

PROGRESS REPORT

1991 – 1994

April 1994



I N F O R M A T I K

Max-Planck-Institut für Informatik

Im Stadtwald

66123 Saarbrücken

Germany

Bibliothek  Nr. 6388 2 Ex

Contents

I	The Algorithms and Complexity Group	3
1	Personnel	5
2	Executive Summary	6
3	Research Themes	8
3.1	Parallel Computing	8
3.1.1	Parallel Sorting	8
3.1.2	Superfast Parallel Algorithms	10
3.1.3	Dynamic Load Balancing	13
3.1.4	Routing and Sorting in Meshes	15
3.1.5	Parallel Algorithms for Geometric Problems on Network Machines	19
3.1.6	Lower Bounds for Parallel Algorithms	21
3.1.7	A Scalar-Product Circuit	24
3.2	Computational Geometry	26
3.2.1	Proximity Problems	26
3.2.2	Randomized Incremental Constructions (RICs)	28
3.2.3	Abstract Voronoi Diagrams	30
3.2.4	Motion Planning	32
3.2.5	Approximate Decision Algorithms for Geometric Pattern Matching	35
3.3	Data Structures and Graph Algorithms	37
3.3.1	Competitive Analysis of On-Line Algorithms	37
3.3.2	Network Flow Algorithms	41
3.3.3	Data Structures for Sets and Sequences	43
3.3.4	Bottom-Up-Heapsort	46
3.4	Realization and the LEDA Project	47
3.4.1	The LEDA Platform for Combinatorial and Geometric Computing	47
3.4.2	Precision and Degeneracy in Geometric Computations	50
3.5	Complexity Theory	53
3.5.1	Circuit Complexity	53
3.5.2	Communication Complexity	55
3.5.3	Symbolic Algebra	57
4	Dissertations	58
5	Visitors	60
6	Teaching Activities	64
7	Organization	64

8	Cooperations	65
8.1	BMFT-projects	65
8.2	SFB 124 VLSI-Entwurfsmethoden und Parallelität	65
8.3	ALCOM	65
8.4	ALTEC	66
8.5	HC & M	66
8.6	GIF	66
8.7	Industry	66
9	Technical Reports	67
 II The Programming Logics Group		 77
1	Members of the working group	79
2	Research programme and results	81
2.1	Introduction	81
2.2	Integrating functional and logic languages	82
2.3	Constraint logic programming and combinatorial optimization	84
2.4	Analysis of declarative programs	85
2.5	Automated deduction	87
2.5.1	Background	87
2.5.2	Our results	89
2.6	Program synthesis	92
2.7	Transformation of Logical Systems	93
2.8	Nonmonotonic reasoning	95
2.9	Logical frameworks	97
2.10	Knowledge representation using non-classical logics	99
3	Journal and conference activities	102
3.1	Editorial positions	102
3.2	Conference positions	102
3.2.1	Memberships in Organizing Committees	102
3.2.2	Memberships in Program Committees	102
4	Teaching activities	104
4.1	Lectures and Seminars	104
4.2	Doctorates awarded	105
4.3	Habilitations	105
4.4	Masters theses in progress	105
5	Grants	106
5.1	CCL - Construction of Computational Logics	106
5.2	COMPASS - A Comprehensive Algebraic Approach to System Specification and Development	107

5.3	MEDLAR II – MEchanizing Deduction in the Logics of prActical Reasoning	109
5.4	Detecting Redundancy of Clauses and Inferences	110
5.5	LOGO – Logic Engineering	111
5.6	Automation of Proof by Mathematical Induction	112
5.7	ACCLAIM – Advanced Concurrent Constraint Languages: Application, Implementation, and Methodology	113
5.8	MInd IndUS Collaboration on Proof by Mathematical Induction	114
5.9	PROCOPE – Construction of Non-Classical Logics	115
5.10	EDDS – Efficient Data Structures for Deduction Systems	115
5.11	SOFTI II – Logic of Programming	117

6	Recent Publications	118
----------	----------------------------	------------

The "Max-Planck-Institut für Informatik"

The *Max-Planck-Institut für Informatik* was founded in November 1988 and opened on Dec. 1st, 1990.

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 are to be processed, sometimes in real time. In such a situation efficient algorithms and data structures as well as the exploitation of parallelism 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. It is still an unsolved problem of how to actually build such machines.

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

Today's computer systems are complex in 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 better understand the nature of such systems, how to develop them such that they behave predictably and as wanted and that they are by construction insensitive to certain faults of 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 aforementioned forms of complexity.

Structure

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

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

So far two research groups have taken up their work. The algorithms and complexity group, headed by K. Mehlhorn started on Dec. 1st, 1990. On Jan. 1st, 1991 H. Ganzinger began to build up his programming logics group. At present 16 research associates, 26 doctorate students and 11 postdocs are affiliated with the institute. The scientific staff is complemented by an administrative unit (11 persons, including secretaries), by a computing support unit (5 persons) and by our library staff (2 persons). The computing support unit currently operates a network of approximately 100 workstations.

The *Max-Planck-Gesellschaft* also supports basic research at universities in the former GDR.

About 30 research groups are funded by the *Max-Planck-Gesellschaft* at present. One of these is associated with our institute. It is led by Prof. Gössel and investigates issues in fault tolerant computing.

For 1994 we expect to be able to expand the institute by one or two more research groups. In full operation the institute will accommodate about 200 researchers, including doctoral students and postdocs.

Grants

The institute carries out a number of projects related to research grants awarded by the European Community, by the German Science Foundation (DFG) and by the German Ministry of Research and Technology (BMFT); for the descriptions of these grants see sections I.8 and II.5.

Results

The following two chapters describe in detail, for each of the two research groups, research programme and results obtained since 1991. These results are disseminated mainly through scientific publications, including about 150 articles in journals, books or proceedings of major international conferences, and through computer programs such as the LEDA library of efficient algorithms, the ACID collection for term indexing data structures and the SATURATE experimental theorem prover.

