. van Leeuwen)
University of Warwick, Coventry, Great Britain (Prof. M. Paterson)
Max-Planck-Institut für Informatik, Saarbrücken, Germany (Prof. K. Mehlhorn)

9.3 ALTEC

ALTEC (Basic Algorithms for Future Technologies) is the extension of ALCOM to Eastern Europe. The goals are similar, but, as the project is of a much smaller nature, not so highly set. Through the project we get some additional travel money, and some money for personnel. The project has been successful in strengthening contacts between the various involved sites. Twice a year a workshop is organized.

The project has started in 1992 and will run out in March 1995. A prolongation is not planned. Though we will continue to cooperate at an informal level.

The ALTEC project is chaired by Prof. van Leeuwen. Jop Sibeyn is our local contact person.

Partners and group leaders of the project are:

University of Bordeaux, Bordeaux, France (Prof. Cori)
Comenius University, Bratislava, Slovakia (Prof. Rovan)
Slovac Academy of Science, Bratislava, Slovakia (Prof. Sykora)
Eötvös University, Budapest, Hungary (Prof. Rónyai)
Charles University, Prague, Czech Republic (Professors: Kucera, Wiedermann)
University of Utrecht, Utrecht, Netherlands (Prof. van Leeuwen)
Warsaw University, Warsaw, Poland (Prof. Rytter)
Max-Planck-Institut für Informatik, Saarbrücken, Germany (Prof. Mehlhorn)

9.4 HCM

Human Capital and Mobility (HCM) is the post-doc program of the European Union. The project strives for the development of more efficient algorithms, and to make these algorithms accessible to the non-algorithmic community. We have two post-doc positions paid from money of this project.

The project has started in 1994, and runs for three years. HCM is not a cooperation project: there are no partners, though other institutes profit from the same funding.

9.5 GIF

The German-Israeli Foundation for Scientific Research and Development (GIF) is a bi-national science foundation. It was created by the two governments in order to promote and fund joint civil research and development projects in basic and applied research.

We have been successfully working with our Israeli partner Micha Sharir from Tel Aviv University and our German partner Emo Welzl from Free University in Berlin on randomized techniques and related studies concerning arrangements in computational geometry. Our work benefited from mutual visits.

In the GIF II project we will study various basic problems in computational and discrete geometry involving arrangements of curves and surfaces, and investigating the use of randomized techniques to solve many of these problems.

The GIF I project, *Randomized Techniques and Related Studies of Arrangements in Computational Geometry*, runs from 1-10-1990 until 30-9-1994, (including an extension of one year without additional funding). The GIF II project, *Study of Arrangements and Randomized Techniques in Discrete and Computational Geometry*, spans the period from 1-1-1995 until 31-12-97. The local contact person is Stefan Schirra.

Partners and group leaders of the project are:

Freie Universität, Berlin, Germany (Prof. E. Welzl)
Tel-Aviv University, Tel-Aviv, Israel (Prof. M. Sharir)
Max-Planck-Institut für Informatik, Saarbrücken, Germany (Prof. K. Mehlhorn)

9.6 Industry

Our main industrial partner is Siemens AG, Munich. Dr. Hammer from Siemens and Torben Hagerup and Thomas Lauer cooperate on load balancing algorithms for parallel machines. Dr.

Lauther has cooperated with Stefan Näher, Kurt Mehlhorn and Christian Uhrig on efficient graph algorithms and a planning tool for scheduling cargo trains. In the past year, Christian Uhrig has spent three months with Lauther's group. Thomas Lauer holds a Siemens Ph.D. scholarship. Kurt Mehlhorn serves on the scientific advisory board of Siemens corporate research.

Through the LEDA project we have loose contacts to several other companies. These relations are of the producer-consumer type, that is, we receive bug reports and sometimes enthusiastic comments.

10 Recent Publications

10.1 In Journals

[1] H. Alt, L. Guibas, K. Mehlhorn, R. Karp, and A. Widgerson. A method for obtaining randomized algorithms with small tail probabilities. *Algorithmica*, to appear.

[2] S.R. Arikati and A. Maheshwari. Realizing degree sequences in parallel. *SIAM Journal Discrete Math.*, to appear.

[3] H. Bast and T. Hagerup. Fast parallel space allocation, estimation and integer sorting. *Information and Computation*, to appear.

[4] G. Bilardi, S. Chaudhuri, D. Dubhashi, and K. Mehlhorn. A lower bound for area universal networks. *Information Processing Letters*, 51(2):101–106, July 1994.

[5] S. Chaudhuri. Tight bounds on oblivious chaining. *SIAM Journal on Computing*, 23, December 1994.

[6] J. Cheriyan and T. Hagerup. A randomized maximum-flow algorithm. *SIAM Journal on Computing*, to appear, April 1995.

[7] J. Cheriyan, T. Hagerup, and K. Mehlhorn. An $o(n^3)$-time maximim-flow algorithm. *SIAM Journal on Computing*, to appear.

[8] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi. Exact ground states of Ising spin glasses: New experimental results with a branch and cut algorithm. to appear in the *Journal of Statistical Physics*, 1995.

[9] P. Dietz, K. Mehlhorn, R. Raman, and C. Uhrig. Lower bounds for set intersection queries. *Algorithmica*, to appear.

[10] R. Fleischer. A tight lower bound for the worst case of bottom-up-heapsort. *Algorithmica*, 11:104–115, 1994.

[11] R. Fleischer, H. Jung, and K. Mehlhorn. A communication-randomness tradeoff for two-processor systems. *Information and Computation*, to appear, January 1995.

[12] P. Gupta, R. Janardan, and M. Smid. Further results on generalized intersection searching problems: counting, reporting, and dynamization. *Journal of Algorithms*, to appear, 1995.

[13] T. Hagerup. A lower bound for the emulation of PRAM memories on processor networks. *Information and Computation*, to appear.

[14] T. Hagerup. Fast deterministic processor allocation. *Journal of Algorithms*, to appear, May 1995.

[15] T. Hagerup and Jörg Keller. Fast parallel permutation algorithms. *Parallel Processing Letters*, to appear.

[16] T. Hagerup and Mirosław Kutyłowski. Fast integer merging on the EREW PRAM. *Algorithmica*, to appear.

[17] T. Hagerup and M. Maas. Generalized topological sorting in linear time. *Nordic Journal of Computing*, 1:38–49, 1994.

[18] P.J. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. *Computational Geometry: Theory and Applications*, 4:137–156, 1994.

[19] M. Kaufmann and K. Mehlhorn. A linear-time algorithm for the homotopic routing problem in grid graphs. *SIAM Journal of Computing*, 23(2):227–246, 1994.

[20] Michael Kaufmann, Heiko Schröder, and Jop F. Sibeyn. Asymptotically optimal and practical routing on the reconfigurable mesh. *Parallel Processing Letters*, to appear, 1995.

[21] H.-P. Lenhof and M. Smid. An optimal construction method for generalized convex layers. *International Journal of Computational Geometry & Applications*, 3:245–267, 1993.

[22] H.-P. Lenhof and M. Smid. Using persistent data structures for adding range restrictions to searching problems. *RAIRO Theoretical Informatics and Applications*, 28:25–49, 1994.

[23] H.-P. Lenhof and M. Smid. Maintaining the visibility map of spheres while moving the viewpoint on a circle at infinity. *Algorithmica*, 13:301–312, 1995.

[24] K. Mehlhorn and S. Näher. LEDA: A platform for combinatorial and geometric computing. *Communications of the ACM*, 38(1):96–102, 1995.

[25] M. Müller, Ch. Rüb, and W. Rülling. A circuit for exact summation of floating-point numbers. *Information Processing Letters*, to appear, 1995.

[26] C. Schwarz, M. Smid, and J. Snoeyink. An optimal algorithm for the on-line closest-pair problem. *Algorithmica*, 12:18–29, 1994.

[27] M. Smid. Dynamic rectangular point location, with an application to the closest pair problem. *Information and Computation*, 116:1–9, 1995.

10.2 In Conference Proceedings

[1] S. Albers. A competitive analysis of the list update problem with lookahead. *Proceedings Mathematical Foundations of Computer Science (MFCS'94)*, 1994.

[2] S. Albers. Improved randomized on-line algorithms for the list update problem. *Proceedings 6th ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*, to appear, January 1995.

[3] S. Albers and H. Koga. New on-line algorithms for the page replication problem. In *Proceedings 4th Scandinavian Workshop on Algorithm Theory (SWAT'94)*. LNCS 824, Springer Verlag, 1994.

[4] A. Andersson, T. Hagerup, J. Håstad, and O. Petersson. The complexity of searching a sorted array of strings. In *Proceedings 26th ACM Symposium on Theory of Computing (STOC'94)*, pages 317–325, May 1994.

[5] S. Arikati, A. Maheshwari, and C. Zaroliagis. Saving bits made easy. In *Proc. 6th Canadian Conference on Computational Geometry (CCCG'94)*, pages 140–146, August 1994. Also Tech. Rep. MPI-I-94-148, 1994.

[6] S.R. Arikati and A. Maheshwari. Realizing degree sequences in parallel. In *Proc. 5th ISAAC'94, Lecture Notes in Comp. Sci. (LNCS) 834:261–269, Springer-Verlag*, 1994.

[7] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: short, thin, and lanky. In *Proc. 27th Annual ACM Symposium on the Theory of Computing (STOC'95), to appear*, 1995.

[8] S. Arya and D. M. Mount. Approximate range searching. In *Proc. 11th Ann. ACM Sympos. Comput. Geometry, to appear*, 1995.

[9] S. Arya, D. M. Mount, and O. Narayan. Accounting for boundary effects in nearest neighbor searching. In *Proc. 11th Ann. ACM Sympos. Comput. Geometry, to appear*, 1995. Also Tech. Rep. MPI-I-94-159.

[10] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 573–582, January 1994.

[11] S. Arya, D. M. Mount, and M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 703–712, 1994.

[12] S. Arya and M. Smid. Efficient construction of a bounded degree spanner with low weight. In *Proc. 2nd Annu. European Sympos. Algorithms (ESA)*, volume 855 of *Lecture Notes in Computer Science*, pages 48–59, 1994.

[13] G. Barnes and U. Feige. Short random walks on graphs. In *Proceedings 25th ACM Symposium on Theory of Computing (STOC'93)*, pages 728–737, May 1993.

[14] Greg Barnes. A method for implementing lock-free shared data structures. In *Proc. 5th ACM Symposium on Parallel Algorithms and Architectures*, 1993.

[15] P.G. Bradford, V. Choppella, and G.J.E. Rawlins. Lower bounds for the matrix chain ordering problem. To appear in the Proceedings of LATIN '95, 1995.

[16] Phillip G. Braford, Jean-Yves Marion, and Lawrence S. Moss. The additive fragment of linear logic is $\mathcal{NC}^1$-complete. In *To appear in the Proceedings of the International Conference in Logic and Computational Complexity*, 1995.

[17] C. Burnikel, K. Mehlhorn, and S. Schirra. How to compute the Voronoi diagram of line segments: Theoretical and experimental results. In *Algorithms - ESA'94*. LNCS, Springer-Verlag, September 1994.

[18] C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. In *Proc. of the 5th ACM-SIAM Symp. on Discrete Algorithms*, pages 16–23, 1994.

[19] S. Chaudhuri and D. Dubhashi. (Probabilistic) Recurrence Relations Revisited. In *Proceedings II Latinamerican Symposium on Theoretical Informatics (LATIN '95)*, 1995. to appear.

[20] S. Chaudhuri and T. Hagerup. Prefix graphs and their applications. In *Proc. 20th International Workshop on Graph-Theoretic Concepts in Computer Science(WG 94)*, Springer Lecture Notes in Computer Science, to appear.

[21] S. Chaudhuri and C. Zaroliagis. Shortest path queries in digraphs of small treewidth. In *Proc. 22nd Int'l Colloquium on Automata, Languages and Programming (ICALP'95), to appear*. LNCS, Springer-Verlag, 1995.

[22] B. Chlebus, A. Gambin, and P. Indyk. PRAM computations resilient to memory faults. In *Proc. 2nd Annual European Symposium on Algorithms (ESA '94)*, pages 401–412. LNCS 855, Springer-Verlag, 1994.

[23] Bogdan S. Chlebus, M. Kaufmann, and Jop F. Sibeyn. Deterministic permutation routing on meshes. In *Proc. 5th Symposium on Parallel and Distributed Processing*, pages 814–821. IEEE, 1993.

[24] A. Datta. Efficient parallel algorithms for geometric k-clustering problems. In *Proc. 11th Annual Symposium on Theoretical Aspects of Computer Science (STACS '94)*, pages 475–486. LNCS 775, Springer-Verlag, 1994.

[25] O. Devillers, M.J. Golin, K. Kedem, and S. Schirra. Revenge of the dog: Queries on Voronoi diagrams of moving points. In *Proc. of the 6th Canadian Conference on Computational Geometry*, pages 122–127, 1994.

[26] H. Djidjev, G. Pantziou, and C. Zaroliagis. On-line and dynamic algorithms for shortest path problems. In *Proc. 12th Symp. on Theoretical Aspects of Computer Science (STACS' 95), to appear*. LNCS, Springer-Verlag, March 1995. Also Tech. Rep. MPI-I-94-114, 1994.

[27] D. Dubhashi, K. Mehlhorn, D. Ranjan, and C. Thiel. Searching, sorting and randomised algorithms for central elements and ideal counting in posets. In *Proceedings 13th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'93)*, pages 436–443. LNCS 761, Springer-Verlag, 1993. Also Technical Report MPI-I-93-154, Max-Planck-Institut Saarbrücken, 1993.

[28] R. Fleischer. Decision trees : Old and new results. In *Proc. 25th Symp. on Theory of Computing (STOC'93)*, pages 468–477, May 1993. Also Technical Report MPI-I-92-125.

[29] R. Fleischer. A simple balanced search tree with $O(1)$ worst-case update time. In *Proc. 4th International Symposium on Algorithms and Computation (ISAAC'93)*, pages 138–146. LNCS 762, Springer Verlag, December 1993.

[30] N. Garg, H. Saran, and V.V. Vazirani. Finding separator cuts in planar graphs within twice the optimal. In *Proceedings, 35$^{\text{th}}$ Annual IEEE Symposium on Foundations of Computer Science*, pages 14–23, November 1994.

[31] P. Gupta, R. Janardan, and M. Smid. Efficient algorithms for generalized intersection searching on non-iso-oriented objects. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 369–378, 1994.

[32] P. Gupta, R. Janardan, and M. Smid. Fast algorithms for collision and proximity problems involving moving geometric objects. In *Proc. 2nd Annu. European Sympos. Algorithms (ESA)*, volume 855 of *Lecture Notes in Computer Science*, pages 278–289, 1994.

[33] P. Gupta, R. Janardan, and M. Smid. On intersection searching problems involving curved objects. In *Proc. 4th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 824 of *Lecture Notes in Computer Science*, pages 183–194, 1994.

[34] P. Gupta, R. Janardan, M. Smid, and B. Dasgupta. The rectangle enclosure and point-dominance problems revisited. In *Proc. 11th Ann. ACM Sympos. Comput. Geometry, to appear*, 1995. Also Tech. Rep. MPI-I-94-142.

[35] T. Hagerup. Optimal parallel string algorithms: Merging, sorting and computing the minimum. In *Proceedings, 26th Annual ACM Symposium on Theory of Computing (STOC 1994)*, pages 382–391, 1994.

[36] T. Hagerup, J. Katajainen, N. Nishimura, and P. Ragde. Characterizations of k-terminal flow networks and computing network flows in partial k-trees. In *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1995.

[37] J. Hartmanis. The structure of the complexity of computations: a guided tour through complexity classes. In *Proc. of the IFIP 13th World Computer Congress on Technology and Foundations: Information Processing '94, Vol. 1*, pages 213–220. IFIP Transactions / A-51, Elsevier, 1994.

[38] J. Hartmanis and S. Chari. On the intellectual terrain around NP. In *Proc. 2nd Italian Conference on Algorithms and Complexity (CIAC '94)*, pages 1–11. LNCS 778, Springer-Verlag, 1994.

[39] D. Kagaris, G. Pantziou, S. Tragoudas, and C. Zaroliagis. Quickest paths: Parallelization and dynamization. In *Proc. 28th Hawaii Int'l Conference on System Sciences (HICCS-28), to appear*, 1995.

[40] S. Kapoor and M. Smid. New techniques for exact and approximate dynamic closest-point problems. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 165–174, 1994.

[41] Michael Kaufmann, Harald Lauer, and Heiko Schröder. Fast deterministic hot-potato routing on processor arrays. In *Proc. Annual International Symposium on Algorithms and Computation*, pages 333–341. Springer-Verlag, 1994.

[42] Michael Kaufmann, Uli Meyer, and Jop F. Sibeyn. Towards practical permutation routing on meshes. In *Proc. 6th Symposium on Parallel and Distributed Processing*, pages 664–671. IEEE, 1994. Also Tech. Rep. MPI-I-94-153, 1994.

[43] Michael Kaufmann, Jop F. Sibeyn, and Torsten Suel. Derandomizing algorithms for routing and sorting on meshes. In *Proc. 5th Symposium on Discrete Algorithms*, pages 669–679. ACM-SIAM, 1994.

[44] D. Kavvadias, G. Pantziou, P. Spirakis, and C. Zaroliagis. Efficient sequential and parallel algorithms for the negative cycle problem. In *Proc. 5th Int'l Symp. on Algorithms and Computation (ISAAC'94)*, pages 270–278. LNCS 834, Springer-Verlag, August 1994.

[45] D. Kavvadias, G. Pantziou, P. Spirakis, and C. Zaroliagis. Hammock-on-ears decomposition: A technique for the efficient parallel solution of shortest paths and other problems. In *Proc. 19th Symp. on Mathematical Foundations of Comp. Science (MFCS '94)*, pages 462–472. LNCS 841, Springer-Verlag, August 1994. Also Tech. Rep. MPI-I-94-131, 1994.

[46] L. Kucera. Coloring k-colorable graphs in constant expected parallel time. In *Proc. 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'93)*, pages 167–176. LNCS 790, Springer-Verlag, 1994.

[47] H.-P. Lenhof and M. Smid. An animation of a fixed-radius all-nearest-neighbors algorithm. In *Proc. 10th Annual ACM Sympos. Comput. Geom.*, page 387, 1994.

[48] A. Maheshwari and A. Lingas. A simple optimal parallel algorithm for reporting paths in a tree. In *Proc. 11th Annual Symposium on Theoretical Aspects of Computer Science (STACS '94)*, pages 487–495. LNCS 775, Springer-Verlag, 1994.

[49] K. Mehlhorn and S. Näher. The implementation of geometric algorithms. In *13th World Computer Congress IFIP94*, volume 1, pages 223–231. Elsevier Science B.V. North-Holland, Amsterdam, 1994.

[50] K. Mehlhorn, R. Sundar, and Ch. Uhrig. Maintaining dynamic sequences under equality test in polylogarithmic time. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms (SODA'94)*, pages 213–222, 1994.

[51] Sabine Öhring, Jop F. Sibeyn, and Ondrej Sýkora. Optimal VLSI-layout for the efficient Petersen based interconnection network family. In *Proc. 6th International Conference Parallel and Distributed Computing and Systems*, pages 121–124. IASTED, 1994. Also Tech. Rep. TR-II-SAS-06/94/22, Institute for Informatics, Slovac Ac ademy of Sciences, 1994.

[52] Andrea Pietracaprina, Geppino Pucci, and Jop F. Sibeyn. Constructive deterministic PRAM simulation on a mesh-connected computer. In *Proc. 6th Symp. on Parallel Algorithms and Architectures*, pages 248–256. ACM, 1994.

[53] Ch. Rüb. Lower bounds for merging on the hypercube. *Proc. 2nd Italian Conference on Algorithms and Complexity, LNCS 778*, pages 213–222, 1994.

[54] Ch. Rüb. On the average running time of odd-even merge sort. *Proc. 12th Symposium on Theoretical Aspects of Computer Science, STACS'95*, 1995. To appear.

[55] Jop F. Sibeyn. Desnakification of mesh sorting algorithms. In *Proc. 2nd European Symposium on Algorithms, LNCS 855*, pages 337–390. Springer-Verlag, 1994. Also Tech. Rep. MPI-I-94-102, 1994.

[56] Jop F. Sibeyn. Deterministic sorting on circular arrays. In *Proc. 8th International Parallel Processing Symposium*, pages 406–410. IEEE, 1994.

[57] Jop F. Sibeyn. Independent sets and list ranking on meshes. In *Proc. Computing Science in the Netherlands*, pages 271–280, Amsterdam, Netherlands, 1994. SION.

[58] Jop F. Sibeyn, Bogdan S. Chlebus, and Michael Kaufmann. Shorter queues for permutation routing on meshes. In *Proc. 19th Symposium on the Mathematical Foundations of Computer Science, LNCS 841*, pages 597–607. Springer-Verlag, 1994.

[59] Jop F. Sibeyn and Tim Harris. Exploiting locality in LT-RAM computation. In *Proc. 4th Scandinavian Workshop on Algorithm Theory*, pages 338–349. Springer-Verlag, 1994.

[60] Jop F. Sibeyn and Michael Kaufmann. Deterministic 1-k routing on meshes. In *Proc. 11th Symposium on Theoretical Aspects of Computer Science, LNCS 775*, pages 237–248. Springer-Verlag, 1994. Also Tech. Rep. MPI-I-93-163, 1993.

10.3 Technical Reports

- **MPI–I–93–166**

  Efficient algorithms for generalized intersection searching on non-iso-oriented objects

  Author(s): Prosenjit Gupta, Ravi Janardan, Michiel Smid

  In a generalized intersection searching problem, a set S of colored geometric objects is to be preprocessed so that, given a query object q, the distinct colors of the objects of S that are intersected by q can be reported or counted efficiently. These problems generalize the well-studied standard intersection searching problems and are rich in applications. Unfortunately, the solutions known for the standard problems do not yield efficient solutions to the generalized problems. Recently, efficient solutions have been given for generalized problems where the input and query objects are iso-oriented, i.e., axes-parallel, or where the color classes satisfy additional properties, e.g., connectedness. In this paper, efficient algorithms are given for several generalized problems involving non-iso-oriented objects. These problems include: generalized halfspace range searching in $\mathcal{R}^d$, for any fixed $d \geq 2$, segment intersection searching, triangle stabbing, and triangle range searching in $\mathcal{R}^2$. The techniques used include: computing suitable sparse representations of the input, persistent data structures, and filtering search.

- **MPI–I–94–102**

  Desnakification of mesh sorting algorithms (revised and extended version)

  Author(s): Jop Sibeyn

  In all recent near-optimal sorting algorithms for meshes, the packets are sorted with respect to some snake-like indexing. Such algorithms are useless in many practical applications. In this paper we present deterministic algorithms for sorting with respect to the more natural row-major indexing.

  For 1-1 sorting on an $n \times n$ mesh, we give an algorithm that runs in $2 \cdot n + o(n)$ steps, with maximal queue size five. It is considerably simpler than earlier algorithms. Another algorithm performs k-k sorting in $k \cdot n/2 + o(k \cdot n)$ steps. Furthermore, we present *uni-axial* algorithms for row-major sorting. Uni-axial algorithms have clear practical and theoretical advantages over bi-axial algorithms. We show that 1-1 sorting can be performed in $2\frac{1}{2} \cdot n + o(n)$ steps. Alternatively, this problem is solved in $4\frac{1}{3} \cdot n$ steps for *all* n. For the practically important values of n, this algorithm is much faster than any algorithm with good *asymptotical* performance.

- **MPI–I–94–103**

  On the intellectual terrain around NP

  Author(s): Suresh Chari, Juris Hartmanis

  In this paper we view $P \stackrel{?}{=} NP$ as the problem which symbolizes the attempt to understand what is and is not feasibly computable. The paper shortly reviews the history of the developments from Gödel's 1956 letter asking for the computational complexity of finding proofs of theorems, through computational complexity, the exploration of complete problems for NP and PSPACE, through the results of structural complexity to the recent insights about interactive proofs.

- **MPI–I–94–104**

  Komplexitätstheorie und effiziente Algorithmen

  Editor: Rudolf Fleischer

  This publication contains abstracts of the 22nd workshop on complexity theory and efficient algorithms. The workshop was held on February 8, 1994, at the Max Planck Institute for Computer Science, Saarbrücken, Germany.

- **MPI–I–94–105**

  An implementation of a convex hull algorithm: version 1.0

  Author(s): Michael Müller, Joachim Ziegler

  We give an implementation of an incremental construction algorithm for convex hulls in $\mathbb{R}^d$ using *Literate Programming* and *LEDA* in C++. We treat convex hulls in arbitrary dimensions without any non-degeneracy assumption. The main goal of this paper is to demonstrate the benefits of the literate programming approach. We find that the time we spent for the documentation parts is well invested. It leads to a much better understanding of the program and to much better code. Besides being easier to understand and thus being much easier to modify, it is first at all much more likely to be correct. In particular, a literate program takes much less time to debug. The difference between traditional straight forward programming and literate programming is somewhat like the difference between having the idea to a proof of some theorem in mind versus actually writing it down accurately (and thereby often recognizing that the proof is not as easy as one thought).

- **MPI–I–94–106**

  New on–line algorithms for the page replication problem

  Author(s): Susanne Albers, Hisashi Koga

  The page replication problem arises in the memory management of large multiprocessor systems. Given a network of processors, each of which has its local memory, the problem consists of deciding which local memories should contain copies of pages of data so that a sequence of memory accesses can be accomplished efficiently. We present new competitive on-line algorithms for the page replication problem and concentrate on important network topologies for which algorithms with a constant competitive factor can be given. We develop the first optimal randomized on-line replication algorithm for trees and uniform networks; its competitive factor is approximately 1.58. Furthermore we consider on-line replication algorithms for rings and present general techniques that transform large classes of c-competitive algorithms for trees into $2c$-competitive algorithms for rings. As a result we obtain a randomized on-line algorithm for rings that is 3.16-competitive. We also derive two 4-competitive on-line algorithms for rings which are either deterministic or memoryless. All our algorithms improve the previously best competitive factors for the respective topologies.

- **MPI–I–94–110**

  Quickest paths: faster algorithms and dynamization

  Author(s): Dimitrios Kagaris, Grammati E. Pantziou, Spyros Tragoudas, Christos D. Zaroliagis

  Given a network $N = (V, E, c, l)$, where $G = (V, E)$, $|V| = n$ and $|E| = m$, is a directed graph, $c(e) > 0$ is the capacity and $l(e) \geq 0$ is the lead time (or delay) for each edge $e \in E$, the quickest path problem is to find a path for a given source–destination pair such that the total lead time plus the inverse of the minimum edge capacity of the path is minimal. The problem has applications to fast data transmissions in communication networks. The best previous algorithm for the single pair quickest path problem runs in time $O(rm + rn \log n)$, where r is the number of distinct capacities of N. In this paper, we present algorithms for general, sparse and planar networks that have significantly lower running times. For general networks, we show that the time complexity can be reduced to $O(r^*m + r^*n \log n)$, where r^* is at most the number of capacities greater than the capacity of the shortest (with respect to lead time) path in N. For sparse networks, we present an algorithm with time complexity $O(n \log n + r^*n + r^*\tilde{\gamma} \log \tilde{\gamma})$, where $\tilde{\gamma}$ is a topological measure of N. Since for sparse networks $\tilde{\gamma}$ ranges from 1 up to $\Theta(n)$, this constitutes an improvement over the previously known bound of $O(rn \log n)$ in all cases that $\tilde{\gamma} = o(n)$. For planar networks, the complexity becomes $O(n \log n + n \log^3 \tilde{\gamma} + r^*\tilde{\gamma})$. Similar improvements are obtained for the all–pairs quickest path problem. We also give the first algorithm for solving the dynamic quickest path problem.