The institute makes an effort to offering a variety of courses to computer science students of the "Universität des Saarlandes". During the current semester (Winter 93/94), about 28% of all courses in computer science are taught by members of the institute.

During the past three years 10 doctoral dissertations and 3 "Habilitationen" have been successfully completed.

Members of the institute have been involved in the organization of 6 workshops and conferences. In 25 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 9 scientific journals.

These figures are a clear indication for the success, the visibility and the outside appreciation of the institute's work in our community.

Part I

The Algorithms and Complexity Group

1 Personnel

As of December 1st, 1993, the group consists of

Director:

Prof. Kurt Mehlhorn

Senior Researcher:

Privatdozent Dr. Torben Hagerup

Dr. Stefan Näher

Dr. Christine Rüb

Dr. Michiel Smid

Researcher:

Dr. Susanne Albers

Dr. Rudolf Fleischer

Dr. Michael Müller

Dr. Stefan Schirra

Dr. Jop Sibeyn

Postdocs:

Dr. Srinivasa Arikati

Dr. Sunil Arya

Dr. Greg Barnes

Dr. Shiva Chaudhuri

Dr. Devdatt Dubhashi

Dr. Vince Grolmusz

Graduate students:

Christoph Burnikel

Gerhard Klär

Thomas Lauer

Hans-Peter Lenhof

Markus Paul

Volker Priebe

Ronald Rasch

Thomas Schilz

Erik Schwarzenecker

Christoph Storb

Christian Thiel

Christian Uhrig

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. The outcome of our theoretical work is publications and the outcome of our practical work is publications and software, e.g., the LEDA platform for combinatorial and geometric computing.

More specifically, our research concentrates on Computational Geometry, Parallel Algorithms, Data Structures and Graph Algorithms, Computational Complexity, and Implementation of Algorithms. In each of the five areas we have achieved significant results over the past three years. We give some highlights now and refer the reader to section 3 for a detailed discussion of our results.

Parallel Algorithms (coordinators: Torben Hagerup and Christine Rüb): For both the PRAM and the mesh model of parallel computation we have developed very efficient algorithms for such basic tasks as sorting, merging, routing, and load balancing. Some of the algorithms have even been shown to be optimal. We have also contributed decisively to the development of superfast (\equiv sublogarithmic running time) PRAM algorithms, sometimes called the \log^* -revolution. More on the applied side we have developed parallel algorithms for some geometric problems, e.g., the convex hull and the triangulation problem, whose running time on existing parallel machines, e.g., the INTEL Hypercube, scales almost linearly. We have also designed an integrated circuit for the high-precision scalar product required for Kulisch-arithmic.

Computational Geometry (coordinators: Kurt Mehlhorn and Michiel Smid): We have done an (almost) definite study of proximity problems in the plane and in higher dimensional space and we have further investigated the unifying conceptual and algorithmic role of Abstract Voronoi diagrams. For robot motion planning and for geometric pattern matching we have developed approximation algorithms which bring the problems in these areas closer to practically useful solutions. We have also contributed to randomized incremental constructions, one of the most useful algorithmic paradigms in computational geometry.

Data Structures and Graph Algorithms (coordinators: Torben Hagerup, Kurt Mehlhorn, and Stefan Näher): We have developed new algorithms for the network flow problem and new data structures for problems on sets and sequences. We have also investigated the influence of lookahead in on-line algorithms and settled an old conjecture concerning heap-sort.

Realization and the LEDA-project (coordinators: Kurt Mehlhorn and Stefan Näher): We have developed the LEDA platform for combinatorial and geometric computing. It is used by several hundred academic and industrial groups worldwide as the basis of their algorithm development. Our experience with the implementation of geometric algorithms has spurred an investigation of precision and degeneracy issues.

Complexity theory: We have concentrated on circuit complexity and its relation to communication complexity.

Our five research areas are heavily intertwined; in fact, most of us contributed to at least two areas.

M. Kaufmann and T. Hagerup have completed their Habilitation in 1992 and 1993, respectively. M. Kaufmann has since then become Associate Professor of Computer Science at the University of Tübingen. The Habilitation procedure of Stefan Näher is ongoing. Ten graduate students have completed their Ph.D. work within the last three years and graduate students are currently working in our group. Section 4 surveys the topics of their work and the actual or expected completion dates.

The group contributes to the master's program in Computer Science at the Universität des Saarlandes. 35 master's students have written their theses under our supervision in the last three years, and in the winter term 1993/94 we offer 8 courses and seminars. Section 6 gives more details.

The group is involved in seven national and international research projects and cooperates with two industrial partners. Section 8 gives details.

3 Research Themes

3.1 Parallel Computing

3.1.1 Parallel Sorting

Investigator: Torben Hagerup

Sorting is one of the most important and well-studied problems in sequential computation. Its role in parallel computation is perhaps even more dominant. It is therefore not surprising that parallel sorting has been thoroughly investigated. Our group has participated intensively in this effort and made substantial contributions.

The classical setting for the study of sorting is the *comparison* model, in which information about the input elements can be obtained only through pairwise comparisons. It is well-known that the sequential complexity of sorting n elements in this model is $\Theta(n \log n)$. A celebrated result by Ajtai *et al.* [1] states that n elements can be sorted in $O(\log n)$ time on a parallel machine with n processors. In view of the sequential lower bound, the algorithm of Ajtai *et al.* makes optimal use of the available processors and essentially settles the question for the case of at most n processors. Employing more than n processors to sort n elements, we can hope to sort faster than in $\Theta(\log n)$ time on the CRCW PRAM. Two other lower bounds, however, quickly become relevant. One, by Beame and Hastad [4], says that we cannot compute the parity of n bits faster than in $\Theta(\log n / \log \log n)$ time unless we use more than a polynomial number of processors (which we will consider unfeasible); by implication, we cannot sort any faster. For some applications of sorting this lower bound can be circumvented by resorting to so-called *padded sorting*, where the output array is allowed to contain slightly more than n cells, cells not holding an input element being marked with a special *null* value; after this modification, computing the parity no longer reduces to sorting, which renders the lower bound of Beame and Hastad irrelevant. The second lower bound [2, 3, 5] states that even if we only count comparisons (in which case there is no difference between padded sorting and standard sorting), we still cannot sort faster than in $\Theta(\log n / \log k)$ time with kn processors, for $k \geq 2$. We have worked towards the ultimate goal for padded sorting set by this very general lower bound. In the randomized case, we reached the goal completely by giving an algorithm with expected running time $O(\log n / \log k)$ [13], in the deterministic case we came close [14]. These results have found applications in string sorting (see below) and in the construction of Voronoi diagrams [15].