- **MPI–I–94–111**

  On the width and roundness of a set of points in the plane

  Author(s): Michiel Smid, Ravi Janardan

  Let S be a set of points in the plane. The width (resp. roundness) of S is defined as the minimum width of any slab (resp. annulus) that contains all points of S. We give a new characterization of the width of a point set. Also, we give a *rigorous* proof of the fact that either the roundness of S is equal to the width of S, or the center of the minimum-width annulus is a vertex of the closest-point Voronoi diagram of S, the furthest-point Voronoi diagram of S, or an intersection point of these two diagrams. This proof corrects the characterization of roundness used extensively in the literature.

- **MPI–I–94–112**

  On-line and dynamic shortest paths through graph decompositions (preliminary version)

  Author(s): Hristo N. Djidjev, Grammati E. Pantziou, Christos D. Zaroliagis

  We describe algorithms for finding shortest paths and distances in a planar digraph which exploit the particular topology of the input graph. We give both sequential and parallel algorithms that work on a dynamic environment, where the cost of any edge can be changed or the edge can be deleted. For outerplanar digraphs, for instance, the data structures can be updated after any such change in only $O(\log n)$ time, where n is the number of vertices of the digraph. The parallel algorithms presented here are the first known ones for solving this problem. Our results can be extended to hold for digraphs of genus $o(n)$.

- **MPI–I–94–113**

  Fast algorithms for collision and proximity problems involving moving geometric objects

  Author(s): Prosenjit Gupta, Ravi Janardan, Michiel Smid

  Consider a set of geometric objects, such as points, line segments, or axes-parallel hyperrectangles in $\mathbb{R}^d$, that move with constant but possibly different velocities along linear trajectories. Efficient algorithms are presented for several problems defined on such objects, such as determining whether any two objects ever collide and computing the minimum inter-point separation or minimum diameter that ever occurs. The strategy used involves reducing the given problem on moving objects to a different problem on a set of static objects, and then solving the latter problem using techniques based on sweeping, orthogonal range searching, simplex composition, and parametric search.

- **MPI–I–94–114**

  On–line and dynamic algorithms for shortest path problems

  Author(s): Hristo N. Djidjev, Grammati E. Pantziou, Christos D. Zaroliagis

  We describe algorithms for finding shortest paths and distances in a planar digraph which exploit the particular topology of the input graph. An important feature of our algorithms is that they can work in a dynamic environment, where the cost of any edge can be changed or the edge can be deleted. For outerplanar digraphs, for instance, the data structures can be updated after any such change in only $O(\log n)$ time, where n is the number of vertices of the digraph. We also describe the first parallel algorithms for solving the dynamic version of the shortest path problem. Our results can be extended to hold for digraphs of genus $o(n)$.

- **MPI–I–94–115**

  Efficient construction of a bounded degree spanner with low weight

  Author(s): Sunil Arya, Michiel Smid

  Let S be a set of n points in $\mathbb{R}^d$ and let $t > 1$ be a real number. A t-spanner for S is a graph having the points of S as its vertices such that for any pair p, q of points there is a path between them of length at most t times the euclidean distance between p and q.

An efficient implementation of a greedy algorithm is given that constructs a t-spanner having bounded degree such that the total length of all its edges is bounded by $O(\log n)$ times the length of a minimum spanning tree for S. The algorithm has running time $O(n \log^d n)$. Also, an application to the problem of distance enumeration is given.

- **MPI–I–94–117**

  On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm

  Author(s): Kurt Mehlhorn, Petra Mutzel

  We give a detailed description of the embedding phase of the Hopcroft and Tarjan planarity testing algorithm. The embedding phase runs in linear time. An implementation based on this paper can be found in [Mehlhorn-Mutzel-Naeher-94].

- **MPI–I–94–119**

  Time–space lower bounds for directed s-t connectivity on JAG models

  Author(s): Greg Barnes, Jeff A. Edmonds

  Directed s-t connectivity is the problem of detecting whether there is a path from a distinguished vertex s to a distinguished vertex t in a directed graph. We prove time-space lower bounds of $ST = \Omega(n^2/\log n)$ and $S^{1/2}T = \Omega(mn^{1/2})$ for Cook and Rackoff's JAG model, where n is the number of vertices and m the number of edges in the input graph, and S is the space and T the time used by the JAG. We also prove a time-space lower bound of $S^{1/3}T = \Omega(m^{2/3}n^{2/3})$ on the more powerful node-named JAG model of Poon. These bounds approach the known upper bound of $T = O(m)$ when $S = \Theta(n \log n)$.

- **MPI–I–94–120**

  A method for implementing lock-free shared data structures

  Author(s): Greg Barnes

  We are interested in implementing data structures on shared memory multiprocessors. A natural model for these machines is an asynchronous parallel machine, in which the processors are subject to arbitrary delays. On such machines, it is desirable for algorithms to be *lock-free*, that is, they must allow concurrent access to data without using mutual exclusion. Efficient lock-free implementations are known for some specific data structures, but these algorithms do not generalize well to other structures. For most data structures, the only previously known lock-free algorithm is due to Herlihy. Herlihy presents a simple methodology to create a lock-free implementation of a general data structure, but his approach can be very expensive.

  We present a technique that provides the semantics of exclusive access to data without using mutual exclusion. Using this technique, we devise the *caching method*, a general method of implementing lock-free data structures that is provably better than Herlihy's methodology for many well-known data structures. The cost of one operation using the caching method is proportional to $T \log T$, where T is the sequential cost of the operation. Under Herlihy's methodology, the cost is proportional to $T + C$, where C is the time needed to make a logical copy of the data structure. For many data structures, such as arrays and *well connected* pointer-based structures (e.g., a doubly linked list), the best known value for C is proportional to the size of the structure, making the copying time much larger than the sequential cost of an operation. The new method can also allow *concurrent updates* to the data structure; Herlihy's methodology cannot. A correct lock-free implementation can be derived from a correct sequential implementation in a straightforward manner using this method. The method is also flexible; a programmer can change many of the details of the default implementation to optimize for a particular pattern of data structure use.

- **MPI–I–94–121**

  Short random walks on graphs

  Author(s): Greg Barnes, Uriel Feige

  We study the short term behavior of random walks on graphs, in particular, the rate at which a random walk discovers new vertices and edges. We prove a conjecture by Linial that the expected time to find $\mathcal{N}$ distinct vertices is $O(\mathcal{N}^3)$. We also prove an upper bound of $O(\mathcal{M}^2)$ on the expected time to traverse $\mathcal{M}$ edges, and $O(\mathcal{M}\mathcal{N})$ on the expected time to either visit $\mathcal{N}$ vertices or traverse $\mathcal{M}$ edges (whichever comes first).

- **MPI–I–94–122**

  Realizing degree sequences in parallel

  Author(s): Srinivasa Arikati, Anil Maheshwari

  A sequence d of integers is a degree sequence if there exists a (simple) graph G such that the components of d are equal to the degrees of the vertices of G. The graph G is said to be a realization of d. We provide an efficient parallel algorithm to realize d. Before our result, it was not known if the problem of realizing d is in NC.

- **MPI–I–94–131**

  Hammock-on-ears decomposition:
  A technique for the efficient parallel solution of shortest paths and other problems

  Author(s): Dimitris Kavvadias, Grammati E. Pantziou, Paul Spirakis, Christos D. Zaroliagis

  We show how to decompose efficiently in parallel *any* graph into a number, $\tilde{\gamma}$, of outerplanar subgraphs (called *hammocks*) satisfying certain separator properties. Our work combines and extends the sequential hammock decomposition technique introduced by G. Frederickson and the parallel ear decomposition technique, thus we call it the *hammock-on-ears decomposition*. We mention that hammock-on-ears decomposition also draws from techniques in computational geometry and that an embedding of the graph does not need to be provided with the input. We achieve this decomposition in $O(\log n \log\log n)$ time using $O(n+m)$ CREW PRAM processors, for an n-vertex, m-edge graph or digraph. The hammock-on-ears decomposition implies a general framework for solving graph problems efficiently. Its value is demonstrated by a variety of applications on a significant class of (di)graphs, namely that of *sparse (di)graphs*. This class consists of all (di)graphs which have a $\tilde{\gamma}$ between 1 and $\Theta(n)$, and includes planar graphs and graphs with genus $o(n)$. We improve previous bounds for certain instances of shortest paths and related problems, in this class of graphs. These problems include all pairs shortest paths, all pairs reachability, and detection of a negative cycle.

- **MPI–I–94–136**

  Near–optimal distributed edge coloring

  Author(s): Devdatt Dubhashi, Alessandro Panconesi

  We give a distributed randomized algorithm to edge color a network. Given a graph G with n nodes and maximum degree Δ, the algorithm,

  - For any fixed $\lambda > 0$, colours G with $(1+\lambda)\Delta$ colours in time $O(\log n)$.
  - For any fixed positive integer s, colours G with $\Delta + \frac{\Delta}{(\log \Delta)^s} = (1+o(1))\Delta$ colours in time $O(\log n + \log^{2s} \Delta \log\log \Delta)$.

  Both results hold with probability arbitrarily close to 1 as long as $\Delta(G) = \Omega(\log^{1+d} n)$, for some $d > 0$. The algorithm is based on the Rödl Nibble, a probabilistic strategy introduced by Vojtech Rödl. The analysis involves a certain pseudo–random phenomenon involving sets at the vertices of the graph.

- **MPI–I–94–137**

  Improved parallel integer sorting without concurrent writing

  Author(s): Susanne Albers, Torben Hagerup

  We show that n integers in the range $1 \mathinner{\ldotp\ldotp} n$ can be stably sorted on an EREW PRAM using $O(t)$ time and $O(n(\sqrt{\log n \log\log n} + (\log n)^2/t))$ operations, for arbitrary given $t \geq \log n \log\log n$, and on a CREW PRAM using $O(t)$ time and $O(n(\sqrt{\log n} + \log n / 2^{t/\log n}))$ operations, for arbitrary given $t \geq \log n$. In addition, we are able to sort n arbitrary integers on a randomized CREW PRAM within the same resource bounds with high probability. In each case our algorithm is a factor of almost $\Theta(\sqrt{\log n})$ closer to optimality than all previous algorithms for the stated problem in the stated model, and our third result matches the operation count of the best known sequential algorithm. We also show that n integers in the range $1 \mathinner{\ldotp\ldotp} m$ can be sorted in $O((\log n)^2)$ time with $O(n)$ operations on an EREW PRAM using a nonstandard word length of $O(\log n \log\log n \log m)$ bits, thereby greatly improving the upper bound on the word length necessary to sort integers with a linear time-processor product, even sequentially. Our algorithms were inspired by, and in one case directly use, the fusion trees of Fredman and Willard.

- **MPI–I–94–142**

  The rectangle enclosure and point-dominance problems revisited

  Author(s): Prosenjit Gupta, Ravi Janardan, Michiel Smid, Bhaskar Dasgupta

  We consider the problem of reporting the pairwise enclosures among a set of n axes-parallel rectangles in $\mathbb{R}^2$, which is equivalent to reporting dominance pairs in a set of n points in $\mathbb{R}^4$. For more than ten years, it has been an open problem whether these problems can be solved faster than in $O(n\log^2 n + k)$ time, where k denotes the number of reported pairs. First, we give a divide-and-conquer algorithm that matches the $O(n)$ space and $O(n\log^2 n + k)$ time bounds of the algorithm of Lee and Preparata, but is simpler. Then we give another algorithm that uses $O(n)$ space and runs in $O(n\log n\log\log n + k\log\log n)$ time. For the special case where the rectangles have at most α different aspect ratios, we give an algorithm that runs in $O(\alpha n\log n + k)$ time and uses $O(n)$ space.

- **MPI–I–94–143**

  Some correlation inequalities for probabilistic analysis of algorithms

  Author(s): Devdatt Dubhashi, Desh Ranjan

  The analysis of many randomized algorithms, for example in dynamic load balancing, probabilistic divide-and-conquer paradigm and distributed edge-coloring, requires ascertaining the precise nature of the correlation between the random variables arising in the following prototypical "balls-and-bins" experiment. Suppose a certain number of balls are thrown uniformly and independently at random into n bins. Let X_i be the random variable denoting the number of balls in the ith bin, $i \in [n]$. These variables are clearly not independent and are intuitively negatively related. We make this mathematically precise by proving the following type of correlation inequalities:

  – For index sets $I, J \subseteq [n]$ such that $I \cap J = \emptyset$ or $I \cup J = [n]$, and any non–negative integers t_I, t_J,

  $$\Pr[\sum_{i\in I} X_i \geq t_I \mid \sum_{j\in J} X_j \geq t_J] \leq \Pr[\sum_{i\in I} X_i \geq t_I].$$

  – For any disjoint index sets $I, J \subseteq [n]$, any $I' \subseteq I, J' \subseteq J$ and any non–negative integers $t_i, i \in I$ and $t_j, j \in J$

  $$\Pr[\bigwedge_{i\in I} X_i \geq t_i \mid \bigwedge_{j\in J} X_j \geq t_j] \leq \Pr[\bigwedge_{i\in I'} X_i \geq t_i \mid \bigwedge_{j\in J'} X_j \geq t_j].$$

  Although these inequalities are intuitively appealing, establishing them is non–trivial; in particular, direct counting arguments become intractable very fast. We prove the inequalities of the first type by an application of the celebrated FKG Correlation Inequality. The proof for the second uses only elementary methods and hinges on some *monotonicity* properties.

  More importantly, we then introduce a general methodology that may be applicable whenever the random variables involved are negatively related. Precisely, we invoke a general notion of *negative association* of random variables and show that:

  – The variables X_i are negatively associated. This yields most of the previous results in a uniform way.

  – For a set of negatively associated variables, one can apply the Chernoff-Hoeffding bounds to the sum of these variables. This provides a tool that facilitates analysis of many randomized algorithms, for example, the ones mentioned above.

- **MPI–I–94–144**

  Stochastic majorisation: exploding some myths

  Author(s): Devdatt Dubhashi, Desh Ranjan

  The analysis of many randomised algorithms involves random variables that are not independent, and hence many of the standard tools from classical probability theory that would be useful in the analysis, such as the Chernoff–Hoeffding bounds are rendered inapplicable. However, in many instances, the random variables involved are, nevertheless *negatively related* in the intuitive sense that when one of the variables is "large", another is likely to be "small". (this notion is made precise and analysed in [1].) In such situations, one is tempted to conjecture that these variables are in some sense *stochastically* *dominated* by a set of *independent* random variables with the same marginals. Thereby, one hopes to salvage tools such as the Chernoff–Hoeffding bound also for analysis involving the dependent set of variables. The analysis in [6, 7, 8] seems to strongly hint in this direction. In this note, we explode myths of this kind, and argue that stochastic majorisation in conjunction with an independent set of variables is actually much less useful a notion than it might have appeared.

- **MPI–I–94–145**

  Prefix graphs and their applications

  Author(s): Shiva Chaudhuri, Torben Hagerup

  The *range product problem* is, for a given set S equipped with an associative operator $\circ$, to preprocess a sequence $a_1, \ldots, a_n$ of elements from S so as to enable efficient subsequent processing of queries of the form: Given a pair (s,t) of integers with $1 \le s \le t \le n$, return $a_s \circ a_{s+1} \circ \cdots \circ a_t$. The generic range product problem and special cases thereof, usually with $\circ$ computing the maximum of its arguments according to some linear order on S, have been extensively studied. We show that a large number of previous sequential and parallel algorithms for these problems can be unified and simplified by means of prefix graphs.

- **MPI–I–94–147**

  Efficient collision detection for moving polyhedra

  Author(s): Elmar Schömer, Christian Thiel

  In this paper we consider the following problem: given two general polyhedra of complexity n, one of which is moving translationally or rotating about a fixed axis, determine the first collision (if any) between them. We present an algorithm with running time $O(n^{8/5+\epsilon})$ for the case of translational movements and running time $O(n^{5/3+\epsilon})$ for rotational movements, where ϵ is an arbitrary positive constant. This is the first known algorithm with sub-quadratic running time.

- **MPI–I–94–148**

  Efficient computation of compact representations of sparse graphs

  Author(s): Srinivasa Arikati, Anil Maheshwari, Christos D. Zaroliagis

  Sparse graphs (e.g. trees, planar graphs, relative neighborhood graphs) are among the commonly used data-structures in computational geometry. The problem of finding a compact representation for sparse graphs such that vertex adjacency can be tested quickly is fundamental to several geometric and graph algorithms. We provide here simple and optimal algorithms for constructing a compact representation of $O(n)$ size for an n-vertex sparse graph such that the adjacency can be tested in $O(1)$ time. Our sequential algorithm runs in $O(n)$ time, while the parallel one runs in $O(\log n)$ time using $O(n/\log n)$ CRCW PRAM processors. Previous results for this problem are based on matroid partitioning and thus have a high complexity.

- **MPI–I–94–149**

  Revenge of the dog: queries on Voronoi diagrams of moving points

  Author(s): Oliver Devillers, Mordecai Golin, Stefan Schirra, Klara Kedem

  Suppose we are given n moving postmen described by their motion equations $p_i(t) = s_i + v_i t$, $i = 1, \ldots, n$, where $s_i \in \mathbb{R}^2$ is the position of the i'th postman at time $t = 0$, and $v_i \in \mathbb{R}^2$ is his velocity. The problem we address is how to preprocess the postmen data so as to be able to efficiently answer two types of nearest neighbor queries. The first one asks who is the nearest postman at time t_q to a dog located at point s_q. In the second type a fast query dog is located a point s_q at time t_q, its velocity is v_q where $v_q > |v_i|$ for all $i = 1, \ldots, n$, and we want to know which postman the dog can catch first. We present two solutions to these problems. Both solutions use deterministic data structures.

- **MPI–I–94–150**

  On characteristic points and approximate decision algorithms for the minimum Hausdorff distance

  Author(s): L. Paul Chew, Klara Kedem, Stefan Schirra

  We investigate *approximate decision algorithms* for determining whether the minimum Hausdorff distance between two points sets (or between two sets of nonintersecting line segments) is at most ε. An approximate decision algorithm is a standard decision algorithm that answers YES or NO except when ε is in an *indecision interval* where the algorithm is allowed to answer DON'T KNOW. We present algorithms with indecision interval $[\delta - \gamma, \delta + \gamma]$ where δ is the minimum Hausdorff distance and γ can be chosen by the user. In other words, we can make our algorithm as accurate as desired by choosing an appropriate γ. For two sets of points (or two sets of nonintersecting lines) with respective cardinalities m and n our approximate decision algorithms run in time $O((\varepsilon/\gamma)^2(m+n)\log(mn))$ for Hausdorff distance under translation, and in time $O((\varepsilon/\gamma)^2 mn\log(mn))$ for Hausdorff distance under Euclidean motion.

- **MPI–I–94–153**

  Towards practical permutation routing on meshes

  Author(s): Michael Kaufmann, Uli Meyer, Jop F. Sibeyn

  We consider the permutation routing problem on two-dimensional $n \times n$ meshes. To be practical, a routing algorithm is required to ensure very small queue sizes Q, and very low running time T, not only asymptotically but particularly also for the practically important n up to 1000. With a technique inspired by a scheme of Kaklamanis/Krizanc/Rao, we obtain a near-optimal result: $T = 2 \cdot n + \mathcal{O}(1)$ with $Q = 2$. Although Q is very attractive now, the lower order terms in T make this algorithm highly impractical. Therefore we present simple schemes which are asymptotically slower, but have T around $3 \cdot n$ for *all* n and Q between 2 and 8.

- **MPI–I–94–155**

  Lecture notes: selected topics in data structures

  Author(s): Michiel Smid

  This text contains the lecture notes for the course *Ausgewählte Kapitel aus Datenstrukturen*, which was given by the author at the Universität des Saarlandes during the winter semester 1993/94. The course was intended for 3rd/4th year students having some basic knowledge in the field of algorithm design. The following topics are covered: Skip Lists, the Union-Find Problem, Range Trees and the Post-Office Problem, and Maintaining Order in List.

- **MPI–I–94–156**

  Dynamic algorithms for geometric spanners of small diameter: randomized solutions

  Author(s): Sunil Arya, David M. Mount, Michiel Smid

  Let S be a set of n points in $\mathbb{R}^d$ and let $t > 1$ be a real number. A t-spanner for S is a directed graph having the points of S as its vertices, such that for any pair p and q of points there is a path from p to q of length at most t times the Euclidean distance between p and q. Such a path is called a t-spanner path. The spanner diameter of such a spanner is defined as the smallest integer D such that for any pair p and q of points there is a t-spanner path from p to q containing at most D edges.

  A randomized algorithm is given for constructing a t-spanner that, with high probability, contains $O(n)$ edges and has spanner diameter $O(\log n)$. A data structure of size $O(n \log^d n)$ is given that maintains this t-spanner in $O(\log^d n \log\log n)$ expected amortized time per insertion and deletion, in the model of random updates, as introduced by Mulmuley.

  Previously, no results were known for spanners with low spanner diameter and for maintaining spanners under insertions and deletions.

- **MPI–I–94–158**

  Further improvements of Steiner tree approximations

  Author(s): Marek Karpinski, Alexander Zelikovsky

  The Steiner tree problem requires to find a shortest tree connecting a given set of terminal points in a metric space. We suggest a better and fast heuristic for the Steiner problem in graphs and in rectilinear plane. This heuristic finds a Steiner tree at most 1.757 and 1.267 times longer than the optimal solution in graphs and rectilinear plane, respectively.

- **MPI–I–94–159**

  Accounting for boundary effects in nearest neighbor searching

  Author(s): Sunil Arya, Dave Mount, Onuttom Narayan

  Most analyses of nearest neighbor searching algorithms have been made under the assumption that the number of data points is very large. This assumption simplifies the analysis by eliminating boundary effects, because for any query point the statistical distribution of the data points surrounding it is independent of the location of the query point. This assumption is often not met in applications such as vector quantization, where there is a relationship between dimension and the number of points. We provide an accurate analysis of the number of buckets visited in L_∞ nearest neighbor searching by the bucketing and k-d tree algorithms, assuming 2^d points uniformly distributed in dimension d. Our analysis is tight in the limit as d approaches infinity. Empirical evidence is presented showing that the analysis applies even in low dimensions.

- **MPI–I–94–160**

  Implementation of a sweep line algorithm for the straight line segment intersection problem

  Author(s): Kurt Mehlhorn and Stefan Näher

  We describe a robust and efficient implementation of the Bentley-Ottmann sweep line algorithm based on the LEDA library of efficient data types and algorithms. The program computes the planar graph G induced by a set S of straight line segments in the plane. The nodes of G are all endpoints and all proper intersection points of segments in S. The edges of G are the maximal relatively open subsegments of segments in S that contain no node of G. All edges are directed from left to right or upwards. The algorithm runs in time $O((n+s)logn)$ where n is the number of segments and s is the number of vertices of the graph G. The implementation uses exact arithmetic for the reliable realization of the geometric primitives and it uses floating point filters to reduce the overhead of exact arithmetic.

- **MPI–I–94–162**

  On the parallel complexity of degree sequence problems

  Author(s): Srinivasa R. Arikati

  An integer sequence d is called a degree sequence if there exists a simple graph G such that the degrees of its vertices are precisely the components of d; in that case, G is a realization of d. Given d and an integer k, we study two problems: (i) compute a k-edge-connected realization of d, (ii) compute a k-vertex-connected realization of d. The main contributions of this paper are the first parallel algorithms for these problems. Specifically, we show that problem (i) can be solved in $\tilde{O}(k)$ time using a polynomial number of processors. For problem (ii) we present an efficient algorithm when $k=2$; the algorithm runs in logarithmic time using a linear number of processors.

Part III

The Programming Logics Group

1 Personnel

Director:
Harald Ganzinger

Researchers:
David Basin
Alexander Bockmayr
Michael Hanus (–December 1994)
Seán Matthews
Hans Jürgen Ohlbach
Andreas Podelski (October 94–)
Rolf Socher (–February 1994)

Post-doctoral fellows and Guests:
Penny Anderson (previously at INRIA Sophia Antipolis)
Yannis Dimopoulos (–July 1994; previously at University of Athens)
Peter Madden (previously at the University of Edinburgh)
Sergei Vorobyov (previously at INRIA-Lorraine & CRIN)
Emil Weydert (previously at IMS Stuttgart)

Ph.D. students:
Peter Barth
Hubert Baumeister
Detlef Fehrer (–October 1994)
Peter Graf
Jörn Hopf
Ullrich Hustadt
Manfred Jaeger
Thomas Kasper
Andreas Nonnengart
Renate Schmidt
Georg Struth
Jürgen Stuber
Andreas Tönne
Uwe Waldmann
Christoph Weidenbach
Luca Viganò
Frank Zartmann

Secretaries:
Ellen Fries
Christine Kiesel (March 1994–)
Ellen Schreck (–March 1994)

2 Visitors

Since December 1993, the following researchers (28 in total) have visited, or are visiting, our group.

1994

Dov Gabbay	01.07.91–01.07.95	Imperial College, London
Dragan Cvetkovic	01.07.92–01.01.94	University of Yugoslavia
Witold Charatonik	01.06.93–01.08.93	University of Wroclaw
Leo Bachmair	01.07.93–01.07.94	SUNY at Stony Brook
David Plaisted	01.08.93–01.08.94	University of North Carolina
Leszek Pacholski	22.11.93	University of Wroclaw
Shahid Rahman	01.01.94–01.04.95	Universität des Saarlandes
Jerzy Marcinkowski	01.01.94–01.02.94	University of Wroclaw
Chris Brink	01.01.94–01.02.94	University of Cape Town
Helge Ritter	09.02.94–10.02.94	Universität Bielefeld
Christoph Brzoska	22.02.94	Universität Karlsruhe
Robert Nieuwenhuis	24.02.94–26.02.94	Universidad Politecnica de Cataluna
Ta Chen	01.03.94–01.04.94	SUNY at Stony Brook
Scot Drysdale	01.03.94	Dartmouth College
Janusz Miroforidis	01.04.94–01.06.94	University of Wroclaw
Pierre Lescanne	21.04.94–22.04.94	INRIA-Lorraine & CRIN
Zhenyu Qian	26.04.94	Universität Bremen
Madala Rama Koteswara Krishna Rao	01.05.94–01.08.94	Tata Institute, Bombay
Penny Anderson	19.05.94–20.05.94	INRIA, Sophia-Antipolis
Alex Simpson	17.06.94–25.06.94	University of Edinburgh
Matt Kaufmann	05.07.94	Computational Logic Inc.
Peter B. Andrews	06.07.94	Carnegie Mellon University
Kai Hauser	04.07.94–06.07.94	UCA at Berkeley
Alan Bundy	01.09.94–01.12.94	University of Edinburgh
Ernst-Rüdiger Olderog	05.12.94–17.12.94	Universität Oldenburg

1995

Michael Maher	16.01.95–17.01.95	IBM at Yorktown Heights
Bernhard Steffen	16.01.95–17.01.95	Universität Passau
Tiziana Margaria	16.01.95–17.01.95	Universität Passau

3 Executive Summary

The research unit 'Programming Logics' applies methods of mathematical logic to a variety of problems in computer science. Computation is deduction, a principle that is taken literally in the area of Logic Programming. Formal specifications of software and hardware are formulae in logical systems. Program development and verification is based on proving theorems about specifications and programs. Computation often means to simulate some model of the real world. While in logic programming and in program synthesis and verification one applies, to a large extent, the classical logics known from mathematics, simulation of the real world requires logics that allow us to treat incomplete and changing knowledge and to reason about beliefs, wishes, knowledge, and the like, of their agents. For this purpose so called 'non-classical' logics have to be designed, investigated and applied.

Our work is both theoretical and practical in nature. A large fraction of it is essentially concerned with searching for new and better methods for finding proofs with the support of a computer. As the practical worth of results in this area can often not be judged from the theory alone, we are engaged in various implementation projects in which we try to obtain experimental evidence of the practical potential of our results.