Although the algorithm of Ajtai *et al.* [1] is optimal in the comparison-based setting, this is not necessarily so if the elements to be sorted allow additional operations besides comparisons. E.g., small integers can be sorted in linear time sequentially, so that we could hope to sort n integers in $O(\log n)$ time with just $O(n / \log n)$ processors. For very small integers, this is indeed possible [7]. For integers of about the same size as n ,

the most interesting case, the goal has never been reached, but we have given the best algorithms known for the PRAM model: For the EREW and CREW PRAMs in [7], for the deterministic CRCW PRAM in [9], and for the randomized CRCW PRAM in [8]. For the related problem of integer merging, we showed in [11], quite surprisingly, that two sorted sequences of n sufficiently small integers can be merged in $o(\log n)$ time on the very weak EREW PRAM, for which close to no sublogarithmic-time algorithms are known.

The recent surge of interest in computational biology has revitalized the area of string processing, the strings of interest being chiefly those encoded by molecules such as DNA. We have studied parallel sorting and merging of strings of characters, equipped with the usual lexicographical ordering, the assumption being that two characters, but not two strings, can be compared in constant time [12, 10]. Our most recent work gives, in particular, a complete characterization of the complexity of string merging on the EREW PRAM.

We plan to continue work in all of the directions described above: padded sorting, integer sorting, and string sorting. In addition, we would like to discover simpler versions of the known efficient algorithms for standard parallel sorting [1, 6].

References

- [1] M. Ajtai, J. Komlós, E. Szemerédi *An $O(n \log n)$ sorting network*. Proceedings 15th Ann. ACM Symp. on Theory of Computing (STOC), 1983, 1–9
- [2] N. Alon, Y. Azar *The average complexity of deterministic and randomized parallel comparison-sorting algorithms*. SIAM J. Comput., Vol. 17, 1988, 1178–1192
- [3] Y. Azar, U. Vishkin *Tight comparison bounds on the complexity of parallel sorting*. SIAM J. Comput., Vol. 16, 1987, 458–464
- [4] P. Beame, J. Hastad *Optimal bounds for decision problems on the CRCW PRAM*. J. ACM, Vol. 36, 1989, 643–670
- [5] R. B. Boppana *The average-case parallel complexity of sorting*. Inform. Process. Lett., Vol. 33, 1989, 145–146
- [6] R. Cole *Parallel merge sort*. SIAM Journal Comput., Vol. 17, 1988, 770–785

Work of our group:

- [7] S. Albers, T. Hagerup *Improved Parallel Integer Sorting without Concurrent Writing*. Proceedings 3rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1992, 463–472
- [8] H. Bast, T. Hagerup *Fast Parallel Space Allocation, Estimation and Integer Sorting (revised)*. Technical Report MPI-I-93-123
- [9] P.C.P. Bhatt, K. Diks, T. Hagerup, V. C. Prasad, T. Radzik, S. Saxena *Improved deterministic parallel integer sorting*. Inform. and Comput., Vol. 94, 1991, 29–47
- [10] T. Hagerup *Optimal Parallel String Algorithms: Sorting, Merging and Computing the Minimum*. Technical Report MPI-I-93-152
- [11] T. Hagerup, M. Kutylowski *Fast Integer Merging on the EREW PRAM*. Proceedings 19th International Colloquium on Automata, Languages and Programming (ICALP), Vol. 623, 1992, 318–329
- [12] T. Hagerup, O. Petersson *Merging and Sorting Strings in Parallel*. Proceedings 17th Symposium on Mathematical Foundations of Computer Science, Springer Lecture Notes in Computer Science, Vol. 629, 1992, 298–306
- [13] T. Hagerup, R. Raman *Waste Makes Haste: Tight Bounds for Loose Parallel Sorting*. Proceedings 33rd Annual Symposium on Foundations of Computer Science (FOCS), 1992, 628–637
- [14] T. Hagerup, R. Raman *Fast Deterministic Approximate and Exact Parallel Sorting*. Proceedings 5th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), 1993, 346–355
- [15] S. Sen *Tight bounds for some problems in computational geometry: The complete sub-logarithmic parallel time range*. Technical Report MPI-I-93-129

3.1.2 Superfast Parallel Algorithms

Investigators: Holger Bast, Shiva Chaudhuri and Torben Hagerup

The *compaction* problem of size n is as follows: Given an array A of size n , k of whose cells contain an object, place the objects in A in an array of size exactly k ; the number k , in general, is not provided as part of the input. In the PRAM model of computation,

compaction is intimately related to *processor allocation*, variously known as *load balancing* or *processor scheduling*, and as such is one of the most fundamental concerns of efficient algorithms. Compaction problems of size n can be solved in $O(\log n / \log \log n)$ time with optimal speedup by means of prefix summation [2]. On the other hand, a lower bound by Beame and Hastad [1] precludes the existence of faster efficient algorithms; this, in effect, became the limiting factor of many fast parallel algorithms. It came as a surprise when in 1991 Matias and Vishkin [5] demonstrated that *approximate* compaction problems of size n can be solved in $O(\log^* n)$ expected time by an n -processor randomized algorithm. This heralded a host of fast algorithms for problems that previously were thought not to have fast solutions. We have contributed many of these algorithms, which we term “superfast algorithms”.