During the last year it has become more and more evident that a unifying theme of the broad spectrum of research within the group is how to exploit mathematical techniques within meta-mathematical reasoning. Axiomatic presentations of theories, like software and hardware systems, are not just unstructured collections of formulae. Often they contain axiomatizations of standard mathematical theories such as numbers, orderings, sets, on top of which other structures of a less fundamental nature are defined. Specific reasoning in non-standard logics might be considered as attempts to distinguish certain structures by moving them to the meta-level. For instance, modal logic might be conceived as an approach to capturing certain interesting classes of binary relations (accessibility relations) within predicate logic formulations. Talking about mathematics, mathematical logic, is meta-mathematics. But at the same time it is, itself, a mathematical discipline. Propositional logic is a logic and also a (Boolean) algebra. For a theorem prover to be useful in practice it should not just implement search in uninterpreted spaces of formulas; it should apply mathematics wherever possible, whereas search should be confined to the uninterpreted symbols of a more ad hoc nature as they correspond to entities of data, programs or hardware that one is specifying. On the level of programming, the CLP approach explicitly applies this principle by offering a well-defined interface between constraint solvers and general execution principles such as resolution or reduction.

Let us briefly survey the major results in our main research areas.

Non-Classical Logics (Hans Jürgen Ohlbach)
In this area we investigate non-monotonic logics, in particular default reasoning or probabilistic reasoning, we look at modal logics as a compromise between efficiency and expressibility, and we study the relation between logics and algebras.

Hans Jürgen Ohlbach has proved a representation theorem for distributive lattices with (extra) function symbols. He has shown how to obtain from the axiomatic specification of such a logic a second-order formula that completely describes all its representations. A wide class of logics can be treated in this way. From the second-order formulation it is often possible to obtain an equivalent first-order presentation automatically by applying the quantifier elimination procedure SCAN that has now been implemented and provided with various interfaces. The first-order formulation may then serve as a basis for proving theorems in the given logic by applying standard methods.

Renate Schmidt has improved de Rijke's representation theorem for Peirce algebras. She has shown that one of his two pre-conditions for the theorem is, in fact, unnecessary.

Andreas Nonnengart has investigated strategies for the finite saturation of first-order translations of certain modal logics. If such a presentation can be finitely saturated, theorem proving within that modal logic becomes much more efficient as inferences between logical axioms are no longer needed. His semi-functional translation method seems to be quite effective in this respect. In some cases, saturated presentations of a logic can be obtained mechanically with the Saturate system.

Manfred Jaeger is pursuing an approach for probabilistic reasoning in which both statistical and subjective probabilities are modelled by probability measures on the domain. He proposes to use cross-entropy as a measure for the dissimilarity of two probability measures, and has been able to justify this choice not only with respect to desired logical behaviour but also from statistical considerations. From an epistemological analysis of default reasoning principles he derives a statistical model that in fact validates the cross-entropy minimization principle for default reasoning about probabilities.

Logic Programming (Alexander Bockmayr, Michael Hanus)
We have continued our research on CLP($\mathcal{PB}$) and on the efficient integration of functions in logic programming by suitably designed program analysis and narrowing strategies.

Michael Hanus, in cooperation with Antoy and Echahed, has exhibited an optimal narrowing strategy for inductively sequential rewrite systems. The approach extends the concept of needed redexes (Huet and Lévy) to narrowing derivations. The result has then been further generalized to other classes of rewrite systems.

LSE narrowing, which is an optimal strategy for a different class of rewrite systems, has been further investigated by Alexander Bockmayr, in cooperation with Andreas Werner. They have generalized LSE Narrowing to conditional term rewriting and shown how implementations can be designed so that repeated tests for irreducibility of substitutions are to a large extent avoided.

In CLP($\mathcal{PB}$) one allows for pseudo-Boolean constraints in logic programs. CLP($\mathcal{PB}$) admits a high-level, natural formulation of optimization problems. 0-1 constraint satisfaction is NP-complete and poses a difficult practical problem. Peter Barth and Alexander Bockmayr have made considerable progress in finding a complete, yet practically useful constraint solving method. To that end they have succeeded in combining polyhedral methods from mathematical programming with simplification techniques from automated deduction. They propose solving constraints by successively computing certain consequences, called cutting planes, thereby eventually converging to certain solved forms of the constraint. Inferences which are at the same time simplifications are preferred. The result is a logic-based and/or polyhedral branch-and-cut algorithm for pseudo-Boolean constraint solving. The method can also be applied with success to certain difficult classes of propositional satisfiability problems.

Higher-Order Logic (David Basin, Seán Matthews)
Under this rather general headline we study at present program synthesis calculi, formalized metatheory and realizations of labelled deductive systems.

FS_0, a logic which is intended for doing metatheory, has been implemented in Isabelle by Seán Matthews. The implementation demonstrates that FS_0 is a usable theory and may be regarded as a successful case study in implementing an unusual logic in a logical framework.

David Basin, Seán Matthews and Luca Viganò have looked at the problem of formalizing "badly behaved" logics in natural deduction-style type theories such as Isabelle or the Edinburgh LF. Labelled deductive systems have been proposed as a mechanism in which "bad behaviour" of a logic can be quite elegantly coded as computation with labels. Our experience with implementing modal logics with the LDS methodology in the Isabelle framework has been very positive. The implementation is modular in the sense that it is independent of the specific properties of the accessibility relations and we expect that this modularity will allow us to define specific efficient tactics for proof search.

David Basin has continued his work on using logical frameworks for formalizing program development calculi. In his formalizations program synthesis and program verifications are just two sides of the same coin. A correctness proof may start with a metavariable for the program in question, and, upon its completion, will then return a correct program as an instantiation for this variable. Higher-order resolution which effects such instantiation remains implicit. Hence he arrives at very simple reformulations of previously proposed calculi in which one is able to also formally prove the correctness of more advanced methods of program development from first principles.

David Basin, in cooperation with Nils Klarlund, has shown how to apply the second-order monadic logic of strings to specifying (and verifying) hardware modules in a way that is independent of their word length, thereby making the regular structure of such devices explicit. Verification is in this case fully automatic and practical because of the use of state-of-the-art OBBD technology inside an implementation of a finite version of S1S, called MONA, that they have done at Aarhus. For instance they have been able to identify and to correct certain errors in a previous specification of a D-type flip-flop.

Automated Theorem Proving for Predicate Logic (H. Ganzinger)
Leo Bachmair and Harald Ganzinger have extended their framework for saturation-based theorem proving for first-order logic with equality to arbitrary transitive relations, in particular partial equivalences, orderings and accessibility relations in modal logics. Superposition was generalized to ordered chaining, an ordered, clausal version of transitivity. They have been able to improve previous theoretical results by Bledsoe, Hines, Shostak, and others considerably. One of the main improvements is that one can now treat several transitive relations simultaneously, even if their additional logical properties are different. In addition, questions of completeness in the presence of simplification techniques have been analysed. The problem of chaining through variables has been explored and positively solved for many interesting cases of transitive relations. These results have been implemented in a theorem prover, Saturate, that has been developed in cooperation with Pilar Nivela and Robert Nieuwenhuis. Initial practical experience that has been obtained with this implementation is very promising.

Leo Bachmair, Harald Ganzinger and Jürgen Stuber have started to combine algebraic methods such as Buchberger's algorithm into their calculi. The goal is to avoid the naive computation with the axioms of an algebraic theory, e.g., the axioms of a commutative ring, but rather compile their effect into specific inference and simplification techniques. Simplification plays a major role in this respect since algebra is mainly a theory of normal forms. As a first step they have extended superposition to the AC-case and proved a stronger form of redundancy to be compatible than other authors. In a second paper they have presented an AC-calculus that is specialized to rings. Top-level superposition inferences and AC-unification are shown to produce much fewer inferences compared to the naive approach. A new problem of chaining through variables appears here, which needs to be further investigated.

Christoph Weidenbach has continued his work on sorted variants of resolution and tableau calculi. He has show that methods that treat type information specifically also contribute to efficient satisfiability checking for certain classes of propositional formulae.

As a primitive basis for implementation of theorem provers, Peter Graf has improved and supplemented his library of data structures for the efficient storing and retrieval of sets of first-order terms with given matching and unifiability properties. His latest idea is a data structure called substitution trees which is the common generalization of discrimination trees and abstraction trees which combines their advantages, while avoiding their disadvantages.

David Plaisted has obtained results on a variety of subjects. For example he has mathematically analysed the search efficiency of a number of commonly used theorem proving strategies. He has introduced a notion of search space complexity and has shown that most strategies have exponential search behaviour, even on simple clause sets. He has also proposed a concept of using orderings that reflect the search complexity of branches in a search within his semantic hyperlinking method. The use of orderings, in addition, provides one with a unification-based way of enumerating ground instances of formulas instead of more naive enumeration that is adopted in pure semantic clause linking otherwise.

4 Journal and Conference Activities

4.1 Editorial positions

The following staff of the programming logics group are editors of various technical journals:
Harald Ganzinger has been an editor of *Information Processing Letters* since November 1990, and an editor of *Mathematical Systems Theory* since June 1992.
Michael Hanus became an editor of the *Journal of Functional and Logic Programming*, published electronically by MIT Press, in 1993.
Hans Jürgen Ohlbach is an editor of *Methods of Logic in Computer Science*, and together with Dov Gabbay (London) and Ruy de Queiroz (Recife, Brasil) founding editor of a new electronic journal, the 'Bulletin of the Interest Group in Pure and Applied Logic'. The first three volumes are accessible via WWW:

`http://www.mpi-sb.mpg.de/guide/staff/ohlbach/igpl/Bulletin.html`

4.2 Conference Positions

4.2.1 Memberships in Organizing Committees

D. Basin Workshop on Correctness and Metatheoretic Extensibility of Automated Reasoning Systems on the 12th International Conference on Automated Deduction, Nancy, 1994.

A. Bockmayr Session on Constraint Programming on the 14th European Conference on Operational Research, Jerusalem, 1995

H. Ganzinger Workshop on Construction of Computational Logics, Rottach-Egern, 1995

M. Hanus Workshop on Integration of Declarative Paradigms, at the International Conference on Logic Programming, Santa Margherita Ligure, 1994

H. J. Ohlbach First international conference on Temporal Logic, Bonn, 1994.
European Summer School in Language, Logic and Information (ESSLI'95), Barcelona.

C. Weidenbach 18th Deutsche Jahrestagung für Künstliche Intelligenz, Saarbrücken, 1994

4.2.2 Memberships in Program Committees

P. Barth Workshop on Computational Propositional Logic, 19th German Conference on Artificial Intelligence, Bielefeld, 1995

D. Basin 3rd Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Abingdon, 1994
5th International Conference on Logic Programming and Automated Reasoning, Crimea, 1994.
12th International Conference on Automated Deduction, Nancy, France, 1994
Theoretical Aspects of Software, Colloquium on Formal Aspects of Software Engineering, Aarhus, 1995.
International Joint Conference on Theory and Practice of Software Development, Aarhus, 1995

H. Ganzinger 10th International Conference on Automated Deduction, Nancy, 1994
10th Annual IEEE Symposium, Logic in Computer Science, Paris, 1994
International Conference on Algebraic and Logic Programming, 1994
1st International Conference on Contraints in Computational Logics, 1994
12th Symposium on Theoretical Aspects of Computer Science, München, 1995

M. Hanus International Logic Programming Symposium, Ithaca, 1994
5th International Conference on Logic Programming and Automated Reasoning, Crimea, 1994
6th International Logic Programming Symposium, Portland, Oregon, USA, 1995.
International Symposium on Programming Languages, Implementation, Logics and Programs, Utrecht, 1995
11th Workshop on Logic Programming, Vienna, 1995.

H. J. Ohlbach 12th International Conference on Automated Deduction, Nancy, 1994
European Conference on Artificial Intelligence, Amsterdam, 1994
1st International Conference on Temporal Logic, Bonn, Germany, 1994
4th Conference on Principles of Knowledge Representation and Reasoning, 1994

C. Weidenbach Workshop on Computational Propositional Logic, 19th German Conference on Artificial Intelligence, Bielefeld, 1995

4.3 Organization of Workshops and Conferences

Hans Jürgen Ohlbach together with Dov Gabbay (London) organized the first International Conference on Temporal Logic (ICTL), the first in a new conference series. It took place in July 11-14, 1994 in the Gustav Stresemann Institute (GSI) in Bonn. There were about 100 participants from more than 20 countries. The proceedings with 34 papers appeared in the Springer Lecture Notes in Artificial Intelligence series, vol 827. The next ICTL will be in 1996 in Manchester, England.

David Basin, with Fausto Giunchiglia (IRST, Italy) and Matt Kaufmann (CLINC, USA) organized the workshop on "Correctness and Metatheoretic Extensibility of Automated Reasoning Systems" held in conjunction with CADE 12, June 1994. The workshop brought together researchers examining the question of how one builds correct theorem proving systems and how one can extend these systems in correctness preserving ways. Over thirty participants attended from within Europe and the United States. Abstracts of the talks can be found in the technical report 9405-10 of the Instituto Per La Ricerca Scientifica E Tecnologica, Trento, Italy.

Jörn Hopf, with Hans-Jürgen Appelrath (University of Oldenburg), Wolfgang Banzhaf (University of Dortmund), Volker Claus (University of Stuttgart), Heinz Mühlenbein (GMD, St. Augustin) and Lothar Thiele (ETH Zürich) organized the workshop 'Genetic Algorithms within the Framework of Evolutionary Computation' held during the conference KI-94, September 1994. More than 20 researchers, mostly from Europe and the United States, met to consider theoretical foundations as well as practical applications of Evolutionary Programming, Evolution Strategies, Artificial Life and Genetic Algorithms. Invited talks were given concerning biological principles and other foundations. All papers and invited talks can be found in the technical report MPI-I-94-241, of our Institute. The abstracts also appeared in 'KI-94 Workshops' by Gesellschaft für Informatik, Bonn.

Michael Hanus, together with Hassan Aït-Kaci (Vancouver) and Juan José Moreno Navarro (Madrid), organized the workshop on 'Integration of Declarative Paradigms' held in conjunction with ICLP'94 in Santa Margherita Ligure, Italy, June 1994. The workshop brought together researchers from the functional and logic programming communities with presentations on various aspects on the integration of logic programming with other declarative paradigms (functions, types, constraints, concurrency etc.). The proceedings are available as an MPI Technical Report (94-224). A special issue with papers from this workshop will appear in the journal 'Computer Languages' in 1995.

5 Teaching Activities

Apart from the core Computer Science course on programming languages, the group also contributes a wide range of general and specialist lectures and seminars to to the logic and computation curriculum organised together with DFKI and the Computer Science Department. (Unless specified, courses were taught at the Universität des Saarlandes.)
Key: L – Lectures, LE – Lectures and exercises, E – Exercises, S – Seminar,
FoPra – Project class.

Winter Semester 1993/1994

Foundations of Program Verification Calculi
D. Basin, S. Matthews – LE
Constraintlösungstechniken A. Bockmayr – LE
Rechnergestütztes Beweisen H. Ganzinger – LE
Deduktion In klassischen und nichtklassischen Logiken
H. J. Ohlbach – LE (at Univ. Darmstadt)
Synthesis using Higher Order Unification
H. Ganzinger, D. Basin – FoPra
Logic programming M. Hanus – L (University of Tartu, Tartu, Estonia)

Summer Semester 1994

Logik und Optimierung A. Bockmayr – LE
Statische Analyse deklarativer Programme M. Hanus – LE
Formale System Spezifikation S. Matthews – L
Term rewriting techniques and automated theorem proving
H. Ganzinger with L. Bachmair (CADE-12 tutorial)
Nichtmonotone Logik E. Weydert – L
Inductive Theorem Proving D. Basin – L
2nd International Summer School in Logic for Computer Science,
University of Chambéry, France
Genetische Algorithmen H.-J. Ohlbach – FoPra

Winter Semester 1994/1995

Verbandstheorie und algebraische Logik H. J. Ohlbach – LE
Programmiersprachen H. Ganzinger – LE

Summer Semester 1995

Lambda-Kalkül und Typtheorie H. Ganzinger, D. Basin – LE
Beweistheorie S. Matthews – L
Logik und Wahrscheinlichkeit E. Weydert – S

6 Dissertations and Habilitations

6.1 Doctorates (to be completed in 1995)

P. Barth *Logic-Based 0-1 Constraint Solving in Constraint Logic Programming* (submitted December 94, defense February 95)
M. Jaeger *A logic for subjective and objective probabilities* (to be submitted July, 95).
A. Nonnengart, *A Resolution-Based Calculus for Temporal Logics* (to be submitted February, 95)
P. Graf, *Term Indexing*, (to be submitted August, 95)

6.2 Habilitations

M. Hanus.
D. Basin.

6.3 Masters Theses in Progress

Ayari Abdelwaheb: *Program Synthesis with Higher-Order Unification* under D. Basin.
Alexander Bach: *Lineare Logik als Typsystem für funktionale Programme* under D. Basin and A. Tönne.
Michael Christen: *N.N.* under H. Ganzinger and J. Stuber.
Thorsten Engel: *Quantorenelimination in Prädikatenlogik 2. Stufe* under H. J. Ohlbach.
Hichem Harakirti: *N.N.* under P. Graf.
Christoph Meyer: *Verteilte Hyperresolution* under P. Graf.
Erik Mohr: *Resolutionskalküle für Modallogiken* under A. Nonnengart.
Georg Rock: *Transformations of first-order formulae for automated reasoning* under C. Weidenbach.
Stefan Schlobach; *N.N.* under H. J. Ohlbach.
Jan Smaus: *Killer Transformation* under H. J. Ohlbach.
Jan Timm: *Testing of the satisfiability of ordering constraints* under C. Weidenbach.

6.4 Masters Theses

Joachim Becker: *Effiziente Subsumption in Deduktionssystemen* under P. Graf.
Thomas Schanne: *Vier gewinnt, ein Fallbeispiel zur Programmsynthese mit genetischen Algorithmen* under H. J. Ohlbach.

7 Grants

CCL: Construction of Computational Logics

Description

The past two decades have seen a proliferation of different programming styles: functional, logical, constraint-based, object-oriented, among others. More recently, it has been recognized that these styles complement rather than exclude each other by being suitable for particular problem domains. As a consequence, combining programming paradigms has emerged as a significant research direction of its own. Fortunately the modes of computation are in each case firmly based on logic. Computation means simplifying or solving problems represented by logical formulae. Hence combination of programming paradigms means combination of logics in a common logical framework.

Constraint logic programming has been the first entirely successful step towards this ambition of combining logics. Constraints are logical systems specifically tailored to particular theories, for example, numbers, trees, orderings. Constraints allow convenient notations for particular problem domains, efficiency thanks to dedicated solvers, and modularity by isolating the solver from the purely logical part of the computation. Besides new ways of exploiting the above properties, we are also interested in new applications of constraints. Exploiting the structure of (large) constraint logic programs for automating proofs that would simply fail otherwise belongs to the first category. Investigating the use of new systems of constraints for type checking purposes or search guiding information belongs to the second.

According to the above two basic lines, the aims of the CCL-project are the following:

- to investigate specific instances of combination problems for logics and constraints of particular interest;

- to investigate new constraints and to design algorithms for combining existing constraint systems;

- to develop or improve theorem proving techniques for certain logics of special importance for programming, by taking advantage of constraint systems;

- to contribute to a coherent framework for combining programming paradigms and other logical theories, thus enabling the programmer to combine elements of each of them in a unified environment.

Technical Data

Starting date:	July 24, 1992
Duration:	3 years
Funding:	ESPRIT Basic Research Action
Staff at MPI f. Informatik:	Peter Barth Alexander Bockmayr Harald Ganzinger Michael Hanus Uwe Waldmann

Partners

Cosytec, Paris; DFKI, Saarbrücken; INRIA-Lorraine, Nancy; TU München; CIS, Univ. München; LRI, Université Paris–Sud; RWTH Aachen; Universidad Complutense de Madrid; Universitat Politécnica de Catalunya.

COMPASS: A Comprehensive Algebraic Approach to System Specification and Development

Description

In software technology, concepts, methods and development environments for the construction of data-processing systems from self-contained, generic and reusable components are becoming increasingly important, if not mandatory. The decomposition of systems supports a breakdown of the production process into feasible tasks. Generic and reusable components help to avoid duplications of effort, to ease prototyping, testing and verification and to speed up production processes. The industrial production of generic and reusable software components, however, is only possible under certain conditions:

- The requirements on a component must be specifiable in a precise way.
- The functional behaviour of a component must be determined in a precise way.
- For each component, especially for a critical one, the correctness (meaning that the behaviour satisfies the requirements) must be provable.
- The integration of components into large systems must be supported in such a way that the behaviour and the correctness of the components are preserved.

The state of the art in software technology does not yet allow the systematic development of system components that meet the four demands above. In particular, tools are missing that support such a development with strong requirements on the correctness of the components properly and fully. The algebraic approach to system specification and development is most promising in this respect, but there is still a broad gap between the state of the art of the algebraic approach and the practical needs of the specification of system components.

The main objective of the Basic Research Working Group `compass` is to bridge this gap by further development and consolidation of the algebraic approach in a comprehensive way. In spite of more than fifteen years of research in the algebraic approach, there is a considerable need of clarification, unification, extension and integration of other programming and specification paradigms. Most partners are involved in national and/or European-funded projects, and much of the work described in the objectives will be done in the context of these other projects. The purpose of `compass` is largely to provide opportunities for interaction between the members of these separate projects.

Technical Data

Starting date:	1992
Duration:	3 years
Funding:	ESPRIT Basic Research Working Group
Staff at MPI f. Informatik:	Harald Ganzinger Hubert Baumeister

Partners

Århus Universitet; Universitat Politécnica de Catalunya, Barcelona; Technische Universität Berlin; Technische Universität Braunschweig; Universität Bremen; Technische Universität Dresden; University of Edinburgh; Universitá di Genova; Universitá degli Studi de L'Aquila; INESC, Lisboa; Technische Universität München; CRIN, Nancy; University of Nijmegen; University of Oslo; Oxford University; CNRS - Université de Paris-Sud; Ecole Normale Supérieure/CNRS, Paris; Ludwig-Maximilians-Universität, München.

MEDLAR II: MEchanizing Deduction in the Logics of prActical Reasoning

Description

MEDLAR II builds on the success of the original MEDLAR project in mechanizing logics of practical reasoning to handle time, action, belief, knowledge and intent. In MEDLAR II the concept of a practical reasoner will be developed, an agent capable of acting autonomously and interacting flexibly with its real world environment. Specific reasoning capabilities are being synthesised for combinations of logics within a general framework for knowledge representation, so that examples of reasoning in natural language dialogue and the planning of robots can be demonstrated.

Technical Data

Starting date:	24 July 1992
Duration:	3 years
Funding:	ESPRIT Basic Research Action 6471
Staff at MPI f. Informatik:	Hans Jürgen Ohlbach Christoph Weidenbach

Partners

RISC-Linz; Technische Hochschule Darmstadt; Universität München; ONERA-CERT, Toulouse; Institut Nationale Polytechnique de Grenoble; Universite Paul Sabatier, Toulouse; Universita di Torino; University of Oslo; The Imperial College of Science, Technology and Medicine, London; International Computers Limited.

Detecting Redundancy of Clauses and Inferences

Description

Saturation transforms a set of first-order formulae (with equality) into a representation of the theory that makes further theorem proving much more efficient. Saturated sets of axioms may be used both in a purely goal-oriented way and with ordering restrictions, without loosing completeness. However, saturation terminates only if the overwhelming majority of inferences can be proved to be redundant, such that they do not give rise to new formulae. It is the goal of this project to derive practically useful criteria for the redundance of clauses and inferences and to investigate the degree to which powerful redundancy criteria can turn ordered paramodulation into an efficient decision procedure.

Technical Data

Starting date:	1.7.1992
Duration:	2 years (ended: 30.06.94)
Funding:	DFG Schwerpunkt Deduktion
Staff at MPI f. Informatik:	Harald Ganzinger Uwe Waldmann

Partners

The project is a part of the 'Schwerpunktprogramm Deduktion', the partners of which include: Universität Kaiserslautern; Technische Hochschule Darmstadt; Technische Universität München; Universität Koblenz-Landau; Humboldt-Universität Berlin; Technische Hochschule Darmstadt; Technische Universität Berlin; Technische Universität München; Universität-GH Paderborn; Technische Hochschule Darmstadt; Universität Tübingen; Ludwigs-Maximilians-Universität München; Universität Karlsruhe; Universität Leipzig; Universität des Saarlandes; Technische Universität Braunschweig; Universität Karlsruhe; Universität Tübingen; Universität München; Technische Hochschule Darmstadt.

LOGO: Logic Engineering

Description

The subject of the Logo project is the development of techniques for developing, investigating and automating application oriented logics.

There are several workpackages. In the first workpackage we develop methods for synthesising from a Hilbert calculus specification a model theoretic semantics, and from this semantics a translation method can be derived which translates formulae of the new logic into predicate logic. Certain optimizations of the semantics can be performed such that the resulting translation into predicate logic allows for the application of special theory resolution rules which represent the characteristics of the logic.

In the second workpackage we investigate higher order type theory and higher order logics as meta systems for implementing 'object logics' as well as complex software systems.

In another workpackage a hybrid system is to be developed which allows for the combination of various inference and control techniques.

Technical Data

Starting date:	1.9.1991
Duration:	finished 31.12.1994
Funding:	BMFT
Staff at MPI f. Informatik:	David Basin Harald Ganzinger Ullrich Hustadt Hans Jürgen Ohlbach Seán Matthews Andreas Tönne

Automation of Proof by Mathematical Induction

Description

Mathematical induction is required for reasoning about objects or events containing repetition, e.g. computer programs with recursion or iteration, electronic circuits with feedback loops or parameterised components. Thus mathematical induction is a key enabling technology for the use of formal methods in information technology. The goal of this collaboration is to permit an exchange of ideas and cross-fertilization between the leading research groups in this field. The collaboration takes the form of research visits between individuals working on similar or identical problems and more generally annual project seminars. The collaboration is addressing research topics including synthesis of induction rules, generalization, conjecturing of lemmata, strategic search guidance, and applications.

Technical Data

Starting date:	August 1993
Duration:	2 years (1 year extension applied for)
Grant:	DAAD/British Council Academic Research Collaboration Grant
Staff at MPI f. Informatik:	David Basin Seán Matthews Luca Viganò

Partners

University of Edinburgh; Universität des Saarlandes; Universität Darmstadt.

ACCLAIM: Advanced Concurrent Constraint Languages: Application, Implementation, and Methodology

Description

The purpose of this project is to further the conceptual, mathematical and practical foundations for concurrent constraint programming, and in so doing, provide a framework for, design and implement advanced computational tools for the development of complex, symbolic computational tasks. The objectives of the four work-packages are:

- To extend the foundations of concurrent constraint programming to account for a substantially richer class of computational phenomena, and to establish connections with graph-grammars.
- To develop efficient constraint techniques to tackle new application areas and to produce extensible general-purpose constraint systems, reactive (incremental) constraint solving, and hypothetical reasoning.
- To develop frameworks and techniques for compile-time analysis and optimization of concurrent constraint programs, to allow efficient execution of programs on a wide variety of target architectures.
- To improve the implementation technology of concurrent constraint languages to be competitive with imperative languages, such as C, on single-processor architectures; and to

achieve a high degree of parallel execution on a wide variety of multi-processor architectures.

Technical Data

Starting date:	1 September 1992
Duration:	3 years
Foundation:	ESPRIT Basic Research Action 7195
Staff at MPI f. Informatik:	Peter Barth
	Alexander Bockmayr
	Andreas Podelski

Partners

Swedish Institute of Computer Science; Deutsches Forschungsinstitut für Künstliche Intelligenz; INRIA; Katholieke Universiteit Leuven; Universitat Politécnica de Madrid; Università di Pisa; Université d'Aix-Marseille II; RISC-Linz.

PROCOPE: Construction of Non-Classical Logics

Description

In computer science and in particular in the area of artificial intelligence there is an increasing need for logics which allow to reason with knowledge, time and in fact any kind of modality and conditional.

The aim of this project is to develop logic engineering systems which support the examination of such logics and which allow to find suitable and efficient corresponding calculi.

Technical Data

Starting date:	January 1993
Duration:	3 years
Funding:	PROCOPE Programme,
	Deutscher Akademischer Austauschdienst
Staff at MPI f. Informatik:	Hans Jürgen Ohlbach
	Andreas Nonnengart
	Ullrich Hustadt
	Christoph Weidenbach
	Renate Schmidt
	Emil Weydert

Partners

Université Paul Sabatier; Institut de Recherche CNRS.

EDDS: Efficient Data Structures for Deduction Systems

Description

All operations in deduction systems (inference rules, deletion rules, simplifications, and so on) are defined as operations on single objects. Therefore, the performance of a theorem prover crucially depends upon the speed of the basic retrieval operations, such as finding terms that are unifiable with (instances of, or more general than) some *query term*. In order to find resolution partners for a given literal, for example, a theorem prover has to find unifiable literals. Subsumption of clauses can be detected by the retrieval of generalizations or instances of literals of clauses. Even the retrieval of rewrite rules, demodulators, and paramodulants can be accelerated by indexing if the indexing mechanism also supports retrieval in the subterms of the indexed term set.

The importance of the usage of indexing has been shown by the OTTER theorem prover. Due to the consequent usage of Path–Indexing and Discrimination Tree Indexing, this prover became one of the most powerful and fastest deduction systems.

We are developing and implementing new methods such as extended Path–Indexing and Abstraction Tree Indexing in order to speed up term retrieval. Our software is currently being used in the deduction systems STOP (MPI Saarbrücken) and SETHEO (TU München). Very soon we will also embed it into the provers KEIM (Universität des Saarlandes) and DISCOUNT (Uni Kaiserslautern).

Technical Data

Starting date:	1 June 1992
Duration:	3 years (possibly 1 year more)
Funding:	DFG Schwerpunkt Deduktion
Staff at MPI f. Informatik:	Hans Jürgen Ohlbach
	Peter Graf

Partners

The project is a part of the 'Schwerpunktprogramm Deduktion' (Az. 322698). Partners are the same as for the grant 'Detecting Redundancy of Clauses and Inferences.

SOFTI II: Logic of Programming

Description

This project addresses the development of methods and techniques for the efficient construction of reliable software, that is programs that meet their specifications. Reliability is difficult to ensure in practice, especially when large programs comprising different application areas are integrated. Our research investigates the modular combination of various logics and programming paradigms where problems and programs are abstractly stated. Modularity means that the complexity of large systems can be decomposed into manageable pieces and moreover these components may be reused in other domains. This task is especially complex as we do not wish to restrict the kinds of logics and programming languages that may be used. A high-level abstraction during the formalization of functional specifications allows a better control over the 'logic complexity in the small' at the level of the individual system components.

Technical Data

Starting date:	September 1, 1991
Duration:	3 years (ended: 31.012.94)
Funding:	German Ministry for Research and Technology (BMFT)
Staff at MPI f. Informatik:	Peter Barth Hubert Baumeister Alexander Bockmayr Harald Ganzinger Michael Hanus Rolf Socher Jürgen Stuber

TRALOS - Transformation of Logical Systems

Description

We want to investigate methods for finding more or less automatically transformations of axiom systems such that theorems are provable from the original axioms if and only if the transformed theorems are provable from the transformed axioms. Of course the transformations should be such that proving transformed theorems is easier than proving the original theorems. One class of transformations we are investigating are based on representation theorems for algebras.

Technical Data

Starting date:	March 1995
Duration:	2 years
Funding:	DFG
Staff at MPI f. Informatik:	Hans Jürgen Ohlbach Renate Schmidt

CONSOLE: Constraint Solving in Europe

Description

Constraint solving has become more and more important in automated theorem proving, logic programming and algebraic specifications, as well as for industrial applications like constraint programming in CHIP.

The goal of this project is to facilitate the interactions between European research teams in the field of constraint solving, especially concerning visits and exchanges of young researchers. The work is mainly focused on symbolic constraints (i.e. logic formulae interpreted in some tree structure) and on the application of constraints to constraint logic programming languages. We are thereby particularly interested in equality constraints, ordering constraints, membership constraints, set constraints, and combinations of constraint solvers.

The MPI-part in this project is mainly concerned with

- Paramodulation and Superposition Calculi
- Set Constraints and the Monadic Class
- Non-Linear Constraints in CLP($\mathcal{R}$)

- 0-1 Constraints in CLP($\mathcal{PB}$)

Technical Data

Starting date:	Contract has been signed in December 1994
Duration:	2 years
Funding:	Human Capital and Mobility
Staff at MPI f. Informatik:	Peter Barth Alexander Bockmayr Harald Ganzinger Andreas Podelski

Partners

University of Barcelona; University of Lille; ECRC, München; INRIA Lorraine; University of Orsay; Cosytec, Orsay; University of Padova.

8 Research Areas

In the following section we describe the work of the group over the last year. In the bibliographies that accompany each part, entries marked with a bullet '•' in the margin are papers of some kind (official or unofficial) that have been produced by members of the group in the last year.

8.1 Automated Theorem Proving for Predicate Logic

A major goal in this area has been to extend our previous work on saturation-based theorem proving into various directions, the main emphasis being on the efficient support for basic mathematical structures such as equality, orderings, abelian monoids, groups and rings (cf. Sections 8.1 and 8.1). Besides this, and among others, we have continued to work on efficient indexing data structures for terms (cf. Section 8.1.5), on exploiting type structure in first-order theories (cf. Section 8.1.3), and on constraint solvers for certain classes of symbolic constraints (cf. Section 8.1.4). In addition to our practical work on term indexing many of our theoretical results that are described below in more detail have been implemented and to some extent experimentally verified in the form of several prototype provers, cf. Section 8.6.6.

8.1.1 Saturation-Based Theorem Proving

Investigators: Leo Bachmair, Harald Ganzinger, Georg Struth, Jürgen Stuber, Uwe Waldmann

Saturation-based methods compute the closure of a set of formulas under a given inference system. The inference systems are designed in a way such that a saturated set is either obviously inconsistent (e.g. contains $\perp$) or is satisfiable. Saturation-based methods are believed to be not as goal-oriented as other methods, e.g. semantic tableau; however they seem to provide better means for integrating procedures specific to particular mathematical theories. In most applications of theorem proving one has to deal with function symbols such as $+$ or $<$ that have a fixed interpretation with respect to some structure (e.g. the integers) or a specific mathematical theory (e.g. ordered groups) in addition to arbitrary other uninterpreted symbols. A non-naïve treatment of the interpreted symbols is crucial to the performance of the prover, and to this end mathematical and meta-mathematical techniques have to be combined.

Our method may be characterized as follows. First, we apply techniques from term rewriting, thereby freely varying the notion of a rewrite proof, depending on the mathematical theory we want to support. The search for a rewrite proof is a strongly restricted form of goal-oriented backward reasoning. To make it complete, it has to be accompanied by a controlled forward reasoning which changes the presentation of the theory. Saturation proof techniques, based on the notion of a rewrite proof, form a compromise between the purely goal-oriented methods on one side and blind forward reasoning on the other side.

So as to be able to control forward reasoning further, our inference systems are designed to be compatible with some abstract notion of redundancy for formulas and inferences. Only non-redundant inferences and formulas need to be considered. In contrast with side-conditions attached to inferences, redundancy is a non-local property depending on the complete set of formulae at hand. The major application of redundancy is for checking the compatibility of simplification techniques such as reduction by equations, removal of tautologies (modulo the supported theory), subsumption, or quantifier elimination with the given inference system. Compatibility is meant in the sense that the eager or don't-care nondeterministic application of a simplification step does not affect the refutational completeness of the inference system.

Then, and in contrast to constraint logic programming, decision procedures for decidable theories such as abelian groups, dense total orderings, or real closed fields cannot simply be linked to a prover as black boxes [1]. In fact, in the presence of arbitrary uninterpreted functions and predicates the theory usually becomes undecidable. Integration has to be achieved on the level of the inference system and simplification techniques. Specific inference systems and simplification techniques have to be designed to incorporate the basic ideas in a decision procedure. Our general methodology for model-theoretic completeness proofs based on rewrite proofs for atoms greatly helps in the design of these systems.

References

[1] R. S. Boyer and J. S. Moore. Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic. In J. E. Hayes, D. Michie, and J. Richards, editors, *Machine Intelligence 11*, chapter 5, pages 83–124. Clarendon Press, Oxford, 1988.

Transitive Relations

Transitive relations are ubiquitous. In logical terms, a transitive relation is the most primitive form of a replacement (rewriting) system. It is surprising that in the context of automated deduction, rewrite techniques have almost exclusively been applied only to congruences (equality). The first notable exception is the bi-rewriting of Levy and Agustí [5] conceived as the basis of a Knuth/Bendix-like completion procedure for presentations of lattices by inequalities. In [1] and [2] we apply related ideas to refutational theorem proving for arbitrary first-order theories that include one or more, possibly interacting, transitive relations. Transitivity is a particular form of a composition law of the form $R \circ S \subseteq T$ for binary relations R, S, and T, and our techniques allow us to deal with a large class of families of such laws efficiently. An appropriate notion of a rewrite proof for inequalities provides an ordering-restricted form of chaining inference in which a chain $T(s,t)$ from $R(s,u)$ and $S(u,t)$ is computed only if u is strictly maximal among the three terms with respect to a given syntactic ordering. As a consequence, chaining through a variable, the most prolific form of chaining, is not required if the variable is *shielded*, i.e. also occurs as an argument to a function or predicate within the same clause. A shielded variable cannot be maximal. In [2] we present ordered-chaining inference systems for (arbitrary) transitive relations, partial equivalences, and congruences and point out the relations between them. These systems mainly differ in the way they handle disjunctions and in the extent to which chainings through variables can be avoided. The systems are refutationally complete and compatible with redundancy. In [1] we concentrate on partial and total orderings. We improve previously obtained theoretical results by Bledsoe, Hines, and Shostak [3, 4] in many respects: For instance we have succeeded in combining chaining for orderings with superposition for equality. (In the presence of a non-strict ordering, equations arise naturally as conjunctions of inequations.) Moreover we have been able to prove the compatibility not only of the usual simplification techniques, but also of quantifier elimination techniques for dense total orderings without endpoints. As a result chainings through variables can be avoided completely, a property which is also satisfied by our systems with explicit equality and in the presence of arbitrary uninterpreted functions and predicates. This is our first result about the integration of a decision procedure into an in general undecidable context. Not only can we decide the decidable subcases, but in the undecidable case we manage a considerable restriction of the search space.

References

- • [1] L. Bachmair and H. Ganzinger. Ordered chaining for total orderings. In *Proc. 12th International Conference on Automated Deduction*, LNAI, pages 435–450. Springer, 1994. Full version available as MPI-I-93-250.

- • [2] L. Bachmair and H. Ganzinger. Rewrite techniques for transitive relations. In *Proc. 9th IEEE Symposium on Logic in Computer Science*, pages 384–393. IEEE Computer Society Press, 1994. Full version available as Technical Report MPI-I-93-249.

- [3] W. Bledsoe, K. Kunen, and R. Shostak. Completeness results for inequality provers. *Artificial Intelligence*, 27:255–288, 1985.

- [4] L. M. Hines. Completeness of a prover for dense linear logics. *Journal of Automated Reasoning*, 8:45–75, 1992.

- [5] J. Levy and J. Agustí. Bi-rewriting, a term rewriting technique for monotonic order relations. In C. Kirchner, editor, *Rewriting Techniques and Applications*, volume 690 of *Lecture Notes in Computer Science*, pages 17–31, Berlin, 1993. Springer.

Integrating Algebra and Universal Algebra

The associativity and commutativity of binary operations such as $+$ and $*$ is often built into the notation that mathematicians use. Therefore, as the first step towards an integration of computer algebra methods into first-order theorem proving one has to extend the techniques we have just mentioned to deduction with formulae modulo AC. In [1] we have extended the superposition calculus to the AC-case. The main contribution of this paper compared with related work (e.g. [13, 11, 12]) is the relatively simple proof of completeness in the presence of a specific notion of redundancy for clauses and inferences. This seems more appropriate than similar notions of redundancy, such as provided by [13, 11] in that head reduction of terms in clauses modulo AC by equations is among the simplifications that one can prove admissible with our notion. This simplification is an essential ingredient when specializing the calculus to algebraic theories such as groups and rings.

In [2] we investigate the relation between AC-completion and Buchberger's algorithm for constructing the Gröbner basis for a polynomial ideal. The similarity between Buchberger's algorithm and completion appears to have been first observed by Buchberger and Loos [9, 6]. Since then a number of researchers have attempted to clarify, and formalize, the connection between the two methods; see [5] for a discussion of the fundamental issues and further references. The typical approach has been to generalize rewrite-based procedures so that both Buchberger's algorithm and (certain variants of) completion can be derived as special cases. Most of these results apply only to variants of Buchberger's algorithm for polynomials over a commutative ring though, and not to the original algorithm, with a field as the coefficient domain. Only recently has Bündgen [7] shown how to simulate polynomial computations over *finite* fields by associative-commutative completion. We observe that Buchberger's algorithm is not based only on rewriting; in particular, many operations on the coefficients of a polynomial typically do not involve rewriting and should hence be modeled by constraint solving in the context of a hierachical approach to theorem proving, as we have studied it in [4]. We have shown that Buchberger's algorithm is a specific instance of the latter method and hence can, and should, be viewed as a completion procedure for hierarchical equational theories. Unlike previous methods, our results apply also to coefficient fields.

Buchberger's algorithm may be used to decide the word problem in commutative rings. In [3] we consider refutational theorem proving for first-order theories containing the axioms of a commutative ring. Our approach is inspired by the Gröbner basis method, where the essential inferences are generalized into specialized superposition inferences and/or simplification steps. Starting out from a particular canonical AC-rewrite system R for rings we analyse the effect of saturating sets of the form $R \cup \{C\}$, where C is an arbitrary ground clause and code that into certain macro inferences by which clauses are symmetrized [8]. Inferences with symmetrized clauses include the computation of S-polynomials in the Gröbner basis algorithm as an essential ingredient. Essentially, this allows us to replace most explicit inferences between the ring axioms and other clauses with simplification. Inferences between any two symmetrized non-ring axioms tend to produce far fewer new clauses and, in particular, do not always need full AC-unification. Our method is also an extension of work by [10] who considers (possibly failing) Knuth-Bendix completion procedures for equations that contain a particular theory represented by a convergent rewrite system. Our results are preliminary in many respects: the concept of symmetrization and the completeness proofs should be technically simplified, and there is, again, a severe problem of chaining through, and even below, variables for which one should try to find effective restrictions and control strategies.

References

• [1] L. Bachmair and H. Ganzinger. Associative-commutative superposition. Technical Report MPI-I-93-267, Max-Planck-Institut für Informatik, Saarbrücken, 1993. To appear in Proc. CTRS Workshop 1994, LNCS.

• [2] L. Bachmair and H. Ganzinger. Buchberger's algorithm: a constraint-based completion procedure. In *1st Internal Conference on Constraints in Computational Logics*, volume 845 of *Lecture Notes in Computer Science*, pages 285–301. Springer-Verlag, 1994.

• [3] L. Bachmair, H. Ganzinger, and J. Stuber. Combining algebra and universal algebra in first-order theorem proving: The case of commutative rings. In *Proc. 10th Workshop on Specification of Abstract Data Types*, LNCS. Springer, 1995. To appear.

• [4] L. Bachmair, H. Ganzinger, and U. Waldmann. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing*, 5(3/4):193–212, Apr. 1994.

[5] B. Buchberger. History and basic features of the critical pair/completion procedure. *Journal of Symbolic Computation*, 3:3–38, 1987.

[6] B. Buchberger and R. Loos. Algebraic simplification. In *Computer Algebra: Symbolic and Algebraic Computation*, pages 11–43. Springer, 2nd edition, 1983.

[7] R. Bündgen. Simulating Buchberger's algorithm by a Knuth-Bendix completion procedure. In R. V. Book, editor, *Proc. Fourth International Conference on Rewriting Techniques and Applications*, volume 488 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[8] P. Le Chenadec. Canonical forms in finitely presented algebras. In *Proc. 7th Int. Conf. on Automated Deduction*, volume 170 of *LNCS*, pages 142–165. Springer, 1984. Book version published by Pitman, London, 1986.

[9] R. Loos. Term reduction systems and algebraic algorithms. In *Proc. 5th GI Workshop on Artificial Intelligence*, volume 47 of *Informatik Fachberichte*, pages 214–234. Springer, 1981.

[10] C. Marché. Normalised rewriting and normalised completion. In *Proc. 9th Ann. IEEE Symp. on Logic in Computer Science*, pages 394–403. IEEE Computer Society Press, 1994.

[11] R. Nieuwenhuis and A. Rubio. AC-superposition with constraints: No AC-unifiers needed. In *Proc. 12th International Conference on Automated Deduction*, volume 814 of *Lecture Notes in Computer Science*, pages 545–559, Berlin, 1994. Springer-Verlag.

[12] L. Vigneron. Associative-commutative dedution with constraints. In *Proc. 12th International Conference on Automated Deduction*, volume 814 of *Lecture Notes in Computer Science*, pages 530–544, Berlin, 1994. Springer-Verlag.

[13] U. Wertz. First-order theorem proving modulo equations. Technical Report MPI-I-92-216, Max-Planck-Institut für Informatik, Saarbrücken, 1992.

8.1.2 Confluent Rewriting Systems

Investigator: M. R. K. Krishna Rao

Confluence and normalization are two important properties of term rewriting systems. There can be different notions of confluence depending on the reduction strategies employed. Innermost-confluence and outermost-confluence are two such notions. Innermost-confluence is important in giving call-by-value and denotational semantics and outermost-confluence ensures well-definedness of lazy semantics of functional programs. These two restricted notions do not coincide with the general notion of confluence, even for terminating systems. Recently, we proposed a set of sufficient conditions under which the properties of confluence, innermost-confluence and outermost-confluence coincide. The property of semi-completeness (confluence + weak normalization) is very useful in establishing consistency of equality theories. In a recent paper, we established a result on modularity of semi-completeness for hierarchical combinations, generalizing all the existing results.

References

• [1] M. Krishna Rao. Relating confluence, innermost-confluence and outermost-confluence properties of term rewriting systems. Submitted to Acta Informatica, 1994.

• [2] M. Krishna Rao. Semi-completeness of hierarchical and super-hierarchi cal combinations of term rewriting systems. In M. Nielsen, editor, *Trees in Algebra and Programming – CAAP'95*. Springer, 1995.

8.1.3 Order-Sorted Logics

Investigator: Christoph Weidenbach

Exploiting sort restrictions is a very effective way of improving the performance of an automated theorem prover. The basic idea is to restrict the instantiation of variables by attaching sorts to variables. In our approach [6] sorts are sets of unary predicates interpreted as the intersection of the denotations of the unary predicates. If a variable is instantiated by some term in order to apply an inference rule, we require the term to respect the sort of the variable. This prevents redundant inference steps. In addition, clauses with sorted variables can be translated into standard clauses and vice versa, but a clause with sorted variables corresponds to infinitely many standard clauses in the sense that (in general) a refutation using clauses with sorted variables abbreviates infinitely many ground refutations of the corresponding standard clauses.

Traditional approaches to sorted reasoning [5, 1, 3, 4, 2] require the a priori presence of the sort restrictions, assume sorts to be non-empty and do not allow the occurrence of positive sort literals, called *declarations*, in the clause set under consideration. In our approach we do not assume any of these restrictions [6]. An arbitrary standard first-order clause set can be compiled into a clause set with explicit sort restrictions. The sorts may denote empty sets and declarations may occur arbitrarily in the compiled clause set. We extend the standard resolution and free variable tableaux calculi by replacing standard unification with sorted unification. The two calculi generalize their respective standard versions. We have examples which can only be solved by the extended calculi and not by any other theorem proving technology.

We have obtained new results for the free variable tableaux case [7]. The sorted unification algorithm sometimes solves the problem of formula copies for declarations. If sorted unification is decidable for the sort theory of a tableau branch, the branch is closed via sorted unification where the possibly missing copies of declarations are effectively computed and added by tableau expansion rule applications. If the sort theory of a branch is not known to be decidable, we have shown that sorted unification can be restricted to rigid sorted unification, which is always decidable, because only the declarations on the current branch are considered without copies.

Tableaux extended with sorts have one additional nice property: For certain classes of propositional formulas they provide us with an efficient decision procedure for satisfiability. We have experimented with a method by which propositional logic formulae are translated into first-order logic with sorts; we apply our sorted version of tableau construction to the resulting clauses to decide satisfiablity. We have implemented this method in a prototype [8] (see 8.6.6) and obtained some promising experimental results.

References

[1] A. Cohn. A more expressive formulation of many sorted logic. *Journal of Automated Reasoning*, 3(2):113–200, 1987.

[2] A. Frisch. The substitutional framework for sorted deduction: fundamental results on hybrid reasoning. *Artificial Intelligence*, 49:161–198, 1991.

[3] M. Schmidt-Schauß. *Computational aspects of an order sorted logic with term declarations*, volume 395 of *LNAI*. Springer, 1989.

[4] P. Schmitt and W. Wernecke. Tableau calculus for order sorted logic. In K. Bläsius, U. Hedtstück, and C. Rollinger, editors, *Sorts and Types in Artificial Intelligence*, volume 418 of *LNAI*, pages 49–60. Springer, April 1989.

[5] C. Walther. *A Many-sorted Calculus based on Resolution and Paramodulation.* Research Notes in Artificial Intelligence. Pitman Ltd., 1987.

[6] C. Weidenbach. Unification in sort theories and its applications. Technical Report MPI-I-93-211, 1993.

• [7] C. Weidenbach. First-order tableaux with sorts. In K. Broda and M. D. et.al., editors, *TABLEAUX-'94, 3rd Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 247–261. Imperial College of Science Technology and Medicine, TR-94/5, April 1994.

• [8] C. Weidenbach. Sorts, resolution, tableaux and propositional logic. In J. Kunze and H. Stoyan, editors, *KI-94 Workshops, Extended Abstracts*, pages 315–316. GI, Gesellschaft für Informatik, 1994.

8.1.4 Constraint Solving

RPO Ordering Constraints

Investigator: Christoph Weidenbach

Ordering constraints are used to restrict the search space in saturation-based theorem proving. An ordering that is often used in practice is the recursive path ordering (RPO). It generalizes both the lexicographic (LPO) and the multiset path ordering by allowing each function symbol to have a lexicographic or multiset status. The RPO is a total reduction quasi-ordering on ground terms. Recently, Rubio and Nieuwenhuis [5] have presented an AC-compatible, total ordering based on the RPO.

Jouannaud and Okada [2] have proved that the satisfiability of RPO constraints is decidable. We have exhibited a new decision procedure [6] which is a generalization of Rubio's algorithm for LPO constraint solving [4]. Like this, and the algorithms presented in [1, 3], our procedure is based on computing successor terms and should therefore be more efficient in practice.

References

[1] H. Comon. Solving inequations in term algebras. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science, LICS'90*, pages 62–69. IEEE Computer Society Press, Los Alamitos, CA, USA, 1990.

[2] J. Jouannaud and M. Okada. Satisfiability of systems of ordinal notations with the subterm property is decidable. In *Proc. ICALP 91*, volume 510 of *LNCS*, pages 455–468, Madrid, 1991. Springer.

[3] R. Nieuwenhuis. Simple LPO constraint solving methods. *Information Processing Letters*, 47(2):65–69, August 1993.

[4] A. Rubio. *Automated Deduction with Constrained Clauses*. PhD thesis, Departament de Llenguatges i Sistemes Informàtics de la Universitat Politècnica de Catalunya, Barcelona, April 1994.

[5] A. Rubio and R. Nieuwenhuis. A precedence-based total AC-compatible ordering. In *Proc. 5th Int. Conf. on Rewriting Techniques and Applications*, volume 690 of *Lecture Notes in Computer Science*, pages 374–388, Berlin, 1993. Springer-Verlag.

• [6] C. Weidenbach. An algorithm for testing satisfiability of RPO constraints. Submitted, 1995.

Negative Set Constraints

Investigators: Witold Charatonik, Leszek Pacholski

The decidability of systems of mixed positive and negative set constrains has been proved independently by several research groups. Gilleron, Tison, and Tommasi [4] have given a proof based on the notion of a tree set automaton; the proof of Aiken, Kozen, and Wimmers [1] relies on a translation to a diophantine reachability problem. Using the translation of set constraints to monadic formulas that was presented in [2] Charatonik and Pacholski have obtained a third proof, that also allows restricted projections and diagonalizations [3].

References

[1] A. Aiken, D. Kozen, and E. Wimmers. Decidability of systems of set constraints with negative constraints. Technical Report 93-1362, Computer Science Department, Cornell University, June 1993.

[2] L. Bachmair, H. Ganzinger, and U. Waldmann. Set constraints are the monadic class. In *Proc. 8th IEEE Symposium on Logic in Computer Science*, pages 75–85. IEEE Computer Society Press, 1993.

• [3] W. Charatonik and L. Pacholski. Negative set constraints. In *Proc. 9th IEEE Symposium on Logic in Computer Science*, pages 128–136. IEEE Computer Society Press, 1994.

[4] R. Gilleron, S. Tison, and M. Tommasi. Solving systems of set constraints with negated subset relationships. In *34th Annual Symposium on Foundations of Computer Science*, pages 372–380, Palo Alto, CA, USA, Nov. 3–5, 1993. IEEE Computer Society Press, Los Alamitos, CA, USA.

8.1.5 Efficiency in Theorem Provers

We have studied both theoretical and practical aspects of this problem.

Search Efficiency of Theorem Proving Strategies

Investigator: David Plaisted

In [1] we analyze the search efficiency of a number of common refutational theorem proving strategies for first-order logic, including clause linking. Search efficiency is concerned with the total number of proofs and partial proofs generated, rather than with the sizes of the proofs. In general, in the field of automated deduction for full first-order logic, there has been a great deal of attention devoted to the completeness of strategies but little to their efficiency, in the sense of the total work expended in the search for a proof. The main efficiency considerations to date have to do with the times needed by particular implementations to find proofs of particular example theorems, or with the efficiencies of decision procedures for specialized theories. Theoretical measures of search space size would also make it easier to weed out bad strategies early and would stimulate the development of good ones. There is more at issue than just a quantitative measure of performance—analytical measures reveal something about how a strategy works, and how it does subgoaling. This gives some insight into the strategy. A theoretical approach could also help to pinpoint problem areas and weaknesses in a method and lead to improvements. In general, theory does not replace experiment but it does supplement it, and provides insights that might otherwise be missed. Theory tends to make general statements and to be machine-independent, whereas experiment tends to deal in specifics and to be machine-dependent.

We show that most common strategies produce search spaces of exponential size even on simple sets of clauses, or else are not sensitive to the goal. However, clause linking, which uses a reduction to propositional calculus, has behaviour that is more favorable in some respects, a property that it shares with methods that cache subgoals. A strategy which is of interest for term-rewriting based theorem proving is A-ordering, and we analyse it in some detail. We show some advantages of A-ordering over other strategies, which may help to explain its efficiency in practice. We also point out some of its combinatorial inefficiencies, especially in relation to goal-sensitivity and irrelevant clauses.

References

• [1] D. A. Plaisted. The search efficiency of theorem proving strategies: An analytical comparison. Technical Report MPI-I-94-233, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1994.