In [7] we showed that so-called static perfect hashing problems of size n can be solved in $O(\log^* n)$ expected time with optimal speedup. The task here is, given n (presumably large) integers, to map these injectively to a range of size $O(n)$ (by means of a hash function). Gil *et al.* [3] later showed that a dynamic dictionary data structure with $O(\log^* n)$ expected update time and constant lookup time can be derived from any such result. [3] and [4] gave other examples of superfast randomized algorithms. The main result in [6] is that so-called *semisorting* problems of size n can be solved in $O(\log^* n)$ expected time with optimal speedup. Semisorting takes as input n records with integer keys and is like sorting in that it groups together all records with a common key; it differs from sorting in that the different key values may not occur in increasing order. Our semisorting result has numerous applications: It allows a significant simplification of the hashing result of [7] (this still needs to be worked out), it is an essential ingredient in the padded-sorting scheme of [12] (see Section 3.1.1), we used it in a very fast algorithm for computing the Voronoi diagram of random sites in the plane [11], and it leads to simpler simulations between different variants of the CRCW PRAM (again, this is planned work). The results mentioned above all concern randomized algorithms. Until recently no deterministic superfast algorithm for even the basic approximate compaction problem was known. In [10] we provided such an algorithm, with a running time of $O((\log \log n)^3)$. In [8] we showed that no deterministic algorithm can solve approximate compaction problems of size n faster than in time $\Theta(\log \log n)$, which implies that “superfast” deterministic algorithms are, of necessity, slower than their randomized counterparts. Although applications of deterministic approximate compaction are less immediate than those of randomized approximate compaction, we subsequently derived fast deterministic algorithms for padded sorting [13] (see Section 3.1.1) and approximate and exact selection [9]. Work in progress aims at discovering additional applications of our results.

References

- [1] P. Beame, J. Hastad *Optimal bounds for decision problems on the CRCW PRAM.*

J. ACM, Vol. 36, 1989, 643–670

- [2] R. Cole, U. Vishkin *Faster Optimal Parallel Prefix Sums and List Ranking* Inform. and Comput., Vol. 81, 334–352
- [3] J. Gil, Y. Matias, U. Vishkin *Towards a Theory of Nearly Constant Time Parallel Algorithms* Proceedings 32nd Annual Symposium on Foundations of Computer Science (FOCS), 698–710
- [4] M.T. Goodrich *Using Approximation Algorithms to Design Parallel Algorithms that May Ignore Processor Allocation* Proceedings 32nd Annual Symposium on Foundations of Computer Science (FOCS), 1991, 711–722
- [5] Y. Matias, U. Vishkin *Converting High Probability into Nearly-Constant Time - with Applications to Parallel Hashing* Proceeding 23rd Annual ACM Symposium on Theory of Computing (STOC), 1991, 307–316

Work of our group:

- [6] H. Bast, T. Hagerup *Fast Parallel Space Allocation, Estimation and Integer Sorting (revised)* Technical Report MPI-I-93-123
- [7] H. Bast, T. Hagerup *Fast and Reliable Parallel Hashing* Proceedings 3rd Annual ACM-SIAM Symposium on Parallel Algorithms and Architectures (SPAA), 1991, 50–61
- [8] S. Chaudhuri *A Lower Bound for Linear Approximate Compaction* Proceedings of 2nd Israel Symposium on Theory of Comput. and Sys. (ISTCS), 1993, 25–32
- [9] S. Chaudhuri, T. Hagerup, R. Raman *Approximate and Exact Deterministic Parallel Selection* Proceedings 18th Math. Fdtns. of Comp. Sci., Springer LNCS, Vol. 711, 1993, 352–361
- [10] T. Hagerup *Fast Deterministic Processor Allocation* Proceedings 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1993, 1–10
- [11] T. Hagerup, J. Katajainen *Improved Parallel Bucketing Algorithms for Proximity Problems* Proceedings 26th Hawaii International Conference on System Sciences, Vol. 2: Software Technology, 1993, 318–327
- [12] T. Hagerup, R. Raman *Waste Makes Haste: Tight Bounds for Loose Parallel Sorting* Proceedings 33rd Annual Symposium on Foundations of Computer Science (FOCS), 1992, 628–637

- [13] T. Hagerup, R. Raman *Fast Deterministic Approximate and Exact Parallel Sorting* Proceedings 5th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), 1993, 346–355

3.1.3 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 the Siemens AG (division: ZFE BT SE 42). The goal is to find dynamic load balancing algorithms that perform well for large systems (asymptotic results), but 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 do load distribution by a static function mapping the *load packages* to the processors; for examples, see [BI85, MS90]. 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 [KZ88] and [LNRS89].

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 a *parent problem* often creates many related *child problems* that can be solved more efficiently if they are located in 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 [RSU91]. That paper gives an algorithm where a processor initiates a balancing step in dependence of 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 [LMRT91, LM92] and [LMRT92], Lüling and Monien proposed another dynamic load balancing scheme in [LM93]. 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 [LM93] and gave a high-probability analysis. So far we can 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.

We will next try to extend the algorithm further to guarantee that waves are only executed when they are necessary. This is important because we are interested in load *balancing*, and a periodically activated wave conflicts with the idea that a load balancing step is activated only when the load differs heavily from the average.

References

- [BI85] S. Bhatt, I. Ipsen: *How to Embed Trees in Hypercubes*. Yale University Research Report YALEU/DCS/RR-443 1985
- [KZ88] R. Karp, Y. Zhang: *A Randomized Parallel Branch-and-Bound Procedure*. Proceedings of the 20th Annual ACM Symposium on Theory of Computing 1988, pp 290-300
- [LNRS89] T. Leighton, M. Newman, A.G. Ranade, E. Schwabe: *Dynamic Tree Embeddings in Butterflies and Hypercubes* Proc. of the 1st ACM Symp. on Parallel Algorithms and Architectures 1989, pp 224-234
- [MS90] B. Monien, I. H. Sudborough: *Embedding one Interconnection Network in Another* Computing Supp. 7, pp 257-282