Term Indexing

Investigator: Peter Graf

The formulas which a theorem prover generates must be stored, and then accessed in various ways. Like in standard database technology, *indexing* is the key to efficiently retrieving data from large databases. The structure of logical data, however, is much more complicated than the structure of keys to data stored in a dictionary or in a database. As a consequence queries to a logical database are more complex than queries to a standard database. For instance, we might want to find all its matchers or all terms unifiable with a given term in the current database.

Among the known indexing methods for term retrieval in deduction systems, Path-Indexing [1] exhibits good performance in general. However, since Path-Indexing is not a perfect filter, the candidates it finds have to be checked with a unification algorithm to detect failures resulting from occur-checks or indirect clashes. We have developed the Extended Path-Indexing technique [2] which provides a perfect filter for the search of variants, instances, and more general terms and improves the filter properties for unifiable terms. We have also introduced a version of path indexing which is able to cope with associative-commutative function symbols.

Recently we have developed a new indexing technique which combines ideas from the discrimination tree and abstraction tree approaches. In so-called Substitution Tree Indexing [3] we represent terms in a flattened form, that is, as conjunctions of equations of the form $x_0 = f(x_1, \ldots, x_n)$, with x_i variables. The conjunctions are organized in a tree structure so as to exploit sharing (see figure 1). Substitution trees inherit from abstraction trees better sharing, avoiding inefficiencies related to chains of variable renaming. This data-structure has shown very promising performance in various experiments: in the average case the new indexing technique occupied less memory than other known techniques and response was better. As suggested by its name, with substitution trees we can also store idempotent substitutions and efficiently compute simultaneous unifiers $\sigma(x) = \tau(x)$ between two substitutions σ and τ, for all x in the domain of σ and τ. This last feature is exploited in a distributed hyperresolution prover which we are currently implementing. This prover takes uses Substitution Trees to compute simultaneous unifiers for literals efficiently in the electrons and literals of the nuclei.

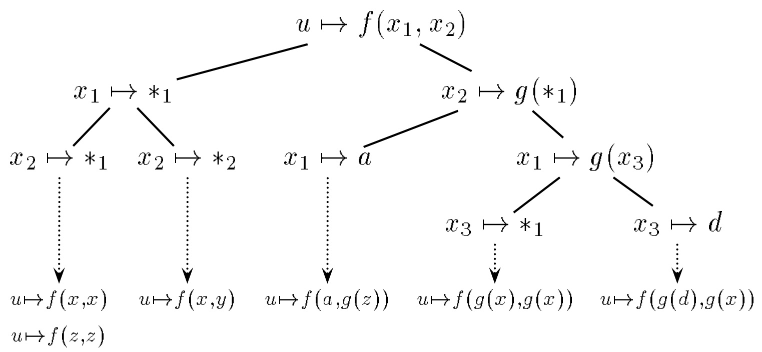

Figure 1: Substitution tree

References

[1] P. Graf. Path indexing for term retrieval. Technical Report MPI-I-92-237, Max-Planck-Institut für Informatik, Saarbrücken, 1992.

- [2] P. Graf. Extended path–indexing. In *Proceedings of CADE-12*, volume 814 of *LNAI*, pages 514–528. Springer, 1994. Full version available as MPI-I-93-253.

- [3] P. Graf. Substitution tree indexing. In *Proc. 6th Int. Conf. on Rewriting Techniques and Applications*. Springer, 1995. Full version available as MPI-I-94-251.

8.1.6 Clause Linking

Investigator: David Plaisted

We combine the semantic hyper-linking theorem proving method, which is a refinement of the clause linking method, with theorem proving methods based on orderings [1]. The idea of semantic hyper-linking is to show that a set S of clauses is unsatisfiable by the failure of a systematic search for a model of S. The search starts with a user-given interpretation, which provides some guidance for the search. Ordered semantic hyper-linking is similar, but it organizes the search differently. The basic principle behind ordered semantic hyper-linking is the following: Suppose we have n independent choices to make in a process of examining a set of possibilities. Thus there are potentially 2^n combinations of choices altogether. We assume that there is an ordering on these choices, and we make the simplest choice first. For each of the resulting alternatives, we recursively attempt to solve the problem. The reasoning is that we may solve the problem before the more complex choices are even seen, thereby saving effort. This seems to be a natural strategy from the standpoint of human problem solving. Applied to theorem proving, the set of choices is infinite, and the order in which they are made has other implications, but the idea is still the same.

Furthermore, a problem with semantic hyper-linking is that sometimes the enumeration of ground terms is necessary. We would like to have a method that is based on unification instead of on the enumeration of ground terms. The proposed method incorporates unification in a natural way, and for certain kinds of semantics we show that the enumeration phase can be done in polynomial time. There is an additional reason to believe that this new version will have better performance. The work required by semantic hyper-linking is strongly influenced by the number of eligible literals that are generated. The proposed method should reduce this number, thereby making the method more efficient and permitting proofs that require more rounds of search.

References

- [1] D. A. Plaisted. Ordered semantic hyper-linking. Technical Report MPI-I-94-235, 1994.

8.2 Non-Classical Logics

The investigation of non-classical logics can be seen in at least three ways. First, there is the 'algorithmic' view: classical propositional logic is NP-complete, first-order predicate logic (PL1) is undecidable and higher-order predicate logic is even incomplete. Inference procedures can therefore be extremely complex, so it is a good idea, for a given application, to try to choose the simplest logic available and appropriate. In order to refine the very coarse classification into three logical systems given above, and to have a finer grained spectrum available, various non-classical logics with expressiveness lying between them have been developed. For example many modal logics lie somewhere between propositional and first-order predicate logic: most of them are decidable, although the decision problem is usually more complex than for propositional logic (usually PSPACE complete). Nevertheless given an application where the expressiveness of such a logic is sufficient, using modal logic gives a better performance than encoding the problem in predicate logic. Second, there is the 'algebraic' view; many non-classical logics correspond to certain algebras, usually lattices (that Boolean lattices or Boolean Algebras correspond to propositional logic is the best known example). An investigation of these logics therefore goes along with an investigation of the corresponding algebras and methods and results can be easily exchanged between the two. And third, there is what we might call the 'applicational' view; there are a number of phenomena which are basically of a logical nature, but which are difficult or even impossible to encode in predicate logic. Examples are nonmonotonicity, in particular default reasoning, or probabilistic reasoning.

We have investigated various aspects of non-classical logics with respect to these three. Our ultimate goal is to provide a large spectrum of logics and inference systems from which the optimal logic for a given application can be chosen, and which is supported with powerful tools. To this end we have developed a unified framework for presenting a wide range of logics and their associated algorithms, and using them cooperatively. In particular we have been trying to integrate these logics into predicate logic in such a way that we can make use of predicate logic inference systems while keeping the characteristics and advantages of the original systems. Finally, we are trying to exchange results and methods from related areas, e.g. lattice theory.

Specifically, our main areas of research are

- the development of a meta-theory for 'engineering' logics, in particular for computing representations and model theoretic semantics for axiomatically presented systems,
- the investigation of logics and algebras lying between propositional and first-order predicate logic,
- the investigation of probabilistic and default reasoning and reason maintenance systems,
- a case study on the combined application of various different logics and inference systems.

We are collaborating with Prof. Dov Gabbay (Imperial College, London). Part of the work has been done in the Esprit project MEDLAR and in the BMFT funded project Logo. A new project (TraLoS, Transformation of Logical Systems) has just been granted from the DFG.

8.2.1 Representation Theorems and Model Theoretic Semantics

Investigator: Hans Jürgen Ohlbach

The problem is as follows: Given an axiomatization Φ of a class of algebras, find a representation $\Upsilon(\Phi)$ for these algebras. Representations we are interested in consist of a mapping of the elements of the algebra onto structures, together with a mapping of the functions of the algebra to operations on the structure. The representation should be presented as a transformation Υ of logical formula such that for each formula φ it is guaranteed that $\Phi \Rightarrow \varphi$ iff $\Upsilon(\Phi) \Rightarrow \Upsilon(\varphi)$. The axiomatizations Φ can be equational axiomatizations of, say, a Boolean algebra with extra functions or an axiomatization of a logic in terms of a binary consequence relation (similar to the Frege style axiomatizations). Typical representations are for instance, that provided by Stone's representation theorem for Boolean Algebras, or Jónsson and Tarski's representation for Boolean Algebras with operators. In logical terms, representations provide classes of models with which a given logic is complete, e.g. classical characteristic frames for modal logics. This kind of semantics is the basis for many inference procedures.

We have shown that given an axiomatization Φ of an extension of a distributive lattice with extra function symbols, and given a characterization of a representation in a certain way, we can translate Φ into formulae of second-order predicate logic (PL2) describing properties of the representation [6]. By means of a quantifier elimination method this formula can in many cases be reduced to a first-order predicate logic (PL1) formula giving a direct axiomatic specification of the properties of the representation.

The representation can for example be a PL1 formula

$$\mathit{satisfiable}(w, f(x)) \Leftrightarrow \forall v\ R(w, v) \Rightarrow \mathit{satisfiable}(v, x)$$

which in this case is a PL1 formulation of the semantics of the modal $\Box$–operator for normal modal logics in terms of the accessibility relation R (here we use PL1 as the target logic into which we compile modal logic). In this case the result of the two step transformation first into PL2 and then PL1 is the axiomatic description of the frame classes of the given modal system, (c.f. the 'correspondence problem' for modal logics).

We give simple criteria for checking whether a particular representation is admissible and have found algorithms for finding the representation automatically, at least in some cases [9, 5].

Our result generalizes on the theorem proving aspects of well known results, in particular of Jónsson and Tarski's representation theorem for Boolean Algebras with operators. The most important aspect of the generalization is that we can compute the representation for instances of these algebras specified with extra axioms describing properties of the operators.

The precondition that Φ axiomatizes at least a distributive lattice covers a large class of extensions of known logics; e.g.if Φ axiomatises a Boolean (Heyting) algebra with extra functions, then we get a the algebraic equivalent of a classical (intuitionistic) logic with modal operators.

We have already investigated the inverse operation of finding, for a given representation (i.e. a semantics in the case of a logic) the corresponding axioms [2].

References

- [1] F. Baader and H. J. Ohlbach. A multi-dimensional terminological knowledge representation language. Technical Report MPI-I-95-2-005, January 1995.

- [2] C. Brink, D. M. Gabbay, and H. J. Ohlbach. Towards automating duality. *Journal of Computers and Mathematics with Applications*, 29(2):73–90, 1994. Special Issue on Automated Reasoning. Also available as Research Report MPI-I-93-220.

- [3] D. Gabbay and H. J. Ohlbach, editors. *Temporal Logic: Proceedings of the First International Conference on Temporal Logic*, volume 827 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer Verlag, 1994.

- [4] H. J. Ohlbach. Computer support for the development and investigation of logics. Technical Report MPI-I-94-228, 1994.

- [5] H. J. Ohlbach. Synthesizing semantics for extensions of propositional logic. Technical Report MPI-I-94-225, June 1994.

- [6] H. J. Ohlbach. Boolean algebras with functions – correspondence, completeness and quantifier elimination. Submitted to IJCAI 95, 1995.

- [7] H. J. Ohlbach. Clause killer transformations. Submitted to the Journal of Automated Reasoning, 1995.

- [8] H. J. Ohlbach. General representation theorems. Technical Report MPI-I-95-2-006, Max-Plack-Institut für Informatik, Saarbrücken, 1995. to appear.

- [9] H. J. Ohlbach, D. M. Gabbay, and D. A. Plaisted. Killer transformations. Technical Report MPI-I-94-226, 1994.
- [10] H. J. Ohlbach and C. Weidenbach. A note on assumptions about skolem functions. *Journal of Automated Reasoning*, 1995. Forthcoming.
- [11] H. J. Ohlbach (editor). Temporal logic: Proceedings of the ICTL workshop. Technical Report MPI-I-94-230, 1994.

8.2.2 Quantifier Elimination

Investigators: Thorsten Engel, Andreas Nonnengart, Hans Jürgen Ohlbach

The computationally complicated part of the generation of representations is the quantifier elimination algorithm that turns PL2 formulae into equivalent PL1 formulae, if possible. This algorithm (SCAN) [1] has been implemented and provided with various interfaces. Besides the direct interface for treating second-order formulae, there are interfaces for calculating the representation from the axioms of the distributive lattice with functions, and first-order circumscription in the McCarthy style. These are accessible via world wide web (WWW) and the program can be used remotely by just filling out some HTML forms. The WWW address is

`http://www.mpi-sb.mpg.de/guide/staff/ohlbach/scan/scan.html`

Quantifier elimination is incomplete, i.e. there is no algorithm that always finds a solution when it exists. In the SCAN algorithm this effect manifests itself in nonterminating resolution loops. Sometimes, however, these resolution loops are unavoidable because the answer consists of an infinite conjunction of disjunctions. We have therefore developed an approach (see [2]) which allows to handle also cases in which the corresponding formula without the predicate quantifier can only be described by infinite disjunctions or conjunctions represented by suitable fixpoint operators. For example, when applied to the induction axiom for Peano arithmetics it results in the well-known, though non-trivial fact that each number can be obtained from 0 by a finite number of applications of the successor relation.

When applied to modal logics the approach turns out to be successful in cases where the SCAN algorithm does not terminate. As an interesting modal logic example we have examined, for instance, the system G, characterized by the so-called Löb axiom

$$\Box(\Box\Phi \Rightarrow \Phi) \Rightarrow \Box\Phi\,.$$

After application of our second-order quantification elimination algorithm we end up with the fixpoint formula

$$\forall x, y(R(x,y) \Rightarrow \mu P(y).(R(x,y) \wedge \forall z(R(y,z) \Rightarrow P(z))))$$

which states that R is transitive and backward well-founded, properties which are difficult to find by model theoretic examination.

References

[1] D. M. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning (KR92)*, pages 425–435. Morgan Kaufmann, 1992.

- [2] A. Nonnengart and A. Szałas. On fixpoint characterizations of modal logics. Forthcoming MPI report, 1995.

8.2.3 Special Representations

Investigator: Renate Schmidt

There are three levels characterizing a concrete representation for a given axiomatization Φ of a logic. The first level is determined by the basic lattice structure, whether you have, say a distributive lattice, a Heyting algebra (intuitionistic logic) or a Boolean algebra or more. This determines the isomorphism between the elements of the lattice and the elements of the representation (usually sets). The second level concerns the extra functions (for example the modal operators). Certain characteristic properties of these functions determine a mapping to certain relations on the representation. The third level correlates properties of the functions, i.e. special axioms in Φ with properties of the relations. The first and second level are not uniquely determined, but usually there is a criterion for assessing whether one representation is better than another (more axioms in Φ become tautologies in $\Upsilon(\Phi)$). Since we want to automate as much as possible the process of finding more suitable representations, we investigated more complex representations in the area of terminological logics (which are closely related to modal logics).

Terminological representation languages are fragments of first-order languages in which predicates have at most two arguments. They may be regarded as algebraic languages describing sets and relations interacting with each other [3, 2, 1, 4, 6]. Concepts (unary predicates) are interpreted as sets and roles (binary predicates) are interpreted as binary relations. Sets give rise to Boolean algebras and relations give rise to relation algebras (due to Tarski). Sets can be combined with relations, and these interactions have an algebraic formalization, for example, in Peirce algebras [1].

In the field of modal logic Peirce algebras and its subreducts provide an algebraic semantics for dynamic logic, arrow logic and dynamic modal logic. Of particular interest (not only to multi-modal logic but also to knowledge representation and computational linguistics) is the class of concrete Peirce algebras and, in particular, the class of full Peirce algebras. The first characterization of full Peirce algebras appears in the PhD Thesis of de Rijke. This is also the first published representation theorem for Peirce algebras. In his characterization de Rijke uses two conditions (besides simplicity), one on the Boolean set algebra and another on the relation algebra. These are the algebraic analogues of two irreflexivity laws required for the completeness proof of the logical analogue of Peirce algebras, dynamic modal logic. We show [5] that in Peirce algebras, because the Boolean set algebra is determined by the relation algebra, one condition, on the relation algebra, is sufficient for the representation theorem. In contrast to the proof of de Rijke which is obtained from the completeness proof of dynamic modal logic, our proof is algebraic. The class of full Peirce algebras is characterized by the class of complete and atomic Peirce algebras in which the set of relational atoms is restricted by two conditions. One requires simplicity. The other requires that each relational element can be uniquely expressed in terms of Boolean atoms, or equivalently, that each relational element can be uniquely expressed in terms of relational identity atoms or relational points. This representation result for Peirce algebras parallels the representation theorems of Jónsson and Tarski, McKinsey, G. Schmidt and Ströhlein for full relation algebras.

References

• [1] C. Brink, K. Britz, and R. A. Schmidt. Peirce algebras. *Formal Aspects of Computing*, 6:1–20, 1994.

[2] C. Brink and R. A. Schmidt. Subsumption computed algebraically. *Computers and Mathematics with Applications*, 23(2-5):329–342, 1992. Also available as Technical Report TR-ARP-3/90, Automated

Reasoning Project, Research School of Social Sciences, Australian National University, Canberra, Australia.

[3] R. A. Schmidt. Algebraic terminological representation. Technical Report MPI-I-91-216, Max-Planck-Institut für Informatik, Saarbrücken, 1991. Also available as Thesis-Reprints TR 011, Department of Mathematics, University of Cape Town, South Africa.

• [4] R. A. Schmidt. Peirce algebras and their applications in artificial intelligence and computational linguistics: Abstract. *SIGALA Newsletter*, 2(1):27, 1994.

• [5] R. A. Schmidt. Representations as full Peirce algebras: Extended abstract. Submitted to AMAST'95, November 1994.

• [6] R. A. Schmidt. Terminological logics and conceptual graphs: An historical perspective. In J. Kunze and H. Stoyan, editors, *KI-94 Workshops: Extended Abstracts*. Gesellschaft für Informatik, Bonn, 1994.

8.2.4 Translation from Modal into Predicate Logic

Investigators: Hans Jürgen Ohlbach, Renate Schmidt

A representation, or a model theoretic semantics, provides a means for translating formulae from the object logic into predicate logic automatically. Therefore one can apply predicate logic inference systems, for example resolution, to these translated formulae. The translation into first-order predicate logic is limited to logics with first-order representations.

Previously we developed the so called functional translation from modal logic into predicate logic. Its advantage is that the term structure in the translated formulae corresponds to the original formula structure. Therefore it is possible to encode the characteristics of the particular modal logic into theory unification algorithms, thus combining the possibilities of resolution systems with special algorithms for the given logic. We can now show how, using this functional translation, one can map certain second-order frame properties into first-order axioms and therefore extend the applicability of resolution systems to these logics [1]. In a forthcoming paper [2] we apply this idea to modal logic with graded modalities. This logic encodes finite sets. This way we can get a much more efficient way of reasoning with sets of finite, but arbitrary large cardinality than was previously possible.

References

• [1] H. J. Ohlbach and R. A. Schmidt. Functional translation and second-order frame properties of modal logics. Technical Report MPI-I-95-2-002, January 1995.

• [2] H. J. Ohlbach, R. A. Schmidt, and U. Hustadt. Translating graded modalities into predicate logic. To appear in *Knowledge and Belief in Philosophy and Artificial Intelligence*, Akademie-Verlag, Berlin, 1995, January 1995.

8.2.5 Saturation of Modal Logic Background Theories

Investigator: Andreas Nonnengart

The functional translation of modal logic formulae requires equational reasoning or theory unification algorithms. Quite often, it is not possible to integrate this into an existing inference system. Prolog is such an example. To overcome this problem, we have developed a compromise, the semi-functional translation, which has almost all the advantages of the functional translation, but does not require equational reasoning.

As a side effect of our translation approach we get that the formulae obtained as the translation result can be very easily distinguished syntactically from the formulae which characterize the modal logic under consideration (its background theory). Because the translated formulae and the background theory are strictly separated it is possible to perform all possible reasoning steps within the background theory, independently of the theorem to be proved. We have called this procedure saturation of the background theory for a modal logic, and it results in formulae which are characteristic for a given modal logic in the sense that it has to be performed for a logic only once, and is then available to be used for any theorem to be checked. The main effect of this saturation (together with some further useful modal logic properties like Segerberg's connectedness assumption) appears when it is possible to reduce a complex background theory to a few unit clauses. For instance, the background theory for the modal logic KD45 which is often used to model consistent belief together with full introspection reduces to a single unit clause.

We have examined many modal logics this way and it turns out that the saturation of background theories has a similar effect in most cases. Not all reduce to single unit clauses but nevertheless the procedure has proved to be significant for every logic [1, 2]. (Interestingly, our saturation theorem proving system (see section 8.6.7) is sometimes able to calculate the saturation from the semi-functional translation automatically.)

Another interesting aspect of this approach is that it allows us to use modal logic in programming (at least for those modal logics with a background theory reducible to one or more unit clauses) without any changes on the logic programming environment. In fact the method is not restricted to modal logic applications; it can also be used to extend logic programming with sorts [3]. The simplification ideas are the same, though the effect is usually not as considerable as for modal logics. Nevertheless, it allows us to use sorts in logic programming languages to some extent, something hardly possible before.

References

[1] A. Nonnengart. First-order modal logic theorem proving and standard PROLOG. Technical Report MPI-I-92-228, Max-Planck-Institut für Informatik, Saarbrücken, 1992.

[2] A. Nonnengart. First-order modal logic theorem proving and functional simulation. In R. Bajcsy, editor, *Proc. 13th Intern. Joint Conference on Artificial Intelligence (IJCAI '93)* (Chambery, France), volume 1, pages 80–85, San Mateo, CA, 1993. Morgan Kaufmann.

• [3] A. Nonnengart. How to Use Modalities and Sorts in Prolog. In *Proceedings of the JELIA'94, LNAI 838, Logics in Artificial Intelligence*. Springer Verlag, 1994.

8.2.6 Logic and Uncertainty

Investigators: Manfred Jaeger, Emil Weydert

The handling of uncertain, and thus defeasible, knowledge is a major research problem in AI. Logical approaches to reasoning about uncertainty try to combine the best of both worlds: the representational power of logic and the practical relevance of probabilistic techniques. An important subarea of this is default reasoning. In our group, we have been concerned with first-order logics of probability and qualitative probabilistic approaches to defaults and defeasible inference.

In the first research area, we have considered extensions of first-order logic that incorporate representations of probabilistic information. Following work by Halpern [2] and Bacchus [1], the syntax of first-order logic is extended with two syntactic operators allowing us to represent statements about two types of probabilities: statistical probabilities indicating the relative frequency with which objects of the domain of discourse satisfy certain predicates, and subjective probabilities expressing a degree of belief of an individual reasoner that a certain proposition is true.

In the work by Halpern and Bacchus, a semantics is defined for the extended language that interprets statistical and subjective probability expressions respectively by probability measures on the domain and on a set of possible worlds. This strong conceptual separation between statistical and subjective probabilities makes it inherently difficult to model the dependency of subjective beliefs on the available statistical information. We primarily address the analysis and formalization of this dependency. In order to integrate it into the semantics of the extended language, both statistical probability expressions and subjective probability expressions are interpreted by probability measures on the domain. That way it becomes possible to effectively compare the subjective probability measure with the statistical probability measure, and to require in the semantics that the subjective measure must be as close as possible to the statistical measure given the partial information about both measures. Jaeger has proposed using cross-entropy as a measure for the dissimilarity of two probability measures [5]. We show in [4, 3] that the use of this function can be justified both in terms of desired logical behaviour, and also on statistical grounds.

A further research task has been the development and investigation of new measures of belief. Traditionally, this has been achieved through (subjective) probability valuations. However, when precise numbers are lacking, irrelevant, computationally expensive or even meaningless, we may need a different concept. Furthermore, when we try to model plain, i.e. logically closed belief, the standard probabilistic account turns out to be inappropriate.

In [8], we have investigated what would constitute a minimal 'quasi-probabilistic' framework. By enforcing additional conditions, we obtain two notions of generalized measures which are of particular interest:

- Ranking measures, which take values in ordered commutative semi-groups and satisfy $R(A \sqcup B) = max\{R(A), R(B)\}$. They subsume earlier semi-qualitative measure concepts and offer a semantics for default conditionals.

- Cumulative measures, which combine the probabilistic and the ranking philosophy, and provide a more fine-grained model of plain belief.

Another possibility is to consider finitely additive measures taking values in nonstandard models of the reals, i.e. admitting infinitesimally small numbers, from which canonical ranking and cumulative measures can be derived. New results from model-theoretic algebra have paved the way to extended probability logics, whose valuation algebras are exponential real-closed fields extending the reals but carrying an explicit standard part. This framework is expressive enough for handling all kinds of default knowledge or inference mechanisms based on cross-entropy [7].

Last but not least, we have explored a new defeasible inference relation, which combines techniques from belief revision, e.g. Jeffrey conditionalization, with strategies from default reasoning, e.g. normality maximization. By making explicit the abnormality parts of defaults, we can get both, syntax-independency and inheritance through exceptional subclasses [6].

References

[1] F. Bacchus. *Representing and Reasoning With Probabilistic Knowledge.* MIT Press, 1990.

[2] J. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.

• [3] M. Jaeger. Minimum cross-entropy reasoning: A statistical justification. Submitted to IJCAI-95.

• [4] M. Jaeger. A logic for default reasoning about probabilities. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1994.

• [5] M. Jaeger. Probabilistic reasoning in terminological logics. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Principles of Knowledge Representation an Reasoning: Proceedings of the Fourth International Conference (KR94)*. Morgan Kaufmann, San Mateo, CA, 1994.

• [6] . Weydert. Default entailment. Submitted.

• [7] . Weydert. Numeric defaults. Submitted.

• [8] E. Weydert. General belief measures. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1994.

8.2.7 Reason Maintenance

Investigator: Detlef Fehrer

Reason maintenance is a major problem for inference systems operating in a dynamically changing environment. Various systems have been developed which keep track of reasoning chains and locate the parts of a database that need to be revised if new information is added, or old information is removed. Based on Gabbay's proposed *Labelled Deductive Systems* [3] we have developed a uniform method for describing all these systems. Our approach works for justification based as well as for assumption based methods, thus giving a *unifying* semantics to both of them. Unlike other systems, our approach is not restricted to propositional Horn clauses, but can treat arbitrary logics, e.g. full first-order logic. This enables us to characterize systems as a whole, including both the reason maintenance component and the problem solver, while nevertheless maintaining a separation between the basic logic and the part that describes the label propagation [1, 2].

References

[1] D. Fehrer. A unifying framework for reason maintenance. In M. Clarke, R. Kruse, and S. Moral, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty: Proc. Europ. Conference ECSQARU '93* (Granada, Spain), volume 747 of *Lecture Notes in Computer Science*, pages 113–120, Berlin, 1993. Springer-Verlag.

• [2] D. Fehrer. *A Unifying Framework for Reason Maintenance*. PhD thesis, Max-Planck-Institut, Saarbrücken, 1995. Forthcoming.

• [3] D. Gabbay. Labelled Deductive Systems. Technical Report MPI–I–94-223.

8.2.8 Case Study: Knowledge Representation using Non-Classical Logics

Investigator: Ullrich Hustadt

In co-operation with a group lead by Prof. Wahlster in Saarbrücken we have developed the knowledge representation system MOTEL as part of a natural language dialogue system [1]. Specifically, we have been developing an extension of terminological logics which is expressive enough to describe agent models which can maintain and exploit an explicit model of the dialogue partners' beliefs, goals, and plans and be used for planning the appropriate dialogue contributions for achieving goals, and determining the effects of dialogue contributions on a dialogue partner. We have sketched these extensions in the previous report. During 1994 we have concentrated on the problem of 'stereotypes'; i.e. how an subject should initially treat an interlocutor. There are two possibilities:

- We ascribe all or a subset of the system's knowledge, beliefs, goals, and plans to the dialogue partner, i.e. the initial agent model mirrors the system.

- We use predefined collections of knowledge, beliefs, goals and plans. At the beginning of the dialogue, the system chooses one of these collections and ascribes it to the dialogue partner.

The first approach is appropriate if the main use of the agent model is to assure that the dialogue partner understands all the utterances of the system as in the case of the UMFE system. But for example in an argument in which the personal attitude towards the topic of the discussion is important, this is evidently not a good approach. If the system wants to convince the dialogue partner that his attitude towards a topic is right, it should not start with the assumption that the dialogue partner already has the same attitude towards this topic.

Using predefined assumptions is usually called the stereotype approach to agent model ascription. In the literature, the term stereotype is used mostly for a collection of knowledge, beliefs, and goals that are typical for members of a group. That is, the properties contained in the stereotype can be likely ascribed to members of the group. This is opposed to the view that defines stereotypes as collecting the knowledge, beliefs and goals common to all members of a group. In both cases we have to find a way to select from a collection of stereotypes the one which we want to ascribe to a dialogue partner.

We have found that a small extension of our logic is already expressive enough to describe these stereotypes [2, 4]. However, we have also been looking at more expressive query languages for our logic. Since the semantics of our language obeys the open-world and open-domain assumptions, queries are answered according to these assumptions too. Hustadt [3] shows the usefulness of query answering in natural language processing systems which is based on closed-world and closed-domain assumptions. An integration into our framework is an open problem.

References

- [1] D. Fehrer, U. Hustadt, M. Jaeger, A. Nonnengart, H. J. Ohlbach, R. A. Schmidt, C. Weidenbach, and E. Weydert. Description logics for natural language processing. In *International Workshop on Description Logics '94*, pages 80–84, Bonn, Germany, 1994. DFKI.

- [2] U. Hustadt. Common and mutual belief for agent modeling. To appear in *Knowledge and Belief in Philosophy and Artificial Intelligence*, Akademie-Verlag, Berlin, 1995, 1994.

- [3] U. Hustadt. Do we need the closed-world assumption in knowledge representation. In F. Baader and M. A. Jeusfeld, editors, *Reasoning about Structured Objects: Knowledge Representation meets Databases. Workshop during the 18th German Annual Conference on Artificial Intelligence (KI-94)*, 1994.

- [4] U. Hustadt. A multi-modal logic for user modeling. In *Proceedings of the Fourth International Conference on User Modeling UM94* (Hyannis, MA), pages 87–92, Bedford, MA, 1994. The MITRE Corporation.

8.2.9 Summary

Our general results about representations of distributive lattices with functions provide an excellent means for defining all kinds of mixed logics, for getting a model theoretic semantics and for turning the model theoretic semantics into translation functions from the source logic into PL1. More sophisticated translations such as the semi-functional or the functional translation allow us to exploit in the general resolution framework the special knowledge about the source logic. One such resolution based theorem prover for translated formulae is the inference machine underlying the MOTEL knowledge representation system.

The quantifier elimination algorithm we have already developed seems to be useful also in other areas. Our first customer of the WWW accessible variant, from Austin, Texas, uses it for computing circumscription in action planning applications.

In the area of probabilistic and nonmonotonic reasoning a number of promising new concepts have been developed.

8.3 Logic and functional programming

Logic programming is based on the idea of using logic on the object level as a programming language, in contrast to using it on the meta-level to reason about programs. Since efficiency is one of the most important requirements of any programming language, the logics used in logic programming are relatively simple. Classical logic programming is based on non-left recursive Horn logic, which, among others, has the nice property that there is a much more efficient resolution strategy, SLD-resolution, than in full first-order logic. The general task, which is characteristic for most research in logic programming, is to find the right balance between expressive power on the one hand and efficient operational behaviour on the other.

A fundamental problem in classical logic programming is that it does not support functions and equality. A user cannot define functions by recursive equations like in other programming languages, but has to express them as relations, which causes many inefficiencies. In addition, even standard domains of computation, like arithmetic or Boolean algebra, are not well supported.

The integration of logic and functional programming has been a major research theme in our group. On the theoretical side, we have studied narrowing calculi, which provide the operational semantics of logic and functional programming languages. On the practical side, we have developed techniques for the analysis and optimization of declarative programs, which are based on abstract interpretation. Very recently, we have presented a new approach to using linear logic for program analysis in functional programming. In the area of constraint logic programming we have been working on the integration of efficient constraint solving techniques for specific domains of computations, in our case pseudo-Boolean constraints, into logic programming languages.

8.3.1 Integration of Functional and Logic Languages

Investigators: Alexander Bockmayr, Michael Hanus

Functional and logic programming are the most important declarative programming paradigms, and interest in the amalgamation of the two has been growing since the beginning of the last decade.

An integrated language has advantages from both the functional and the logic programming points of view: against pure functional programming it offers features like like function inversion, partial data structures and logical variables, while against pure logic languages, it is able to take advantage of the deterministic nature of functions to provide greater efficiency. Hence functions integrated into logic programming allows some of the impure control features of Prolog, like the cut operator to be avoided. These considerations were the motivation for integrating the two language types. Early research in this area concentrated on the definition and improvement of appropriate execution principles for functional logic languages, while in recent years efficient implementations of these execution principles have been developed so that these languages became relevant for practical applications. In [6] we give a survey on the theoretical and practical developments in this area.

A common approach to the integration of functions into logic programs is to have functions defined using equations inside the logic program, then the operational semantics is usually based on *narrowing*, which is a universal unification procedure for equational theories defined by confluent term rewrite systems. Since naïve narrowing is extremely inefficient, many refined narrowing strategies have been proposed. In [1], we present an optimal narrowing strategy for inductively sequential rewrite systems. This is based on the extension of the Huet and Lévy notion of a needed reduction step to narrowing. Our strategy is sound and complete for a large class of rewrite systems, optimal with respect to the cost measure that counts the number of distinct steps of a derivation, computes only independent unifiers, and is efficiently implemented by pattern matching. The basic requirement to obtain this optimal strategy is that the left-hand sides of the rewrite rules do not overlap. In the case of overlapping left-hand sides, we show in [7] that the inclusion of a simplification process between narrowing steps is useful even for a lazy narrowing strategy. This simplification process reduces the search space so that in some cases infinite search spaces are reduced to finite ones. In [5] we extend these results to nonterminating rewrite systems. Here it is necessary to perform the simplification process in a lazy manner in order to avoid infinite simplification derivations.

In [3], we have introduced the LSE narrowing strategy, which is complete for arbitrary canonical rewriting systems and optimal in the sense that two different LSE narrowing derivations cannot generate the same answer substitution. Moreover, all narrowing substitutions computed by LSE narrowing are normalized. An initial impression that one might get from the definition of LSE narrowing is that it is very expensive, because a large number of terms must be checked for reducibility. In [9] we show that many of these terms are identical. We describe how, using left-to-right basic occurrences, the number of terms that have to be tested can be reduced drastically. Using these results, we develop an efficient implementation of LSE narrowing.

In [4], we extend LSE narrowing to the case of confluent and decreasing conditional term rewrite systems. Using the calculus of conditional rewriting without evaluation of the premise [2], we are able to generalize most of our earlier results. This calculus, which establishes the connection between ordinary conditional rewriting and conditional narrowing, shows that some of the results can hold only for successful derivations. In practice, however, this is not a restriction. The methods developed in [9] for an efficient realization of LSE narrowing can also be used in the conditional case.

In [8], we discuss the influence of extra variables on the completeness of various narrowing strategies for conditional rewrite systems. It is well known that narrowing may become incomplete in the presence of extra variables. We show that this is not the case in weakly orthogonal normal conditional rewrite systems. We prove that each narrowing strategy which is complete for such rewrite systems without extra variables can be transformed into a strategy which is complete in the presence of extra variables. Using this technique, we can derive a number of new completeness results; in particular, our method does not require terminating rewrite systems and also permits extra variables in right-hand sides.

References

- [1] S. Antoy, R. Echahed, and M. Hanus. A needed narrowing strategy. In *Proc. of the 21st ACM Symposium on Principles of Programming Languages (POPL'94* (Portland)*)*, pages 268–279, New York, 1994. ACM Press. Extended version available as Technical Report MPI-I-93-243.

[2] A. Bockmayr. Conditional narrowing modulo a set of equations. *Applicable Algebra in Engineering, Communication and Computing*, 4(3):147–168, 1993.

- [3] A. Bockmayr, S. Krischer, and A. Werner. Narrowing strategies for arbitrary canonical systems. *Fundamenta Informaticae*, 1994. To appear.

- [4] A. Bockmayr and A. Werner. LSE narrowing for decreasing conditional term rewrite systems. In *Conditional Term Rewriting Systems CTRS'94, Jerusalem*, 1994.

- [5] M. Hanus. Combining lazy narrowing and simplification. In M. Hermenegildo and J. Penjam, editors, *Proc. 6th International Symposium on Programming Language Implementation and Logic Programming (Madrid, Spain)*, volume 844 of *Lecture Notes in Computer Science*, pages 370–384, Berlin, 1994. Springer-Verlag.

- [6] M. Hanus. The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, 19&20:583–628, 1994.

- [7] M. Hanus. Lazy unification with simplification. In D. Sannella, editor, *Proc. 5th European Symposium on Programming*, volume 788 of *Lecture Notes in Computer Science*, pages 272–286, Berlin, 1994. Springer-Verlag.

- [8] M. Hanus. On extra variables in (equational) logic programming. Technical Report MPI-I-94-246, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1994. to appear in Proc. of the International Conference on Logic Programming, Tokyo, 1995.

- [9] A. Werner, A. Bockmayr, and S. Krischer. How to realize LSE narrowing. In *Algebraic and Logic Programming, ALP'94, Madrid*, pages 59 – 76. Springer, LNCS 850, 1994.

8.3.2 Analysis and Optimization of Declarative Programs

Investigators: Michael Hanus, Frank Zartmann

The functional and logic programming paradigms have been shown to be useful for solving a wide range of problems by allowing us to state these problems declaratively at a high level of abstraction. Because of this high level of programming, efficient implementations of such languages require sophisticated compilation techniques. Although various new compilation techniques based on abstract machines have been developed, high-performance implementations require knowledge about the possible use of the high-level constructs of the language. Therefore, global analysis at compile time and abstract interpretation are widely believed to play an essential rôle in the development efficient implementations of these languages, that are able to bridge the large gap between the language and the machine. Abstract interpretation was developed as a formal analysis technique for imperative languages by the Cousots, and has subsequently been adopted in the context of functional and logic programming. The basic idea of abstract interpretation is to approximate the concrete run-time behaviour of the program by executing the program with abstract values. The execution is guaranteed to terminate as long as there are only finitely many abstract values. If the abstract values have a properly defined relationship to the concrete values, the result of the abstract interpretation correctly approximates the concrete run-time behavior of the program and the results can be used to compile the programs into a specialized and hence more efficient code.

We have used abstract interpretation techniques to analyze and optimize functional logic programs. The usefulness of the availability of run-time information for the efficient implementation of functional logic programs is shown in [1]. In this paper we propose code optimizations for the implementation of normalizing narrowing which depend on run-time information. If it is known at compile time that some arguments of functions are definitely free at run-time, we can infer that particular normalization steps can never be performed and thus we do not need to generate code for them. If some arguments are definitely ground at run time, these function calls are completely evaluable by normalization and again we avoid having to generate code for narrowing. The influence of these optimizations to the run time and space of example programs is also discussed in this paper.

In order to integrate these code optimizations into existing compilers, we need techniques for automatically approximating run-time properties at compile time. For this purpose we present in [4] an abstract interpretation framework for the analysis of functional logic programs. The concrete operational semantics considered in this paper is normalizing innermost narrowing, which combines the deterministic reduction principle of functional languages with the nondeterministic search principle of logic languages. Due to the normalization process between narrowing steps, standard analysis frameworks for logic programming cannot be applied. Therefore, we develop new techniques for correctly approximating the effect of the intermediate normalization process.

Residuation is an alternative method for executing functional logic programs which tries to avoid nondeterministic computation steps when evaluating functions. The residuation principle delays the evaluation of functions during the unification process until the arguments are sufficiently instantiated. This has the advantage that the deterministic nature of functions is preserved, but may be incomplete: if the variables in a delayed function call are not instantiated by the logic program, this function is never evaluated and some answers may be lost. In [2] we improve our previous work on the analysis of programs based on the residuation principle. The abstract interpretation algorithm approximates the possible residuations and instantiation states of variables during program execution. If the algorithm computes an empty residuation set for a goal, then the concrete execution of the goal does not end with a nonempty set of residuations, i.e. the residuation is complete in the sense that it computes only fully evaluated answers.

In 1993 we applied similar ideas to analyze logic programs with nonlinear arithmetic constraints. In the constraint logic programming language CLP($\mathcal{R}$) only linear constraints are solved during the execution of such programs, while nonlinear constraints are delayed until they become linear. This method has the disadvantage that sometimes computed answers are unsatisfiable, or infinite loops occur because of the unsatisfiability of delayed nonlinear constraints. We characterized a class of CLP($\mathcal{R}$) programs for which all nonlinear constraints become linear at run time. In [3] we present a revised and detailed description of our program analysis technique.

References

- [1] M. Hanus. Towards the global optimization of functional logic programs. In P. Fritzson, editor, *Proc. 5th International Conference on Compiler Construction (Edinburgh)*, volume 786 of *Lecture Notes in Computer Science*, pages 68–82, Berlin, 1994. Springer-Verlag.
- [2] M. Hanus. Analysis of residuating logic programs. *Journal of Logic Programming*, 1995. (to appear).
- [3] M. Hanus. Compile-time analysis of nonlinear constraints in CLP($\mathcal{R}$). *New Generation Computing*, 1995. (to appear).

• [4] M. Hanus and F. Zartmann. Mode analysis of functional logic programs. In B. L. Charlier, editor, *Proceedings of the First International Static Analysis Symposium (Namur, Belgium)*, volume 864 of *Lecture Notes in Computer Science*, pages 26–42, Berlin, 1994. Springer-Verlag.

8.3.3 Linear Logic Based Program Analysis

Investigator: Andreas Tönne

Linear logic distinguishes linear (use-once) and non-linear (use-repeatedly) formula. This can be used in a typed λ-calculus to reason about variable usage. Our work presents a uniform approach to applying linear logic to traditional functional programming to obtain optimization information.

We have continued our work on the application of intuitionistic linear logic to functional programming [1, 2] and have succeeded in providing a much better characterization of our previous results; we gave an efficient, redundancy-free type-inference algorithm and we propose a general and uniform approach to using linear λ-calculi to obtain optimization information about the operational behaviour of functional programs as it relates to storage.

The overall approach is based on a dual λ-calculus: one calculus serves as the programming calculus while the other models execution. The translation between these two calculi uses linear typing proofs based on linear logic; these proofs carry operational information about the translated programs, and can be used for optimization purposes. This method is, however, computationally complex, There are exponentially many distinct proofs that all contribute to the optimization information. This typing ambiguity essentially describes all the different ways in which, in the course of the evaluation of a program, storage operations might be arranged: each proof describes an optimal translation into the deterministic execution calculus. Optimal here means that the number of storage operations is minimized with respect to the types of the variables.

References

[1] A. Tönne. Linear logic meets the Lambda calculus, part I. Technical Report MPI-I-93-258, Max-Planck-Institut für Informatik, Saarbrücken, 1993. Revised version to appear.

• [2] A. Tönne. *An approach to linear logic based program analysis.* PhD thesis, Max-Planck-Institut für Informatik, 1995. (to appear).

8.3.4 Constraint Logic Programming

Investigators: Alexander Bockmayr, Peter Barth, Thomas Kasper

Constraint logic programming (CLP) combines the declarative nature of logic programming with the efficiency of constraint solving over some particular domain of computation like linear real arithmetic, CLP($\mathcal{R}$), Boolean algebra, CLP($\mathcal{B}$), or finite domains, CLP($\mathcal{FD}$). *Finite domain constraints* are used to solve complex combinatorial problems, which is one of the main application areas of CLP. We are working on a constraint logic programming language CLP($\mathcal{PB}$) for logic programming which uses *pseudo-Boolean constraints*, i.e. equations or inequalities between integer polynomials in 0-1 variables. This is both a generalization of Boolean constraints, and a restricted form of finite domain constraints where all domains are equal to $\{0,1\}$; i.e. we have $\mathcal{B} \subset \mathcal{PB} \subset \mathcal{FD}$.

One way to solve pseudo-Boolean constraints is to use standard finite domain techniques, which are based on local consistency and constraint propagation from artificial intelligence. These methods have the drawback that they do not exploit the special structure of 0-1 problems and, moreover, finite domain constraint solvers based on local consistency are not complete: a set of constraints can be locally consistent even though it does not admit a global solution. To get completeness, the values in the domains have to be enumerated by some additional backtracking mechanism (or *labeling* procedure). The lack of global consistency causes semantic problems and also means that it is impossible to talk about the failure of constraints in a program. Moreover in practice pure enumeration may not be efficient enough to solve hard combinatorial problems [4].

In order to overcome the problems of local consistency and pure enumeration we have developed a new approach for 0-1 constraint solving, based on cutting plane techniques from mathematical programming [1, 2, 5, 6]. The basic idea is to solve a constraint set by computing strong valid inequalities or *cutting planes* for its 0-1 solution set S (see Figure 2).

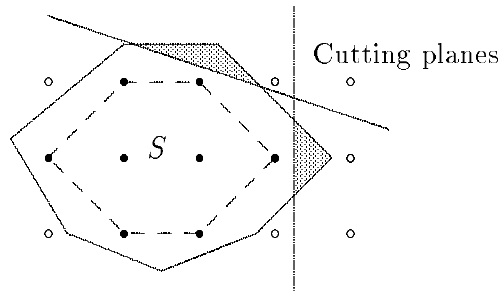

Figure 2: Constraint solving by cutting plane generation

We say that an inequality is valid for S if it is satisfied by all elements of S. An inequality π is stronger than another, π', if the solution set of π is a subset of the solution set of π'. We can compare inequalities using either their 0-1 solutions or their solutions in the non-negative real numbers.

In the first case, the set of strongest valid inequalities is the set Π of *prime inequalities* for S [2], in the second it is the set Γ of *facet-defining inequalities* for the convex hull of S [5]. The sets Π and Γ are viewed as *ideal solved forms*. Constraints are solved by generating cutting planes which give a better approximation of the ideal solved form. The approximation is done in a lazy manner, i.e. the generation of cutting planes is driven by the current state of the logic program. In combination with *branch-and-bound*, we obtain a logic-based or a polyhedral *branch-and-cut* algorithm for pseudo-Boolean constraint solving. Using cutting planes, we can ensure global consistency and improve efficiency of the program by pruning the search space.

The logic-based cutting plane approach has also been used to simplify clausal satisfiability problems in propositional logic [3]. A satisfiability problem is reformulated by adding valid extended clauses that dominate at least one clause in the problem. The reformulated 0-1 problem contains fewer but usually stronger 0-1 inequalities and is typically solved much more quickly.

References

- [1] P. Barth. Linear 0-1 inequalities and extended clauses. Technical Report MPI-I-94-216, Saarbrücken, 1994.

- [2] P. Barth. *Logic-based 0-1 constraint solving in constraint logic programming*. PhD thesis, Fachbereich Informatik, Univ. des Saarlandes, 1994. Forthcoming.

- [3] P. Barth. Simplifying clausal satisfiability problems. In *First International Conference on Constraints in Computational Logics, Munich, Germany*, pages 19–33. Springer, LNCS 845, Sept. 1994.
- [4] P. Barth and A. Bockmayr. Finite domain and cutting plane techniques in CLP(PB). Technical Report MPI-I-94-261, Max-Planck-Institut für Informatik, Saarbrücken, 1994.
- [5] A. Bockmayr. Cutting planes in constraint logic programming. Technical Report MPI-I-94-207, Max-Planck-Institut für Informatik, Saarbrücken, February 1994.
- [6] A. Bockmayr. Solving pseudo-Boolean constraints. In *Constraint Programming: Basics and Trends*. Springer, LNCS, 1995. To appear.

8.4 Higher order logic

One research direction in our group concerns higher-order logics and their applications. In particular, we are interested in using logic as tool to study, implement, and automate other logics and to explore their relationship to programming. Our reasons for this are both theoretical and pragmatic. Theoretically, we are interested in how extensions effect the expressivity of logics; practically, we are looking for methods for making logics more usable (e.g. for interactive theorem proving and theory development) by extending them with new rules or proof procedures. Research closely related to this work addresses logic encodings based on labelled deductive systems; this has a theoretical component, which concerns how natural deduction presentations of modal logics can be embedded within higher-order logical frameworks, but is motivated by the practical concerns of using and automating reasoning in these logics.

Our research also looks at the relationship between higher-order logic and programming. Part of this investigates using logic to develop theories of programs and program synthesis based on higher-order theorem proving. Another aspect concerns logics which serve as programming type systems. In particular we are studying of the second-order λ-calculus with subtyping, and its application to the development of expressive decidable variants suitable for typing object-oriented program languages.

8.4.1 Metatheory in a logical framework

Investigator: Seán Matthews

The initial reason for developing logical frameworks was to provide a system that could be used to implement range of logics in a unform way and make it possible for a user to reuse tools across logics and even to combine different logics together. However we get a bonus: since the implementation is in a well-defined formal theory, we can use that theory to proving, along side the usual theorems of the implemented theory, facts *about* the theory itself; i.e. metatheory.

We have been investigating frameworks where this metatheoretic reasoning can be done, as well as the way that it might make working with a logic easier. The problem with the best known of the proposed logical frameworks (e.g. Isabelle or the LF) is that they are not really designed with metatheory in mind, and allow only very weak reasoning of this sort. We have been considering a very different system, FS_0, proposed by Feferman [2], which *is* intended for doing metatheory. However, FS_0 is in some ways still a very primitive theory, and it is not clear that it can really be used for development work. Thus we have been developing an implementation of the theory, and regard that implementation as a substantial contribution in itself.

FS_0 has two problems in practice: first, while a user is encouraged to think of classes as the extensions of predicates, there is no axiom schema with which we can produce these extensions directly, and they are very difficult to build by hand out of the components provided; second is that, even though FS_0 is designed for encoding logics, it has no built-in facility for dealing with binding and substitution (unlike type-theoretic frameworks).

We have built our implementation (described in [4]) on top of the Isabelle system. The initial decision to use Isabelle has strongly influenced (for the better, we believe) the system we have produced. Our implementation deals with the first problem, of constructing classes, by making extensive use of the techniques described in section 8.4.3 to automate a metatheorem saying what instances of comprehension are provable. The second problem, that there is no binding mechanism available, is not dealt with as part of the implementation, but as a development inside it: we have developed a a very general binding mechanism proposed by Talcott, which is sufficiently first order to be definable. We say 'sufficiently' because it makes uses of higher-order flavoured ideas like homomorphism schemata. However using our system we have found it quite easy to develop such a theory with the 'higher order' facilities provided at the metalevel of the Isabelle logic, instead of being part of the object logic, and this works very well. The theory we have implemented provides a very general binding facility that can be used for implementing arbitrary theories in much the same way as in a type-theoretic framework. Thus, once done, the work of building a binding mechanism need never be repeated.

We have found the combination of Isabelle and FS_0 to be very successful: we have taken FS_0, a theory that seems to be unusably primitive, and build a powerful development system for it, implementing *exactly* the theory that Feferman describes. We have made elaborate use of the facilities that Isabelle provides, to the extent that we believe it would be difficult to produce a custom built system that was anywhere near as effective. Our implementation makes several contributions: first, it is the first convincing demonstration that FS_0 is a usable theory, so long as the right machine support is provided, as proof of this we have been able to develop a complex theory of binding in the system. and impose effectively a very clean structure on the development;[3] second, our implementation can be regarded as a substantial case study implementation of an unusual logic in a framework theorem proving system.

We have also been investigating the sort of metatheorems one can prove with FS_0. Since the original motivation of FS_0 was proof-theoretic, we have been looking there for possible applications, and in [7] we outline how a formal proof of one of the central results of proof theory, cut elimination, for a sequent calculus presentation of propositional logic could be carried out in FS_0. This presentation does not deal with the issues of binding in the encoded language, but rather with how one can formulate a sequent calculus style system in the theory, and how to prove for it a result that uses a complicated nonstructural induction.

Reflection

A special case of metatheoretic reasoning is where a logic is used to encode and reason about itself, then the information produced by this is used to extend the meta level (instead of the object level, like in the approach discussed above) with new results, by adding an axiom to the *meta-level* version of the theory saying essentially that if something is provable in the encoding, then it is true. The result of adding a statement like this (so long as we are careful) can be a theory that has been dramatically proof-theoretically strengthened. This offers an interesting way to increase the strength of a logic in a reliable manner and provides insights into issues of self-reference [1]. However, while for a specially designed framework theory, encoding the logic in itself is easy, in general it is not, and thus we have proposed, in earlier work [5, 6], a way to avoid the effort involved in a full encoding. The abstract properties of an encoding of a logic

[3] We do not believe the implementation in [5] counts as such, being a 'hack' developed for a specific purpose.

are very well known [9], and we have proposed adding these properties to a theory directly, rather than deriving them as theorems. This is a weaker extension than if we go to the effort of building a complete encoding, but it is also much simpler. The question is, how much weaker is it? In [8] we show that this approach works for exactly those theories that have induction over at least what are known as Σ^0_1 formulae, but not over arbitrary first order formulae.

References

[1] S. Feferman. Transfinite recursive progressions of axiomatic theories. *J. Symbolic Logic*, 27:259–316, 1962.

[2] S. Feferman. Finitary inductive systems. In Gabbay [3]. (also appeared in Logic Colloquium '88).

[3] D. Gabbay, editor. *What is a Logical System?* Oxford University Press, Oxford, 1994.

• [4] S. Matthews. Implementing FS_0 in Isabelle: adding structure at the metalevel. Unpublished paper in preparation.

[5] S. Matthews. *Metatheoretic and Reflexive Reasoning in Mechanical Theorem Proving.* PhD thesis, University of Edinburgh, 1992.

[6] S. Matthews. Reflection in a logical system. In A. Yonezawa and B. C. Smith, editors, *Proc. IMSA '92 Workshop on Reflection and Meta-Level Architecture*, pages 178–183, 1992. Also available as Technical Report MPI-I-92-250.

• [7] S. Matthews. A theory and its metatheory in FS_0. In Gabbay [3], chapter 13, pages 329–354.

• [8] S. Matthews and A. Simpson. Reflection using the derivability conditions. To appear as a chapter in the volume in memoriam Roberto Magari.

[9] R. Solovay. Provability interpretations of modal logic. *Israel Journal of Mathematics*, 25:287–304, 1976.

8.4.2 Labelled deductive systems

Investigators: David Basin, Seán Matthews, Luca Viganò

In the face of the enormous range of logics that has been proposed for use in computer science, the notion of a *logical framework* has beed developed, as a foundation for systems that can be used to produce uniform implementations of a wide range of different logics, combine them together, and share tools and interfaces across them. The proposed frameworks however are, for all their aspirations to universality, quite limited in their applications. For instance type theoretic systems like the Edinburgh LF or Isabelle really only work well with logics that can be given a standard natural deduction style presentation; FS_0 suffers from a similar bias, only towards Hilbert presentations of logics. So long as we are interested only in implementing classical or intuitionistic, first-order or higher-order logic, we should have no problems, but there are many 'badly behaved' logics that we might want to implement. For instance if we are interested in modal logic we find that because the deduction theorem fails, a naïve natural deduction presentation is impossible; if we are interested in substructural logics, then thinning or contraction fail, so we have to modify the nature of natural deduction substantially. We might even be interested in non-monotonic logic. The standard logical frameworks have difficulty dealing with all these.