- [RSU91] L. Rudolph, M. Slivkin-Allalouf, E. Upfal: *A Simple Load Balancing Scheme for Task Allocation in Parallel Machines* Proc. of the 1991 ACM Symp. on Parallel Algorithms and Architectures 1991, pp 237-245
- [LMRT91] R. Lüling, B. Monien, F. Ramme: *Load Balancing in Large Networks: A Comparative Study* 3rd IEEE Symposium on Parallel and Distributed Processing 1991, pp. 686-689
- [LMRT92] R. Lüling, B. Monien, M. Räcke, S. Tschöke: *Efficient Parallelization of a Branch & Bound Algorithm for the Symmetric Traveling Salesman Problem* European Workshop on Parallel Computing 1992
- [LM92] R. Lüling, B. Monien: *Load Balancing for Distributed Branch & Bound Algorithms* Proceedings of Int. Parallel Processing Symposium 1992, pp 543-549
- [LM93] R. Lüling, B. Monien: *A Dynamic Load Balancing Algorithm with Provable Good Performance* Proc. of the 5. ACM Symp. on Parallel Algorithms and Architectures 1993, pp 164-172

3.1.4 Routing and Sorting in Meshes

Investigator: Jop F. Sibeyn

Outline of the Problems

One of the main problems in the simulation of idealized parallel computers by realistic ones is that of message routing through the sparse network of links connecting a set of processing units (PUs) among each other. In our research 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. There may also be additional *wrap-around connections* between the two PUs at opposite ends of each row and each column of the network; this type of mesh is called *torus*. A mesh without wrap-around connections is referred to as *square*. We assume the *MIMD model*, where in a single *step* each PU can perform an arbitrary amount of internal computation and transmit one packet of information (of bounded length) to each of its neighbors.

Most research has focused on the 1-1 routing problem, also called the *permutation routing problem*, in which each node is the origin and destination of at most one packet. However, in many practical applications a PU may have to communicate with a number of other PUs at the same time. This motivates the definition of the k - k routing problem, in which each PU can be the source and destination of up to k packets. Other variants are the 1- k routing problem and the general l - k routing problem.

Another fundamental problem that involves the rearranging of packets within a processor network is the *sorting problem*. Again, several variants of the problem have been studied. In the 1-1 sorting problem, each PU initially holds a single packet, where each packet contains a key drawn from a totally ordered set. The packets have to be rearranged such that the packet with the key of rank i is moved to the PU with index i , for all i . In the k - k sorting problem, each PU is the source and destination of k packets.

There are several variants of the above problems. Often considered are meshes of dimension $d > 2$. It turns out that k - k problems for sufficiently large k can be generalized without problem to higher-dimensional meshes, but for small k , particularly for $k = 1$, there are no optimal algorithms known for $d > 2$. Routing and sorting on one-dimensional meshes is important as subroutine for problems on higher-dimensional meshes. In addition, routing and sorting on a one-dimensional mesh with a wrap-around connection, a *ring*, is non-trivial and gives many interesting lower bounds and algorithms.

One may also assume that in addition to the connections with the neighbors there are buses by which fast communication over longer distances is enabled. E.g., there may be one such bus connected to all PUs in each row and one bus connected to all PUs in each column of the mesh. The restriction is that at most one packet may be transferred over a bus at a time.

Routing on One-Dimensional Arrays

On the linear array without wrap-around connections there are trivial algorithms with optimal performance. Makedon and Simvonis were among the first to consider the k - k routing problem on a ring [7]. Independently Kaufmann and Sibeyn developed an algorithm [14] which is faster and more general. In [18] Sibeyn analyses the sorting problem. Among other things, it is established in that paper that sometimes sorting is harder than routing by a factor: 1-1 sorting without making copies has a lower bound of $2/3 \cdot n$ steps, whereas the 1-1 routing problem can be solved trivially in $n/2$ steps. The algorithms developed there resulted in new ideas which led to near-optimal deterministic routing and sorting algorithms for higher-dimensional meshes presented in [15].

Permutation Routing on Two-Dimensional Meshes

The first routing algorithms which came close to the lower bound of $2 \cdot n - 2$ steps were given by Kunde [2] and Rajasekaran and Tsantilas [10]. Leighton, Makedon and Tollis [5] presented the first deterministic algorithm with optimal routing time and constant size queues. The latter paper is of great theoretical importance but the maximal queue size is still impractically large. Rajasekaran and Overholt [9] reduced the queue size Q to less than 200. In [11] Chlebus, Kaufmann and Sibeyn give a further considerable reduction

of Q . One of the algorithms has optimal routing time, $T = 2 \cdot n - 2$ with $Q = 48$, another algorithm has much smaller queues, $Q = 16$, but $T = 2 \cdot n + \mathcal{O}(1)$.

***k-k* Routing and Sorting**

In [3], Kunde showed that *k-k* routing and sorting on squares can be performed in $k \cdot n + o(k \cdot n)$ steps with a queue size of k . A randomized algorithm for *k-k* routing with running time $\max\{4 \cdot n, k \cdot n/2\} + o(k \cdot n)$ was presented by Kaufmann, Rajasekaran and Sibeyn in [12]. Similar results for tori were given in [13]. These algorithms were improved and extended to randomized sorting on squares and tori by using an idea of Reif and Valiant and of Reischuk to randomly select a set of *splitters*. After sorting these splitters, the packets can estimate their rank and determine a corresponding preliminary destination. Then Kunde [4], and independently Kaufmann, Sibeyn and Suel showed that the bounds mentioned can also be achieved deterministically. Both algorithms can easily be generalized to higher-dimensional meshes, but there the latter algorithm performs considerably better.

1-*k* Routing

1-*k* routing reflects practical purposes better than the routing of permutations: if the PUs are working independently of each other and generate packets that have to be transferred to other PUs, then it is unrealistic to assume that every PU is the destination of at most one packet. The parameter k , $1 \leq k \leq n^2$, need not to be known by the PUs, but is needed for stating the complexity of the problem. The 1-*k* routing problem has also nice applications in the context of hot-potato worm-hole routing. *Hot-potato routing* is a routing paradigm in which packets may never be queued at a PU but have to keep moving at all times until they reach their destination [16, 1]. This model is used in many practical systems. In a recent paper of Newman and Schuster [8] it is demonstrated that under a mild condition any efficient 1-*k* routing algorithm with working queue size (the number of packets that already moved and did not yet reach their destination) at most four is useful as a subroutine for the hot-potato worm-hole routing problem. In [19] Sibeyn and Kaufmann present a near-optimal deterministic algorithm running in $\sqrt{k} \cdot n/2 + \mathcal{O}(n)$ steps. An alternative algorithm has slightly worse routing time but working queue size three.