Gabbay, in [4] has proposed a systematic solution to this problem with what he calls *Labelled deductive systems* (LDS). He proposes that, for the purposes of proof theory, formulae be associated with labels generated by some algebra, then formal deductions are performed on the pair of the formula and the label together. The information in the labels can be used in the application of rules, and turns out to be enough to enforce good behaviour on previously badly behaved logics, which can now be implemented in standard frameworks like Isabelle (the 'bad' behaviour of the logic is captured in the algebra defined over the labels, but derivations in the algebra itself are carried out using ordinary logic, which there is no problem implementing). Gabbay has also considered how this approach can be used to combine in a reasonable way, different logics, to build hybrid logics.

We have been looking at how practical it is to use LDS to implement various logics, and have so far been concentrating in particular on modal logics in the Isabelle framework [2]. Our experience has been very positive: we have been able to implement many standard modal logics in uniform manner, parametrised only over the properties of the labelling algebra, which captures the accessibility relation that corresponds to the logic. This allows us to present a large class of propositional modal logics in a structured manner where logics naturally inherit theorems from their ancestors. Equally importantly we have found that LDS implementations of logics are very intuitive to use. While we are not the first to provide a modal logic presentation in a logical framework (Avron [1] and Coen [3], among others, precede us) ours is the first to provide and implement a general system, rather than a few special cases.

We are currently investigating how well the LDS approach applies to other, larger, classes of modal logics, and how it transfers to other logics, such as sub-structural logics. We are also looking at the issues in the second part of Gabbay's proposal: that the LDS approach is an effective foundation for developing technology for combining different logics together.

References

[1] A. Avron, F. Honsell, I. Mason, and R. Pollack. Using typed lambda calculus to implement formal systems on a machine. *J. Automated Reasoning*, 9:309–352, 1992.

• [2] D. Basin, S. Matthews, and L. Vigano. LDS implementations of modal logics in a natural deduction logical framework. Unpublished paper in preparation.

[3] M. Coen. A sequent calculus implementation of S4 for Isabelle. Undocumented package distributed with the Isabelle system, 1991.

• [4] D. Gabbay. Labelled Deductive Systems. Technical Report MPI–I–94-223.

8.4.3 Program synthesis

Investigator: David Basin

An important problem in computer science is program correctness: showing that a program meets a specification or deriving a program to meet a specification. There have been many logics and approaches proposed for these problems and implementing them is a considerable task, as is understanding their correctness and automating derivations in them. Our work is motivated by these practical concerns: we seek to develop a common framework for formalizing calculi with the aim of synthesizing programs (e.g. functional programs like ML or logic programs such as Prolog) from specifications of their behavior.

We have shown how logical frameworks can be used to formalize program development calculi. This is a two level approach: we embed a development logic, like first-order logic or type theory, within a framework logic (a higher-order logic), and then use the framework logic to extend the development logic with new, proven correct proof rules which may be used for program development.

After deriving development calculi, we use them not only to verify programs, but also to calculate programs during correctness proofs. The first activity, verification, consists of showing that a program is correct by specifying in the development calculus the relationship between the program and its specification, then building a proof that the program, in some technical sense, realizes the specification. Our approach to synthesis is similar, but, rather than verifying a specific program, in essence we use higher-order metavariables as 'placeholders' and instantiate these with programs during the course of the correctness proof. This approach can be seen as carrying out a correctness proof where we do not, at first, commit to the program we are verifying, and it becomes constrained to a particular concrete program during proof. Concrete details such as how metavariables are instantiated and substitutions are propagated are handled by the logical framework.

This can be understood by analogy with Prolog, where one proves a predicate like $r(t)$ by building a proof in a Horn clause theory. Alternatively, if we pose a query $r(X)$ the same proof will be built through unification, with t as the satisfying term. The first is verification, the second synthesis. Our setting is a bit more complex since we use not Horn clauses but arbitrary derived rules in a programming logic, and proofs are constructed not automatically by SLD resolution, like in Prolog, but interactively using a more complicated form of resolution. The problems we tackle are also complicated by the fact that we are trying to build recursive programs and therefore require mathematical induction.

Our use of logical frameworks to formalize and derive programming calculi is new. So is the use of higher-order resolution as a means of recasting and simplifying previously proposed calculi. Currently we are using Paulson's Isabelle framework to derive calculi for logic program synthesis and combinational circuit synthesis (based on Hanna's *Formal Synthesis* calculus) [9, 1, 4, 3]. Our work has not only resulted in a machine checked account of the correctness of these calculi but also simplifications and extensions. We have used these calculi to synthesize formally verified logic programs and circuit descriptions. We have also addressed automation of proof in this setting. In particular, we have been able to incorporate strategies from inductive theorem proving (Bundy's rippling calculus) to automate induction and simplification during program synthesis [8, 7, 5].

References

- [1] D. Basin. Logic frameworks for logic programs. In *4th International Workshop on Logic Program Synthesis and Transformation, (LOPSTR'94)*, pages 1–16, Pisa, Italy, June 1994. Springer-Verlag, LNCS 883.

- [2] D. Basin and S. Matthews. A conservative extension of first-order logic and its applications to theorem proving. In R. K. Shyamasundar, editor, *13th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes in Computer Science*, Bombay, 1993. Springer, Berlin. also available as MPI-93-235.

- [3] D. Basin and S. Matthews. Adding metatheoretic facilities to first-order theories. Submitted to journal (this is a heavily revised and corrected version of [2]), 1994.

- [4] D. Basin and S. Matthews. Algebraic factorisation and ripple-carry iteration. Submitted to conference, 1994.

- [5] D. Basin and T. Walsh. Termination orderings for rippling. In *Proc. of 12th International Conference On Automated Deduction (CADE-12)*. Springer-Verlag, June 1994.

- [6] D. Basin and T. Walsh. A calculus for rippling. In *Proc. of Workshop on Conditional Term Rewriting (CTRS-94)*. Springer Verlag, 1995. To appear.

[7] D. A. Basin and T. Walsh. Difference unification. In R. Bajcsy, editor, *Proc. 13th Intern. Joint Conference on Artificial Intelligence (IJCAI '93)* (Chambery, France), volume 1, pages 116–122, San Mateo, CA, 1993. Morgan Kaufmann. Also available as Technical Report MPI-I-92-247.

- [8] I. Kraan, D. Basin, and A. Bundy. Middle-out reasoning for synthesis and induction. Submitted to the Journal of Automated Reasoning.

[9] I. Kraan, D. Basin, and A. Bundy. Middle-out reasoning for logic program synthesis. In *Proc. 10th Intern. Conference on Logic Programing (ICLP '93)* (Budapest, Hungary), pages 441–455, Cambridge, MA, 1993. MIT Press. Also available as Technical Report MPI-I-93-214.

8.4.4 Automated Reasoning in Higher-Order Logic

Investigator: David Basin

Higher-order logic is one of the more powerful logics available for formalizing mathematics and program development. Unfortunately, almost all theorem provers for this logic are interactive tactic based provers. The few fully automatic systems which have been built (e.g. Andrews' TPS) have had limited success due to the large proof search-space.

We have been investigating theorem proving in sublogics and their integration with higher-order theorem provers. In [1], we show how a second-order monadic theory of strings can be used to specify hardware components and their behavior. This logic admits a decision procedure and counter-model generator based on canonical automata for formulas. We have used MONA, a system implementing these procedures, as a tool to analyze such circuits and have verified, or found errors in a number of circuits proposed in the literature. The techniques we use make it easier to identify regularity in circuits, including those that are parameterized or have parameterized behavioral specifications. For example, parameterized adders, or synchronized circuits. Theorem proving with MONA is semantic and does not require lemmas or induction as would be needed when reasoning about these circuits directly in higher-order logic. The logic of MONA itself can easily be embedded in higher-order logic, so it is possible to use MONA as an oracle to solve a large and useful class of decidable higher-order problems.

References

- [1] D. Basin and N. Klarlund. Hardware verification using monadic second-order logic. Submitted to the Seventh Conference on Computer-Aided Verification (CAV '95).

8.4.5 Investigations on Polymorphic λ-Calculi with Subtyping

Investigator: Sergei Vorobyov

The advantages and usefulness of strict typing disciplines in programming with static typing and rigid compile-time type control have been widely accepted, studied, and advocated in software engineering since the sixties, and is reflected in the creation of programming languages like Algol-68, Pascal, ML, Ada, etc. Typed programming should be based on powerful and, preferably, decidable type systems.

The system $F_{\leq}$ is the polymorphic second-order typed λ-calculus with subtyping, combining the universal (or parametric) polymorphism of Girard's system **F** with Cardelli's calculus of subtyping (inheritance polymorphism). $F_{\leq}$ serves as a core calculus of type systems with subtyping and a model for representing polymorphic and object-oriented features in programming languages.

$F_{\leq}$ extends Girard's system **F** with: 1) the additional subtype relation on polymorphic types, 2) the possibility of imposing type bounds in universal type quantification, and 3)o more general typing rules, that take the subtyping relation into account. The presence of subtyping means that terms in $F_{\leq}$ can have infinitely many types, unlike the Church-style simply typed lambda-calculus or the system **F**. In 1992 Pierce proved that this subtyping is undecidable and undecidability of the $F_{\leq}$ subtyping implies the undecidability of typing in $F_{\leq}$.

Rather than weakening the $F_{\leq}$ subtyping (to get decidability), we show that it possesses infinitely many decidable extensions, which we prove by interpreting the $F_{\leq}$ subtyping in M.Rabin's monadic second-order logic of successor functions [3]. However decidability of subtyping does not immediately imply the decidability of the associated typing relation: there exist systems with decidable subtyping and open decidability problem for typing. Decidability of typing is closely connected to the possibility of normalizing typing proofs, i.e. transforming them to unique canonical forms. We show how our extensions of the $F_{\leq}$ subtyping could be incorporated into decidable typing systems extending $F_{\leq}$, and prove the typing proof normalization and the subject reduction theorems for these extensions [2]. Our results generalize to $F_{\leq}$ enriched with recursive types, as shown in [1].

References

- [1] S. Vorobyov. $F_{\leq}$ with recursive types: 'Types-As-Propositions' interpretations in M.Rabin's S2S. In *Journées Francophones des Langages Applicatifs (JFLA'95)*. INRIA, 1995.

- [2] S. Vorobyov. Proof normalization and subject reduction in extensions of $F_{\leq}$. Technical Report MPI–I–95–2–001, 1995.

- [3] S. Vorobyov. Structural decidable extensions of bounded quantification. In *Proceedings of the 22nd ACM Symp. on Principles of Programming Languages*. ACM, 1995.

8.5 Other work

This section contains descriptions of work done by people in the group that does not fit with any of the major research headings we give above.

8.5.1 Program Synthesis

Investigator: David Plaisted

In [1] we present a framework for constructing programming meta-logics as enrichments of some underlying logic. The heart of the system is a systematic method for reasoning about recursion, composition, and fixpoints. Our logic, like others, represents programs as proofs, and programs satisfying a certain specification can be extracted from a proof. However, our emphasis is not on automatically constructing programs using a theorem prover, but on representing them in as abstract a manner as possible so as to facilitate their reuse in different settings. This seems to be of practical importance and also more feasible than automatic program derivation, given the current state of automated reasoning.

The logic is distinguished from others by the way it separates classical and computational aspects, and also by its independence from the underlying logic. The extended logic introduces new program-construction variables into the underlying logic and some constructive inference rules for these variables. No requirements of constructiveness are imposed on the underlying logic. For instance, first-order logic with a sort structure can be used for many applications. The logic does not specify a particular syntax for the programming language, and permits considerable freedom in the underlying computational mechanism; whether deterministic or nondeterministic, functional or relational, terminating or non-terminating, etc. Thus the system is to a large degree independent of the syntax and semantics of the programming language and also from the underlying logic. In this way we obtain a program generation logic with a high degree of abstractness. This flexibility makes it easy to tailor the logic for specific applications. This also allows for the possibility of translations between different such logics.

References

- [1] D. A. Plaisted. An abstract program generation logic. Technical Report MPI-I-94-232, 1994.

8.5.2 Applying Algebraic Specification Techniques to the Specification of Dynamic Systems

Investigator: Hubert Baumeister

The theory of abstract datatypes is well developed and has a rigorous formal basis. However, trying to apply abstract datatype techniques to the specification of dynamic systems one finds that an important aspect of dynamic systems, their changing state, is not well covered by these techniques.

The state of dynamic systems can be modeled as algebras, operations changing the state of a dynamic system as relations between algebras. In [1] we have defined an institution such that abstract datatypes in this institution can be interpreted as relations between algebras. This allows to apply the theory of abstract datatypes to the modeling of the behaviour of dynamic systems. We shall use the view of relations as abstract datatypes to explore the dynamics of objects and object systems. We would like to use this approach to give a formal semantics of the specification language Z within the framework of abstract datatypes.

References

- [1] H. Baumeister. Relations as abstract datatypes: An institution to specify relations between algebras. In *TAPSOFT/FASE, Proceedings of the Sixth International Joint Conference on the Theory and Practice of Software Development, Colloquium on Formal Approaches in Software Engineering*, Aarhus, Denmark, May 1995. Springer.

8.5.3 Data Compression with Genetic Algorithms

Investigator: Jörn Hopf

Genetic algorithms, like evolutionary programming and evolution strategies, is a subarea of evolutionary algorithms. Evolutionary computation does not replace other methods. For traditional applications, evolutionary algorithms can not do as good a job as traditional methods. Rather, evolutionary computation should be considered if other methods either do not exist, are not applicable, or fail. Today they play a considerable role in Artificial Life, a research area that has emerged from classical Artificial Intelligence, control, planning, combinatorial optimization and many other areas.

We are exploring the use of genetic algorithms for data compression. Our aim is to achieve a very high compression rate without a loss of information. We consider data as a point set. A '1' is represented by a black pixel, '0' by a white pixel. Monochrome pictures are already presented in this way. Using a genetic algorithm we try to find fractals representing parts of this point set. These fractals can be represented by a set of parameters (normalized real numbers) which needs many fewer bits then the original bitmap.

There are two major problems to solve: First, we have to find an appropriate coding of the solution of our problem, so that we can treat it using a genetic algorithm, and second, we have to find a way to determine distances in arbitrary point sets as an evaluation function for the genetic algorithm. Since the parameters and the point set represented by them have a non-linear relation we evaluate the fractals with respect to their parameters through neural networks. Using this criterion the genetic algorithm selects individuals for the next generation. This evaluation turns out to be the most important point. Even if the compression process takes longer than traditional methods, the decompression is extremely fast, and thus useful for frequently retransmitted data.

8.6 Implementations

8.6.1 ACID

ACID is 'A Collection of Indexing Data structures'. As described in Section 8.1.5, it is designed to provide efficient support for a variety of different tasks in automated reasoning. For instance, in order to find resolution partners for a given literal, a theorem prover has to search for all occurrences of literals which are unifiable with a given literal. Subsumption of clauses can be detected by the retrieval of generalizations (forward subsumption) or instances (backward subsumption) of literals of clauses. The retrieval of rewrite rules and demodulators can be accelerated by indexing as well.

ACID is available via anonymous ftp. Connect to `ftp.mpi-sb.mpg.de`; the current version with manual can be found in `/pub/tools/deduction/ACID`. In case of problems contact `acid@mpi-sb.mpg.de`.

ACID is implemented in C. but can also be used from Prolog programs via appropriate foreign language interfaces. The term indexing methods are implemented on top of an abstract data type for term construction and access which a user may instantiate by a term module of his/her choice.

8.6.2 The SAXOPHONE prover

In 1992 Ulrich Aßmann [1] proposed an implementation scheme for distributed hyperresolution. In his approach each clause is represented by a process. The number of processes during a run of the prover is constant, since new resolvents are represented by substitutions which are sent to clauses that possibly contain complementary literals. Subsumption is decentralized in the clause processes in order to avoid bottlenecks.

The SAXOPHONE system is an implementation in C of this approach restricted on horn clauses. We use the PVM library [2] for the communication protocol.

The main purpose of our work is to achieve maximum speed performance by massive parallel distribution and advanced indexing techniques. The most critical parts of distributed hyperresolution are process communication, subsumption, and most of all the computation of simultaneous unifiers.

Process communication is extended on sets of substitutions which are represented by substitution trees [3]. Substitution tree indexing provides the algorithms for subsumption and the search for simultaneous unifiers on sets of substitutions.

References

[1] U. Aßmann. *Parallele Modelle für Deduktionssysteme.* PhD thesis, Infix, Köln, 1992.

[2] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM 3 User's Guide and Reference Manual.* Oak Ridge National Laboratory, May 1993. ORNL/TM-12187, pvm@msr.epm.ornl.gov.

• [3] P. Graf. Substitution tree indexing. Technical Report MPI-I-94-251, Max-Planck-Institut fuer Informatik, Saarbruecken, October 1994.

8.6.3 The MOTEL system

The MOTEL system [6, 2] is a prototypical implementation of our approach to knowledge representation for multiple agents (see section 8.2.8 of this report). MOTEL translates modal terminological logic theories into Prolog logic programs or theories for the theorem prover SETHEO.

The representation language of MOTEL contains as a kernel the language $\mathcal{ALCNR}$ [1] which is a decidable sublanguage of first-order predicate logic. Whereas $\mathcal{ALCNR}$ is a single-agent knowledge representation system, i.e. is only able to represent general world knowledge or the knowledge of one agent about the world, MOTEL is a multi-agent knowledge representation system. The MOTEL language allows modal contexts and modal concept forming operators which allow to represent and reason about the beliefs and desires of multiple agents [4, 5].

For the target language Prolog, it has been easy to implement further inferential facilities:

- Hustadt [3] describes a goal-oriented method for abduction in disjunctive logic programs. The results have been used to provide abduction for terminological logics in MOTEL.
- Using the revision operators of Prolog, belief revision has been implemented on the level of axioms.
- Using the correspondence between default theories and general logic programs on the semantical level, we have been able to integrate default reasoning in MOTEL.

As a result, in terms of expressiveness of the language and variety of inferential operations, MOTEL is one of the the most advanced systems available, for handling knowledge representation for multiple agents.

A description of the MOTEL system as well as the sources of the system are accessible via WWW at the URL

`http://www.mpi-sb.mpg.de/guide/staff/hustadt/motel/system.html`

References

[1] F. Baader and B. Hollunder. $\mathcal{KRIS}$: Knowledge Representation and Inference System. system description. Technical Memo DFKI-TM-90-03, Deutsches Forschungszentrum für Künstliche Intellige nz, Saarbrücken, Germany, 1990.

• [2] D. Fehrer, U. Hustadt, M. Jaeger, A. Nonnengart, H. J. Ohlbach, R. A. Schmidt, C. Weidenbach, and E. Weydert. Description logics for natural language processing. In *International Workshop on Description Logics '94*, pages 80–84, Bonn, Germany, 1994. DFKI.

[3] U. Hustadt. Abductive disjunctive logic programming. In P. Codognet, P. M. Dung, A. C. Kakas, and P. Mancarella, editors, *ICLP '93 Postconference Workshop on Abductive Reasoning* (Budapest, Hungary), Cambridge, MA, 1993. MIT Press.

• [4] U. Hustadt. Common and mutual belief for agent modeling. To appear in *Knowledge and Belief in Philosophy and Artificial Intelligence*, Akademie-Verlag, Berlin, 1995, 1994.

• [5] U. Hustadt. A multi-modal logic for user modeling. In *Proceedings of the Fourth International Conference on User Modeling UM94* (Hyannis, MA), pages 87–92, Bedford, MA, 1994. The MITRE Corporation.

[6] U. Hustadt, A. Nonnengart, R. Schmidt, and J. Timm. MOTEL user manual. Technical report MPI-I-92-236, Max Planck Institute for Computer Science, Saarbrücken, Germany, June 1992.

8.6.4 The Quantifier Elimination Algorithm SCAN

Problems like computing first-order circumscription or computing corresponding frame properties for logical axioms reduce to the problem of finding for a formula $\exists p\ \varphi$ with existentially quantified predicate variables p, a first-order formula φ' which is equivalent to $\exists p\ \varphi$, if possible. The algorithm we have developed for this purpose (we called it SCAN) is implemented and it is provided with different kinds of interfaces. Besides the direct interface for treating second-order formulae, there is an interface for computing the representation from the axioms of various algebras and logics with functions, and an interface for computing first-order circumscription in the McCarthy style. These interfaces are accessible via world wide web (WWW) and the program can be used remotely (world wide) by just filling out some HTML forms. The WWW address is

`http://www.mpi-sb.mpg.de/guide/staff/ohlbach/scan/scan.html`

This is the first implemented algorithm which provides this kind of functionality.

8.6.5 CLP

A fully functional prototype of CLP($\mathcal{PB}$) based on logic-cut methods has been implemented on top of Prolog [1] in order to experiment with a CLP-system providing global consistency and a solved form. An efficient implementation in C++ of a logic-cut based constraint solver for pseudo-Boolean constraints, which can be linked into logic programming systems is currently under development.

An implicit enumeration algorithm for solving linear pseudo-Boolean constraints, which is part of the logic-cut based pseudo-Boolean constraint solver, has been implemented and extended to an optimization procedure [2]. The algorithm compares well with linear programming based methods on a variety of standard benchmarks found in MIPLIB [3].

References

• [1] P. Barth. *A Short Guide to CLP($\mathcal{PB}$)*, 1994. System available by anonymous ftp from `ftp.mpi-sb.mpg.de` in directory `pub/tools/CLPPB/clppb.tar.Z`.

• [2] P. Barth. A Davis-Putnam based enumeration algorithm for linear 0-1 optimization. Technical Report MPI-I-95-2-003, 1995.

[3] R. E. Bixby, E. A. Boyd, and R. Indovina. MIPLIB: A Test Set of Mixed-Integer Programming Problems. *SIAM News*, 25(16), 1992.

8.6.6 PROP

PROP is the implementation of a decision procedure for SAT, the problem whether a set of propositional clauses is satisfiable. The decision procedure is based on tableau extended with sorts (see 8.1.3). Our first prototype implementation in C produced promising results: It can compete with the best algorithms for SAT on standard benchmarks, e.g. implementations of the Davis-Putnam [1, 2] procedure. Currently we are implementing additional redundancy criteria and we are integrating the procedure into a first-order refutation procedure. PROP will be available via anonymous ftp during April from `ftp.mpi-sb.mpg.de` in the directory `/pub/tools/deduction/PROP`.

References

[1] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–205, 1960.

[2] D. Loveland. *Automated Theorem Proving: A Logical Basis*, volume 6 of *Fundamental Studies in Computer Science*. North-Holland, 1978.

8.6.7 The Saturate System

The Saturate system is an experimental theorem prover based on saturation. It was originally been developed as an implementation of the superposition calculus by Pilar Nivela and Robert Nieuwenhuis from the Technical University of Catalonia, Barcelona, while they were visiting our institute in 1992. We have extended and modified the system by generalizing superposition to our chaining techniques for arbitrary transitive relations that we have described in Section 8.1. The system supports saturation with various strategies for the selection of negative literals. It contains a complete constraint solver for the LPO, together with a generic (generally incomplete) constraint solver for arbitrary reduction orderings. Its main strength lies in its techniques for proving the redundancy of generated clauses and inferences. We provide complete tautology checking (modulo the built-in theories) by a ground version of saturation. Some of the more sophisticated techniques are case analysis over the possible orderings on variables in redundancy proofs and contextual rewriting for reducing clauses with non-unit clauses. At present the system becomes very slow if more than a few hundreds of clauses are generated. We are currently rewriting it to make use of the efficient datastructures provided by ACID. Using redundancy elimination techniques we have managed to prove a number of 'challenge theorems' while producing fewer than 100–300 clauses in total, meaning that response remained tolerable. This provides evidence of the effectiveness and usefulness of our theoretical results.

The system is written in Prolog and runs under Quintus and SICStus. More information is available under

`http://www.mpi-sb.mpg.de/SATURATE/Saturate.html`

9 Publications

9.1 Journals and Book Chapters

The following journal articles or book chapters have been produced by the group in the last year:

[1] F. Baader and H. J. Ohlbach. A multi-dimensional terminological knowledge representation language. *Journal of Applied Non-Classical Logics*, 1995. forthcoming.

[2] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

[3] L. Bachmair, H. Ganzinger, and U. Waldmann. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing*, 5(3/4):193–212, Apr. 1994.

[4] D. Basin. A term equality problem equivalent to graph isomorphism. *Information Processing Letters*, 51:61 – 66, 1994.

[5] A. Bockmayr, S. Krischer, and A. Werner. Narrowing strategies for arbitrary canonical systems. *Fundamenta Informaticae*, 1994. To appear.

[6] C. Brink, K. Britz, and R. A. Schmidt. Peirce algebras. *Formal Aspects of Computing*, 6:1–20, 1994.

[7] C. Brink, D. Gabbay, and H. J. Ohlbach. Towards automating duality. *Journal of Computers and Mathematics with Applications*, 29(2):73–90, 1994. Full version available as MPI-I-93-220.

[8] R. Chadha and D. A. Plaisted. Correctness of unification without occur check in prolog. *Journal of Logic Programming*, 18:2:99–122, 1994.

[9] Y. Dimopoulos and V. Magirou. A graph theoretic approach to default logic. *Information and Computation*, 112(2), 1994.

[10] D. Gabbay and H. J. Ohlbach, editors. *Temporal Logic: Proceedings of the First International Conference on Temporal Logic*, volume 827 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer Verlag, 1994.

[11] M. Hanus. The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, 19&20:583–628, 1994.

[12] M. Hanus. Analysis of residuating logic programs. *Journal of Logic Programming*, 1995. (to appear).

[13] M. Hanus. Compile-time analysis of nonlinear constraints in CLP($\mathcal{R}$). *New Generation Computing*, 1995. (to appear).

[14] J. Hopf and F. Klawonn. Learning the rule base of a fuzzy controller by a genetic algorithm. In R. Kruse, R. Palm, and J. Gebhardt, editors, *Fuzzy Systems in Computer Science*, Künstliche Intelligenz, pages 63–74. Vieweg, Braunschweig, Germany, 1994.

[15] U. Hustadt. Common and mutual belief for agent modeling. To appear in *Knowledge and Belief in Philosophy and Artificial Intelligence*, Akademie-Verlag, Berlin, 1995, 1994.

[16] P. Madden, A. Bundy, and A. Smaill. Recursive Program Optimization Through Inductive Synthesis Proof Transformation. *Journal of Automated Reasoning*, 1995. To appear. Also available as Technical Report MPI-I-94-239.

[17] S. Matthews. A theory and its metatheory in FS_0. In D. Gabbay, editor, *What is a Logical System?*, chapter 13, pages 329–354. Oxford University Press, Oxford, 1994.

[18] S. Matthews and A. Simpson. Reflection using the derivability conditions. To appear as a chapter in the volume in memoriam Roberto Magari.

[19] H. J. Ohlbach, R. A. Schmidt, and U. Hustadt. Translating graded modalities into predicate logic. To appear in *Knowledge and Belief in Philosophy and Artificial Intelligence*, Akademie-Verlag, Berlin, 1995, January 1995.

[20] H. J. Ohlbach and C. Weidenbach. A note on assumptions about skolem functions. *Journal of Automated Reasoning*, 1995. Forthcoming.

[21] A. Podelski, editor. *Constraint Programming: Basics and Trends*, volume 910 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, New York, 1995. To appear.

9.2 Conferences

The following papers have appeared in conference proceedings:

[1] S. Antoy, R. Echahed, and M. Hanus. A needed narrowing strategy. In *Proc. of the 21st ACM Symposium on Principles of Programming Languages (POPL'94* (Portland)*)*, pages 268–279, New York, 1994. ACM Press. Extended version available as Technical Report MPI-I-93-243.

[2] L. Bachmair and H. Ganzinger. Associative-commutative superposition. Technical Report MPI-I-93-267, Max-Planck-Institut für Informatik, Saarbrücken, 1993. To appear in Proc. CTRS Workshop 1994, LNCS.

[3] L. Bachmair and H. Ganzinger. Buchberger's algorithm: a constraint-based completion procedure. In *1st Internal Conference on Constraints in Computational Logics*, volume 845 of *Lecture Notes in Computer Science*, pages 285–301. Springer-Verlag, 1994.

[4] L. Bachmair and H. Ganzinger. Ordered chaining for total orderings. In *Proc. 12th International Conference on Automated Deduction*, LNAI, pages 435–450. Springer, 1994. Full version available as MPI-I-93-250.

[5] L. Bachmair and H. Ganzinger. Rewrite techniques for transitive relations. In *Proc. 9th IEEE Symposium on Logic in Computer Science*, pages 384–393. IEEE Computer Society Press, 1994. Full version available as Technical Report MPI-I-93-249.

[6] L. Bachmair, H. Ganzinger, and J. Stuber. Combining algebra and universal algebra in first-order theorem proving: The case of commutative rings. In *Proc. 10th Workshop on Specification of Abstract Data Types*, LNCS. Springer, 1995. To appear.

[7] P. Barth. Simplifying clausal satisfiability problems. In *First International Conference on Constraints in Computational Logics, Munich, Germany*, pages 19–33. Springer, LNCS 845, Sept. 1994.

[8] P. Barth and A. Bockmayr. Global consistency in CLP(PB). In *10th Workshop Logic Programming WLP'94, Zurich*, 1994.

[9] P. Barth and A. Bockmayr. Finite domain and cutting plane techniques in CLP($\mathcal{PB}$). In *International Conference of Logic Programming, ICLP'95, Tokyo*, 1995.

[10] D. Basin. IsaWhelk: Whelk interpreted in Isabelle. In *11th International Conference on Logic Programming (ICLP94)*, 1994. Paper is extended abstract.

[11] D. Basin. Logic frameworks for logic programs. In *4th International Workshop on Logic Program Synthesis and Transformation, (LOPSTR'94)*, pages 1–16, Pisa, Italy, June 1994. Springer-Verlag, LNCS 883.

[12] D. Basin and S. Matthews. A conservative extension of first-order logic and its applications to theorem proving. In R. K. Shyamasundar, editor, *13th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes in Computer Science*, Bombay, 1993. Springer, Berlin. also available as MPI-93-235.

[13] D. Basin and T. Walsh. Termination orderings for rippling. In *Proc. of 12th International Conference On Automated Deduction (CADE-12)*. Springer-Verlag, June 1994.

[14] D. Basin and T. Walsh. A calculus for rippling. In *Proc. of Workshop on Conditional Term Rewriting (CTRS-94)*. Springer Verlag, 1995. To appear.

[15] H. Baumeister. Relations as abstract datatypes: An institution to specify relations between algebras. In *TAPSOFT/FASE, Proceedings of the Sixth International Joint Conference on the Theory and Practice of Software Development, Colloquium on Formal Approaches in Software Engineering*, Aarhus, Denmark, May 1995. Springer.

[16] A. Bockmayr. Cutting planes in constraint logic programming (Abstract). In *3rd Intern. Symp. Artificial Intelligence and Mathematics, Ft. Lauderdale, Florida*, 1994.

[17] A. Bockmayr. Using strong cutting planes in constraint logic programming. In *Operations Research '93, 18th Symposium on Operations Research*, pages 47 – 49, 1994.

[18] A. Bockmayr. Solving pseudo-Boolean constraints. In *Constraint Programming: Basics and Trends*. Springer, LNCS, 1995. To appear.

[19] A. Bockmayr and A. Werner. LSE narrowing for decreasing conditional term rewrite systems. In *Conditional Term Rewriting Systems CTRS'94, Jerusalem*, 1994.

[20] W. Charatonik and L. Pacholski. Negative set constraints. In *Proc. 9th IEEE Symposium on Logic in Computer Science*, pages 128–136. IEEE Computer Society Press, 1994.

[21] H. Chu and D. A. Plaisted. Semantically guided first-order theorem proving using hyper-linking. In *Proceedings of the Twelfth International Conference on Automated Deduction*, pages 192–206, 1994. Lecture Notes in Artificial Intelligence 814.

[22] Y. Dimopoulos. Classical methods in nonmonotonic reasoning. In Z. Ras and M. Zemankova, editors, *International Symposium on Methodologies for Intelligent Systems*, LNAI, 1994. to appear.

[23] Y. Dimopoulos. The computational value of joint consistency. In D. Pearce and L. Pereira, editors, *European Workshop on Logics in AI*, LNAI, 1994.

[24] P. Graf. Extended path–indexing. In *Proceedings of CADE-12*, volume 814 of *LNAI*, pages 514–528. Springer, 1994. Full version available as MPI-I-93-253.

[25] P. Graf. Substitution tree indexing. In *Proc. 6th Int. Conf. on Rewriting Techniques and Applications*. Springer, 1995. Full version available as MPI-I-94-251.

[26] M. Hanus. Combining lazy narrowing and simplification. In M. Hermenegildo and J. Penjam, editors, *Proc. 6th International Symposium on Programming Language Implementation and Logic Programming (Madrid, Spain)*, volume 844 of *Lecture Notes in Computer Science*, pages 370–384, Berlin, 1994. Springer-Verlag.

[27] M. Hanus. Lazy unification with simplification. In D. Sannella, editor, *Proc. 5th European Symposium on Programming*, volume 788 of *Lecture Notes in Computer Science*, pages 272–286, Berlin, 1994. Springer-Verlag.

[28] M. Hanus. Towards the global optimization of functional logic programs. In P. Fritzson, editor, *Proc. 5th International Conference on Compiler Construction (Edinburgh)*, volume 786 of *Lecture Notes in Computer Science*, pages 68–82, Berlin, 1994. Springer-Verlag.

[29] M. Hanus and F. Zartmann. Mode analysis of functional logic programs. In B. L. Charlier, editor, *Proceedings of the First International Static Analysis Symposium (Namur, Belgium)*, volume 864 of *Lecture Notes in Computer Science*, pages 26–42, Berlin, 1994. Springer-Verlag.

[30] J. Hopf and F. Klawonn. Learning the rule base of a fuzzy controller by a genetic algorithm. In R. Kruse, R. Palm, and J. Gebhardt, editors, *Fuzzy Systems in Computer Science*, Künstliche Intelligenz, pages 63–74. Vieweg, Braunschweig, Germany, 1994.

[31] U. Hustadt. Do we need the closed-world assumption in knowledge representation. In F. Baader and M. A. Jeusfeld, editors, *Reasoning about Structured Objects: Knowledge Representation meets Databases. Workshop during the 18th German Annual Conference on Artificial Intelligence (KI-94)*, 1994.

[32] U. Hustadt. A multi-modal logic for user modeling. In *Proceedings of the Fourth International Conference on User Modeling UM94* (Hyannis, MA), pages 87–92, Bedford, MA, 1994. The MITRE Corporation.

[33] M. Jaeger. A logic for default reasoning about probabilities. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1994.

[34] M. Jaeger. Probabilistic reasoning in terminological logics. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Principles of Knowledge Representation an Reasoning: Proceedings of the Fourth International Conference (KR94)*. Morgan Kaufmann, San Mateo, CA, 1994.

[35] M. Krishna Rao. Semi-completeness of hierarchical and super-hierarchi cal combinations of term rewriting systems. In M. Nielsen, editor, *Trees in Algebra and Programming – CAAP'95*. Springer, 1995.

[36] P. Madden. Formal methods for automated program improvement. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence. Proceedings of 18th German Annual Conference on Artificial Intelligence*. Springer, 1994.

[37] P. Madden and I. Green. A general technique for automatic optimization by proof planning. In *Proceedings of Second International Conference on Artificial Intelligence and Symbolic Mathematical Computing (AISMC-2)*, King's College, Cambridge, England. Springer Verlag.

[38] A. Nonnengart. How to Use Modalities and Sorts in Prolog. In *Proceedings of the JELIA'94, LNAI 838, Logics in Artificial Intelligence*. Springer Verlag, 1994.

[39] D. A. Plaisted. The search efficiency of theorem proving strategies. In *Proc. 12th Int. Conf. on Automated Deduction*, LNAI, pages 57–71. Springer, 1994. Full verssion available as MPI–I–94–233.

[40] A. Podelski and G. Smolka. Operational semantics of constraint logic programming with coroutining. In L. Sterling, editor, *Proceedings of the 12th International Conference on Logic Programming*, Kanagawa, Japan, 1995. The MIT Press. To appear.

[41] J. Stuber. Computing stable models by program transformation. In P. Van Hentenryck, editor, *Proc. 11th Int. Conf. on Logic Programming*, pages 58–73, Santa Margherita Ligure, Italy, 1994.

[42] S. Vorobyov. Structural decidable extensions of bounded quantification. In *Proceedings of the 22nd ACM Symp. on Principles of Programming Languages*. ACM, 1995.

[43] C. Weidenbach. First-order tableaux with sorts. In K. Broda and M. D. et.al., editors, *TABLEAUX-'94, 3rd Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 247–261. Imperial College of Science Technology and Medicine, TR-94/5, April 1994.

[44] A. Werner, A. Bockmayr, and S. Krischer. How to realize LSE narrowing. In *Algebraic and Logic Programming, ALP'94, Madrid*, pages 59 – 76. Springer, LNCS 850, 1994.

[45] E. Weydert. General belief measures. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1994.

[46] T. Yoshida, A. Bundy, I. Green, T. Walsh, and D. Basin. Coloured rippling: An extension of a theorem proving heuristic. In *ECAI-94*. John Whiley and Sons, 1994.

9.3 Reports

The following papers have appeared as Institute publications or internal reports.

[1] A. Bockmayr, 1994. Cutting planes in constraint logic programming. Technical Report MPI-I-94-207, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract In this paper, we show how recently developed techniques from combinatorial optimization can be embedded into constraint logic programming. We develop a constraint solver for the constraint logic programming language CLP($\mathcal{PB}$) for logic programming with pseudo-Boolean constraints. Our approach is based on the generation of polyhedral cutting planes and the concept of branch-and-cut. In the case of 0-1 constraints, this can improve or replace the finite domain techniques used in existing constraint logic programming systems.

[2] H. Aït-Kaci, M. Hanus, J. J. Moreno Navarro (editors), 1994. Integration of declarative paradigms: Proceedings of the ICLP'94 post-conference workshop. Technical Report MPI-I-94-224.

[3] P. Barth, A. Bockmayr, 1994. Finite domain and cutting plane techniques in CLP(PB). Technical Report MPI-I-94-261, Max-Planck-Institut für Informatik, Saarbrücken.

> **Abstract** Finite domain constraints are one of the most important constraint domains in constraint logic programming. Usually, they are solved by local consistency techniques combined with enumeration. We argue that, in many cases, the concept of local consistency is too weak for both theoretical and practical reasons. We show how to obtain more information from a given constraint set by computing cutting planes and how to use this information in constraint solving and constrained optimization. Focusing on the pseudo-Boolean case CLP(PB), where all domains are equal to the two-element set $\{0, 1\}$, we present specialized cutting plane techniques and illustrate them on a number of examples.

[4] P. Barth, 1994. Linear 0-1 inequalities and extended clauses. Technical Report MPI-I-94-216, Saarbrücken.

> **Abstract** Extended clauses are the basic formulas of the 0-1 constraint solver for the constraint logic programming language CLP(PB). We present a method for transforming an arbitrary linear 0-1 inequality into a set of extended clauses, such that the solution space remains invariant. After applying well-known linearization techniques on non-linear 0-1 constraints followed by the presented transformation method, we are able to handle arbitrary 0-1 constraints in CLP(PB). The transformation method presented relies on cutting planes techniques known from 0-1 integer programming. We develop specialized redundancy criteria and so produce the minimal number of extended clauses needed for preserving equivalence. The method is enhanced by using a compact representation of linear 0-1 inequalities and extended clauses. Unit resolution for classical clauses is generalized to pseudo-Boolean unit resolution for arbitrary linear 0-1 inequalities. We extend the transformation method to constrained transformation when the inequality to be transformed is part of a larger set of linear 0-1 inequalities. Furthermore the method can be used to obtain all strongest extended cover inequalities of a knapsack inequality.

[5] P. Barth, 1994. *A Short Guide to CLP($\mathcal{PB}$)*, 1994. System available by anonymous ftp from `ftp.mpi-sb.mpg.de` in directory `pub/tools/CLPPB/clppb.tar.Z`.

[6] P. Barth, 1995. A Davis-Putnam based enumeration algorithm for linear pseudo-Boolean optimization. Technical Report MPI-I-95-2-003.

[7] D. Basin, F. Giunchiglia, M. Kaufmann (editors), 1994. Proceedings of the workshop on correctness and metatheoretic extensibility of automated reasoning systems. Technical Report 9405-10, Instituto Per La Ricerca Scientifica E Technologica, Trento. Joint publication with IRST, appeared as IRST technical report.

[8] Y. Dimopoulos, 1994. Classical methods in nonmonotonic reasoning. Technical Report MPI-I-94-229, Max-Planck-Institut für Informatik.

[9] D. Fehrer, U. Hustadt, M. Jaeger, A. Nonnengart, H. J. Ohlbach, R. A. Schmidt, C. Weidenbach, E. Weydert, 1994. Description logics for natural language processing. In *International Workshop on Description Logics '94*, pp. 80–84. DFKI, Bonn, Germany.

> **Abstract** In this paper we focus on the application of description logics to natural language processing. In cooperation with the PRACMA Project (PRACMA is Agents.') we have been developing a suitably extended knowledge representation system, called MOTEL. In our approach to agent modeling and natural language processing we use an extension of the well-known description language $\mathcal{ALC}$. Our system MOTEL serves on one hand as a knowledge base for the natural language front-end, and on the other hand, it provides powerful *logical* representation and reasoning components. As our approach is logic based we hope that this enhances the overall capabilities of the natural language processing (NLP) system. We present a brief

overview of MOTEL and the different extensions we are working on, i.e. modal extension of description logics, a cardinality-based approach to quantitative information, reason maintenance, probabilistic, non-monotonic, and abductive reasoning.

[10] D. Gabbay. Labelled Deductive Systems. Technical Report MPI–I–94-223.

Abstract Traditional logics manipulate formulas. The message of this book is to manipulate pairs; formulas and labels. The labels annotate the formulas. This sounds very simple but it turned out to be a big step, which makes a serious difference, like the difference between using one hand only or allowing for the coordinated use of two hands. Of course the idea has to be made precise, and its advantages and limitations clearly demonstrated. 'Precise' means a good mathematical definition and 'advantages demonstrated' means case studies and applications.

[11] H. Ganzinger, 1994. The Saturate system. Available on the world-wide web under URL http://www.mpi-sb.mpg.de/SATURATE/Saturate.html, 1994.

Abstract The Saturate system is an experimental theorem prover based on saturation. It has originally been developed as an implementation of the superposition calculus by Pilar Nivela and Robert Nieuwenhuis from the Technical University of Catalonia, Barcelona. The version the use of which is described in this document now contains extensions by chaining techniques for arbitrary transitive relations implemented by Harald Ganzinger, MPI Informatik, Saarbrücken, with the help of Robert Nieuwenhuis.

[12] P. Graf, 1994. Substitution tree indexing. Technical Report MPI-I-94-251, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract The performance of a theorem prover crucially depends on the speed of the basic retrieval operations, such as finding terms that are unifiable with (instances of, or more general than) a given query term. In this paper a new indexing method is presented, which outperforms traditional methods such as path indexing, discrimination tree indexing and abstraction trees. Additionally, the new index not only supports term indexing but also provides maintenance and efficient retrieval of substitutions. As confirmed in multiple experiments, substitution trees combine maximal search speed and minimal memory requirements.

[13] M. Hanus, 1994. On extra variables in (equational) logic programming. Technical Report MPI-I-94-246, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Abstract Extra variables in a clause are variables which occur in the body but not in the head. It has been argued that extra variables are necessary and contribute to the expressive power of logic languages. In the first part of this paper, we show that this is not true in general. For this purpose, we provide a simple syntactic transformation of each logic program into a logic program without extra variables. Moreover, we show a strong correspondence between the original and the transformed program with respect to the declarative and the operational semantics. In the second part of this paper, we use a similar technique to provide new completeness results for equational logic programs with extra variables. In equational logic programming it is well known that extra variables cause problems since narrowing, the standard operational semantics for equational logic programming, may become incomplete in the presence of extra variables. Since extra variables are useful from a programming point of view, we characterize new classes of equational logic programs with extra variables for which narrowing and particular narrowing strategies are complete. In particular, we show the completeness of narrowing strategies in the presence of nonterminating functions and extra variables in right-hand sides of rewrite rules.

[14] J. Hopf (editor), 1994. Genetic algorithms within the framework of evolutionary computation. Proceedings of the KI-94 Workshop 241, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken.

[15] M. Jaeger, 1994. A probabilistic extension of terminological logics. Technical Report MPI-I-94-208, Max-Planck-Institut für Informatik.

Abstract In this report we define a probabilistic extension for a basic terminological knowledge representation languages. Two kinds of probabilistic statements are introduced: statements about conditional probabilities between concepts and statements expressing uncertain knowledge about a specific object. The usual model-theoretic semantics for terminological logics are extended to define interpretations for the resulting probabilistic language. It is our main objective to find an adequate modeling of the way the two kinds of probabilistic knowledge are combined in what we call default reasoning about probabilities. Cross entropy minimization is a technique that turns out to be a very promising tool towards achieving this end.

[16] H. J. Ohlbach, 1994. Computer support for the development and investigation of logics. Technical Report MPI-I-94-228.

Abstract Symbolic reasoning in a logical framework becomes more and more important for computer applications such as Natural Language Processing Systems or Expert Systems. These applications usually need specifically tailored logics. Therefore we are developing methods and algorithms for supporting the designer of an application system, who is usually not a logician, to develop his own application oriented logic. This paper gives an overview about our current state of these investigations. In particular we consider the correspondences between axiomatic and semantic specifications of a logic and the problem of finding one from the other. Correlated with this area are translation methods from the object logic into predicate logic, and methods for optimizing the translation. Other topics are investigations of the expressiveness of a logic and the axiomatizability of semantic conditions. The basic techniques underlying our approach, so called K-transformations and quantifier elimination, are briefly discussed. They are quite general mechanisms for manipulating predicate logic formulae, and the investigation of logics is only one of their applications. For the technical details of the methods and the proofs I refer to the original papers.

[17] H. J. Ohlbach, 1994. Synthesizing semantics for extensions of propositional logic. Technical Report MPI-I-94-225.

Abstract Given a Hilbert style specification of a propositional extension of standard propositional logic, it is shown how the basic model theoretic semantics can be obtained from the axioms by syntactic transformations. The transformations are designed in such a way that they eliminate certain derived theorems from the Hilbert axiomatization by turning them into tautologies. The following transformations are considered. Elimination of the reflexivity and transitivity of a binary consequence relation yields the basic possible worlds framework. Elimination of the congruence properties of the connectives yields weak neighbourhood semantics. Elimination of certain monotonicity properties yields a stronger neighbourhood semantics. Elimination of certain closure properties yields relational possible worlds semantics for the connectives. If propositional logic is the basis of the specification, the translated Hilbert axioms can be simplified by eliminating the formula variables with a quantifier elimination algorithm. This way we obtain the frame conditions for the semantic structures. All transformations work for arbitrary n-place connectives. The steps can be fully automated by means of PL1 theorem provers and quantifier elimination algorithms. The meta theory guarantees soundness and completeness of all transformation steps. As a by–product, translations into multi–modal logic are developed.

[18] H. J. Ohlbach, 1995. General representation theorems. Technical Report MPI-I-95-2-006, Max-Planck-Institut für Informatik, Saarbrücken. to appear.

Abstract A general methodology for deriving representation theorems for various algebras and structures is presented. Starting with a very simple representation for a simple structure we can get more sophisticated representation by means of a almost automated bootstrapping method. For example one can start with a representation for partial orderings which maps each element x to the set of upward closed subsets containing x. From this it is possible to derive

in a sequence of steps representations for semilattices with functions, lattices with functions, distributive lattices with functions, Boolean algebras with functions etc. These representations correspond to model theoretic semantics for non–classical logics. Therefore we obtain a method for deriving possible worlds semantics from axiomatically presented logics and for computing the corresponding frame properties.

[19] H. J. Ohlbach, D. Gabbay, D. Plaisted, 1994. Killer transformations. Technical Report MPI-I-94-226, Max-Planck-Institut für Informatik, Saarbrücken, Germany. To be published in *Proc. of the 1993 Workshop on Proof Theory in Modal Logic*, Hamburg.

Abstract This paper deals with methods of faithful transformations between logical systems. Several methods for developing transformations of logical formulae are defined which eliminate unwanted properties from axiom systems without losing theorems. The elementary examples we present are permutation, transitivity, equivalence relation properties of predicates and congruence properties of functions. Various translations between logical systems are shown to be instances of K-transformations, for example the transition from relational to functional translation of modal logic into predicate logic, the transition from axiomatic specifications of logics via unary provability relations to a binary consequence relations, and the development of neighbourhood semantics for nonclassical propositional logics. Furthermore we show how to eliminate self resolving clauses like the condensed detachment clause, resulting in dramatic improvements of the performance of automated theorem provers on extremely hard problems. As by–products we get a method for encoding some axioms in Prolog which normally would generate loops, and we get a method for parallelizing some closure computation algorithms.

[20] H. J. Ohlbach, R. A. Schmidt, January 1995. Functional translation and second-order frame properties of modal logics. Technical Report MPI-I-95-2-002, Max-Planck-Institut für Informatik, Saarbrücken. Submitted for publication to the *Journal of Logic and Computation.*

Abstract Normal modal logics can be defined axiomatically as Hilbert systems, or semantically in terms of Kripke's possible worlds and accessibility relations. Unfortunately there are Hilbert axioms which do not have corresponding first-order properties for the accessibility relation. For these logics the standard semantics-based theorem proving techniques, in particular, the relational translation into first-order predicate logic, do not work. There is an alternative translation, the so-called functional translation, in which the accessibility relations are replaced by certain terms which intuitively can be seen as functions mapping worlds to accessible worlds. In this paper we show that from a certain point of view this functional language is more expressive than the relational language, and that certain second-order frame properties can be mapped to first-order formulae expressed in the functional language. Moreover, we show how these formulae can be computed automatically from the Hilbert axioms. This extends the applicability of the functional translation method.

[21] H. J. Ohlbach (editor), 1994. Temporal logic: Proceedings of the ICTL workshop. Technical Report MPI-I-94-230.

[22] D. A. Plaisted, 1994. An abstract program generation logic. Technical Report MPI-I-94-232.

Abstract We present a system for representing programs as proofs, which combines features of classical and constructive logic. We present the syntax, semantics, and inference rules of the system, and establish soundness and consistency. The system is based on an unspecified underlying logic possessing certain properties. We show how proofs in this system can be systematically converted to programs in a class of abstract logic programming languages including term-rewriting systems and Horn clause logic programs. A number of examples of such logic programming languages and underlying logics are given, as well as some proofs that can be expressed in this system and the corresponding programs.

[23] D. A. Plaisted, 1994. Ordered semantic hyper-linking. Technical Report MPI-I-94-235.

Abstract We propose a method for combining the clause linking theorem proving method with theorem proving methods based on orderings. This may be useful for incorporating term-rewriting based approaches into clause linking. In this way, some of the propositional inefficiencies of ordering-based approaches may be overcome, while at the same time incorporating the advantages of ordering methods into clause linking. The combination also provides a natural way to combine resolution on non-ground clauses, with the clause linking method, which is essentially a ground method. We describe the method, prove completeness, and show that the enumeration part of clause linking with semantics can be reduced to polynomial time in certain cases. We analyze the complexity of the proposed method, and also give some plausibility arguments concerning its expected performance.

[24] S. Vorobyov, 1995. Proof normalization and subject reduction in extensions of $F_{\leq}$. Technical Report MPI–I–95–2–001.

Abstract System Fsub, the second-order polymorphic typed lambda-calculus with subtyping, appeared to be undecidable because of the undecidability of its subtyping component. The discovery of decidable extensions of the Fsub-subtyping relation put forward a challenging problem of incorporating these extensions into an Fsub-like typing in a decidable and coherent manner. In this paper we describe a family of systems combining the standard Fsub-typing rules with converging hierarchies of decidable extensions of the Fsub-subtyping and give decidable criteria for successful proof normalization and subject reduction.

[25] C. Weidenbach, 1994. Minimal resolution. Technical Report MPI-I-94-227.

Abstract Minimal resolution restricts the applicability of resolution and factorization to minimal literals. Minimality is an abstract criterion. It is shown that if the minimality criterion satisfies certain properties minimal resolution is sound and complete. Hyper resolution, ordered resolution and lock resolution are known instances of minimal resolution. We also introduce new instances of the general completeness result, correct some mistakes in existing literature and give some general redundancy criteria for minimal resolution calculi.

Part IV

Appendix

1 Present technical configuration

1.1 Technical facilities

The Max-Planck-Institut für Informatik is equipped with 5 servers, approx. 120 workstations (mainly Sun workstations) 2 graphics workstations and some PCs. These facilities are divided into three systems: one system for research group 1, a second system for research group 2 and another system for the service groups (i.e. administration, management, library and computer maintenance).
Network and ftp service (name service, mail etc.) is provided centrally by a Solbourne server 5E/906 that is equipped with 256 MByte main memory, three IPI-2-controllers with 18-GByte-IPI-disks and 10-GByte-SCSI-disks. A central file server cluster (Sun network cluster) with approx. 50 GBytes is responsible for the central data storage/processing.
The division of the three networks takes place via a CISCO box (AGS+).

Further facilities are group-specific:

- **Facilities of research group 1:**

  Research associates of group 1 use about 50 Sun workstations (SLC's, ELC's, Sparc2 and since recently Sparc10 models) as well as two graphics workstations (Silicon Graphics Crimson and Indigo2).
  A Sparc20 Server (two processors and 256 MByte main memory) can be used as an additional compute server.

- **Facilities of research group 2:**

  Similarily, research associates of group 2 use about 50 Sun workstations (SLC's, ELC's, LX, Voyagers and Sparc10 models), two Macintoshs, a PC and ten X terminals. A Solbourne station with 288 MByte main memory and four new 6-series-processors is used as a compute server. In particular cases this server is also used throughout the institute. Also a Sparc20 and a Sparc10 is used as additional compute servers.

- **Facilities of the service groups:**

  The network of the service groups comprises 20 workstations (Sun workstations and applications) that are combined to a Novell network.

Thus - apart from a few exceptions - a homogene SPARC environment is provided. Two Silicon Graphics mono-power-challenge computers for special floating point applications are planned.

1.2 Administration

There is a close cooperation between our institute and other institutions on the university campus such as the computer science department (Fachbereich Informatik, FBI) of the Universität des Saarlandes and the Deutsche Forschungsinstitut für Künstliche Intelligenz (DFKI). The MPI is connected to these institutions via FDDI (the campus backbone). The technical configuration of the university computer science department and many other leading computer science departments (worldwide) is very similar to our configuration. There are two positive consequences: The computer maintenance groups of the MPI, the FBI, the DFKI and the

Rechenzentrum of the Universität des Saarlandes support each other with reference to maintenance, operation, spare parts etc. This helps to save material and staff costs.
The working environment at the MPI and other institutions is very similar and this makes it easy for visitors to share resources.

1.3 External data communication

The MPI and the Rechenzentrum of the Universität des Saarlandes maintain the infrastructure to external data communication.

- National transmission takes place via the Deutsche Wissenschaftsnetz (WIN) at a rate of 2MBit/s.
- International transmission takes place via an additional fixed-point connection (64 kBit/s) to our IP-service-provider XLINK in Karlsruhe. Karlsruhe is linked to the EBONE and the U.S. at a rate of 2MBit/s.
  Note: The 64 kBits link is a bottleneck and will be increased by 2 MBit/s in the next couple of days.

1.4 Future infrastructural steps

The new MPI building will have a communication infrastructure of virtual LANs based on ATM technology, if the products are available in time. We expect to improve performance and scaling.