Routing on Meshes with Buses

Meshes with buses have been considered by Leung and Shende [6] who conjectured that n steps would be a lower bound for permutation routing. In [17] Sibeyn, Kaufmann

and Raman show that this is not true by giving an algorithm which requires $0.78 \cdot n$ steps. This comes close to improved lower bounds. The algorithm and lower bounds are generalized to higher dimensions: for permutations the routing time remains below $2 \cdot n$ for all dimensions $d = o(n^{1/3})$.

References

- [1] Feige, U., P. Raghavan, 'Exact Analysis of Hot-Potato Routing,' *Proc. 33rd Symp. on Foundations of Computer Science*, pp. 553-562, IEEE, 1992.
- [2] Kunde, M., 'Routing and Sorting on Mesh Connected Processor Arrays,' *Proc. VLSI Algorithms and Architectures*, Lecture Notes in Computer Science, 319, pp. 423-433, Springer-Verlag, 1988.
- [3] Kunde, M., 'Concentrated Regular Data Streams on Grids: Sorting and Routing Near to the Bisection Bound,' *Proc 32nd Symposium on Foundations of Computer Science*, pp. 141-150, IEEE, 1991.
- [4] Kunde, M., 'Block Gossiping on Grids and Tori: Deterministic Sorting and Routing Match the Bisection Bound,' *Proc. European Symp. on Algorithms*, LNCS 726, pp. 272-283, Springer-Verlag, 1993.
- [5] Leighton, T., F. Makedon, Y. Tollis, 'A $2n - 2$ Step Algorithm for Routing in an $n \times n$ Array with Constant Size Queues,' *Proc. Symposium on Parallel Algorithms and Architectures*, pp. 328-335, ACM, 1989.
- [6] Leung, J., S.M. Shende, 'On Multi-Dimensional Packet Routing for Meshes with Buses,' In *Proc. 3rd IEEE SPDP*, pp. 834-837, 1991. *J. of Parl. and Dist. Comp.*, to appear.
- [7] Makedon, F., A. Simvonis, 'Multipacket Routing on Rings,' *Proc. 1st Intern. ACPC Conference*, LNCS 591, pp. 226-237, Springer-Verlag, 1991.
- [8] Newman, I., A. Schuster, 'Hot-Potato Worm Routing is almost as easy as Store-and-Forward Packet Routing,' *Proc. ISTCS*, 1993.
- [9] Rajasekaran, S., R. Overholt, 'Constant Queue Routing on a Mesh,' *Journal of Parallel and Distributed Computing*, pp. 160-166, June 1992.
- [10] Rajasekaran, S., Th. Tsantilas, 'Optimal Routing Algorithms for Mesh-Connected Processor Arrays,' *Algorithmica*, 8, pp. 21-38, 1992.

Work of our group:

- [11] Chlebus, B.S., M. Kaufmann, J.F. Sibeyn, 'Deterministic Permutation Routing on Meshes,' *Proc. 5th Symp. on Parallel and Distributed Processing*, IEEE, 1993.
- [12] Kaufmann, M., S. Rajasekaran, J.F. Sibeyn, 'Matching the Bisection Bound for Routing and Sorting on the Mesh,' *Proc. 4th Symp. on Parallel Algorithms and Architectures*, pp. 31-40, ACM, 1992.
- [13] Kaufmann, M., J.F. Sibeyn, 'Optimal Multi-Packet Routing on the Torus,' *Proc. 3rd Scandinavian Workshop on Algorithm Theory*, pp. 118-129, Springer-Verlag, 1992.
- [14] Kaufmann, M., J.F. Sibeyn, 'Deterministic Routing on One-Dimensional Arrays,' *Proc. 4th Symp. on Parallel and Distributed Processing*, pp. 376-383, IEEE, 1992.
- [15] Kaufmann, M., J.F. Sibeyn, T. Suel, 'Derandomizing Algorithms for Routing and Sorting on Meshes,' *Proc 5th Symposium on Discrete Algorithms*, SIAM, 1994.
- [16] Sibeyn, J.F., *Algorithms for Routing on Meshes*, Ph. D. Thesis, Universiteit Utrecht, Utrecht, 1992.
- [17] Sibeyn, J.F., M. Kaufmann, R. Raman, 'Randomized Routing on Meshes with Buses,' *Proc. 1st European Symposium on Algorithms*, LNCS 726, pp. 333-344, Springer-Verlag, 1993.
- [18] Sibeyn, J.F., 'Deterministic Sorting on Circular Arrays,' *Proc. Computing Science in the Netherlands*, SION, 1993.
- [19] Sibeyn, J.F., M. Kaufmann, 'The 1- k Routing Problem on Meshes, with Applications to Worm-Hole Routing,' *Proc. 11th Symp. on Theoretical Aspects of Computer Science*, Springer Verlag, 1994.

3.1.5 Parallel Algorithms for Geometric Problems on Network Machines

Investigator: Christine Rüb

This work is concerned with parallel algorithms for geometric problems in the plane (e.g. convex hull, triangulation of a point set, red-blue intersection detection) that can be implemented efficiently on existing parallel machines. Most of these machines consist of a collection of processors, each with its own local memory, that are connected by some interconnection network. Communication between the processors is done via message passing through this network. Interconnection networks used in existing machines are,

e.g., the two-dimensional mesh (e.g. Intel Paragon), the hypercube (e.g. NCubes NCube, Intel iPSC/860), or the fat tree (e.g. Thinking Machines CM5).

For somebody implementing an algorithm this means that, in general, the interconnection network of the machine he uses is fixed. Also, the number p of available processors is independent of the size n of the input and, in fact, in general $p \ll n$. Other points of consideration are that communication is, to a varying degree, relatively expensive compared to local computation, and that only a few message passing routines like broadcast and global operations on standard data types are available to start with. Last not least there is always a fast sequential algorithm that acts as an opponent: a parallel algorithm has to be significantly faster than the best known sequential algorithm.

From the above observations follows that a practicable parallel algorithm can be implemented to run on various networks and is scalable, i.e., the speedup achieved is a function of the number p of processors and not dependent on the size of the input (ideally, the speedup is p). Since communication is relatively expensive, we are interested in algorithms that do much work locally, i.e. sequentially, and because of the sequential opponent the constant factors involved should be small.

Here we propose parallel algorithms for several geometric problems in the plane that consist of local computation plus some basic parallel routines such as merging, sorting, or prefix computations. One advantage of this approach is that one can abstract from the underlying architecture of the machine used: only the implementation of the basic routines depends on the architecture. These basic routines have in common that they can be implemented efficiently on many parallel machines: they are scalable, have small constant factors, do much work locally, and send mostly long messages. (The latter is important on many parallel machines: sending several short messages takes much more time than sending the same information in one long message.)

We have considered the following geometric problems in the plane:

1. Convex hull of a point set
2. Upper/Lower envelope of non-intersecting line segments
3. Triangulation of a point set
4. All nearest neighbours
5. Point location
6. Red-blue intersection detection
7. Red-blue intersection reporting.

The running times of the algorithms depend on the architecture used; here we list the running times for a hypercube machine. For the first 6 problems we achieve a running time of $O((n/p)\log(n/p) + (\log p)^2)$, and for the 7th problem a running time of $O((n/p)(\log(n/p) + (\log p)^2) + k/p)$ if $p \leq \sqrt{n}$, where p is the number of processors used, n is the size of the input, and k is the number of points of intersection in the last problem. (Note that $O((n/p)(\log(n/p) + (\log p)^2))$ is the time used by the best known

practicable algorithm to sort n elements on a p -node hypercube.)

Problems 1, 2, and 5–7 were considered previously in [DFR93] and [DF93]. The running times achieved there are $O((n/p)(\log(n/p) + (\log p)^2))$ for problems 1 and 2, $O((n/p)(\log n \log p + (\log p)^2))$ for problems 4 and 5, and $O((n/p)(\log n \log p + (\log p)^2) + k/p)$ for problem 6. All these results hold only under the restriction that $n \geq p^2$ ($n \geq p^2 \log p$ for problem 2). That is, our results are asymptotically as least as good as the previously known results and either valid for a larger range of p or else asymptotically faster. This is due to the fact that the basic parallel routines used here are more powerful than the basic routines used in [DF93] and [DFA93], namely sorting, broadcast and total exchange.

We have implemented the basic routines on an Intel iPSC/860 and started to implement some of the proposed geometric algorithms.

References

[DF93] O. Devillers, A. Fabri, Scalable Algorithms for Bichromatic Line Segment Intersection Problems on Coarse Grained Multicomputers, Proceedings of the 3rd Workshop on Algorithms and Data Structures, LNCS 709, 1993, 277–288.

[DFR93] F. Dehne, A. Fabri, A. Rau-Chaplin, Scalable Parallel Geometric Algorithms for Coarse Grained Multicomputers, Proceedings of the 9th Annual ACM Symposium on Computational Geometry, 1993, 298–307.

Work of our group:

[R93] Ch. Rüb, Scalable Parallel Algorithms for some Geometric Problems in the Plane, manuscript, 1993.

3.1.6 Lower Bounds for Parallel Algorithms

Investigator: Shiva Chaudhuri

The CRCW PRAM is one of the most frequently used models of parallel computation. Many algorithms are easily described on this model, and it has therefore become an important vehicle for expressing parallelism. To complement the effort in developing algorithms on this model, it is important to study lower bounds on this model. Our efforts in this direction are classified into the three categories below.

Small domain lower bounds

Lower bounds in parallel computation often depend critically on the domain size of the problem that is being solved. Typically these lower bounds use Ramsey theoretic arguments to force the algorithms to behave in a structured manner on some subset of the inputs. However, applying Ramsey theoretic arguments necessitates assuming an unrealistically large domain size. These lower bounds become invalid when considering smaller domains. Thus, a major thrust of parallel complexity is to prove lower bounds for problems defined on smaller domains.

In [9], we investigated the complexity of *chaining*, a simple problem, on a CRCW PRAM with n processors. Informally, the problem is, given an input consisting of n bits, link the 1's in the input into a chain. We show a lower bound of $\Omega(\alpha(n))$ time for this problem. This lower bound is tight, since the problem can be solved in time $O(\alpha(n))$ [2, 6]. This implies, via reductions, lower bounds for several related problems: ordered chaining, prefix maxima, range maxima and parenthesis matching with nesting level. These problems appear frequently as subproblems in parallel algorithms, for example, in integer sorting, merging, lowest common ancestor and compaction (see [5, 4, 6]).

A measure of CRCW PRAM complexity

The computation of Boolean functions by circuits leads naturally to their study in all models of parallel computation. Much work has been done on investigating properties of Boolean functions which are measures of the difficulty of computing the function. *Sensitivity* is one such measure of Boolean functions which has been extensively studied [3]. Cook, Dwork and Reishuk [3] show that the complexity of computing a function f on a CREW PRAM is related to the sensitivity of f . They prove a lower bound of $\Omega(\log s_f)$ on the time required to compute f . Because of the close relationship between CRCW PRAMs and unbounded fan-in circuits, it is an interesting open problem to find a measure of Boolean functions which classifies the complexity of computing the function on a CRCW PRAM. In [11], we investigate this problem.

The AND function has sensitivity n and therefore takes $\Theta(\log n)$ time on a CREW PRAM. However, AND can be computed in constant time on a CRCW PRAM. Thus, sensitivity is not an appropriate measure of CRCW PRAM complexity. We investigate another measure, *everywhere sensitivity*, defined by Vishkin and Wigderson [7]. An intuitive interpretation of the everywhere sensitivity of a function is the minimum number of input bits whose values need to be revealed to convince an adversary of the value of the function. Our main result is that computing a function $f : D^n \rightarrow R$ of everywhere sensitivity $es(f)$ requires time $\Omega(\log \log es(f) / (\log 4P|D| - \log es(f)))$ on a CRCW PRAM with $P \geq n$ processors and unbounded memory. The lower bound holds for nonuniform algorithms as well. For computing, with n processors, a Boolean function of everywhere sensitivity n , for instance, PARITY, this gives a lower bound of $\Omega(\log \log n)$. This is weaker than the bound of $\Omega(\log n / \log \log n)$ obtained by Beame and Håstad [1]. How-

ever, surprisingly, for n processors and everywhere sensitivity $\Omega(n)$, the bound is tight for nonuniform algorithms.

Lower bounds for approximate problems

A useful paradigm in computation is the computation of increasingly accurate approximations to the object sought, until it is eventually computed exactly. The AKS sorting network is perhaps the most dramatic example of the use of this paradigm. Recently, some problems, which have the common feature that they are all *approximate* versions of problems, have been the subject of much study. For some applications, it is enough to solve the approximate version, which can often be solved faster than the exact version [13, 8]. For each approximate problem, there is an *accuracy parameter* $\lambda \geq 1/(n+1)$. When $\lambda = 1/(n+1)$ each of the above three problems reduces to its exact version, which is known to require $\Omega(\log n / \log \log n)$ time, by the lower bound of Beame and Håstad [1]. However, as λ increases, these lower bound techniques no longer apply; in fact, many of these problems can be solved in *poly*($\log \log n$) time. In [11], we develop techniques for proving lower bounds for some of these problems.

In approximate selection, the task is to find, from n elements, an element whose rank differs from a specified rank by at most λn . In approximate counting, given a bit vector, the goal is to compute an integer that lies between $s/(\lambda+1)$ and $s(\lambda+1)$, where s is the number of 1's in the input vector. We prove the following bounds: approximate selection with accuracy $\lambda \leq 1/4$ with Cn processors requires $\Omega(\log[\log n / \log C])$ time. Approximate counting with accuracy $\lambda \geq 2$ using Cn processors requires $\Omega(\log[\log n / (\log \lambda + \log C)])$ time. These bounds are easily seen to imply lower bounds for other approximate problems such as *interval allocation* and *approximate prefix summation* [12, 13]. In particular, the bound for approximate counting directly implies the bound for *approximate compaction* proved in [10].

The methods used to prove the lower bounds are of independent interest, being general enough to have applications to other computational models.

References

- [1] P. Beame and J. T. Håstad. Optimal bounds for decision problems on the CRCW PRAM. *Journal of the ACM*, **36** (1989), pp. 643–670.
- [2] O. Berkman and U. Vishkin, “Recursive \star -Tree Parallel Data Structure”, *Proc. of 30th IEEE FOCS*, 1989, 196–202.
- [3] S. Cook, C. Dwork and R. Reischuk. Upper and Lower Time Bounds for Parallel Random Access Machines Without Simultaneous Writes. *SIAM Journal on Computing*, Vol. 15, No. 1, (1986), pp. 87-97.

- [4] J. Gil and L. Rudolph, "Counting and Packing in Parallel", *International Conference on Parallel Processing*, 1986, 1000-1002.
- [5] Y. Matias and U. Vishkin, "On Parallel Hashing and Integer Sorting", *Proc. of 17th ICALP*, 1990, 729-743.
- [6] P. Ragde, "The Parallel Simplicity of Compaction and Chaining", *Proc. 17th ICALP*, 1990, 744-751.
- [7] U. Vishkin and A. Wigderson. Trade-offs between depth and width in parallel computation. *SIAM Journal on Computing.*, 14 (1985) pp. 303-314.

Work of our group:

- [8] S. Chaudhuri, T. Hagerup and R. Raman. Approximate and Exact Deterministic Parallel Selection. In *Proc. 18th Math. Fdtns. of Comp. Sci.*, Springer LNCS, Vol. 711, (1993), pp. 352-361.
- [9] S. Chaudhuri and J. Radhakrishnan, "The Complexity of Parallel Prefix Problems on Small Domains", *Proc. 33rd IEEE FOCS*, 1992, 638-647.
- [10] S. Chaudhuri, "A Lower Bound for Linear Approximate Compaction", *Proc. of 2nd Israel Symp. on Theory of Comp. and Sys.*, 1993, 25-32.
- [11] S. Chaudhuri, "Sensitive Functions and Approximate Problems", *Proc. of 34th IEEE FOCS*, 1993.
- [12] T. Hagerup. Fast Deterministic Processor Allocation. In *Proc. 4th ACM-SIAM SODA* (1993), pp. 1-10.
- [13] T. Hagerup and R. Raman. Fast Approximate and Exact Parallel Sorting. In *Proc. 5th Annual SPAA* (1993), pp. 346-355.

3.1.7 A Scalar-Product Circuit

Investigators: Michael Müller, Christine Rüb, Wolfgang Rülling

In recent years, methods for solving numerical problems have been developed that, in contrast to traditional numerical methods, compute intervals, which are proven to contain the true solution of the given problem (cf. [1], [2]). These methods rely on an exact evaluation of inner product expressions in order to obtain good (i.e. small) enclosure

intervals. Practical experiments have shown that, using these methods, even ill conditioned problems can generally be solved with maximum accuracy. Since the exact inner product computation is a basic operation for these methods we developed a circuit to support the *difficult part of the inner product computation: the accurate accumulation of the partial products.*

Probably the best way to do the accumulation exactly is to use a long fixed-point accumulator in which all intermediate sums can be stored with full accuracy. But if the summands are products of double precision floating-point numbers, the accumulator must have more than 4000 bits. The problem is that a carry resulting from an addition can propagate over almost the whole accumulator. A similar problem arises when we want to determine the most significant bit of the accumulator to round its contents to a floating-point number. It is too expensive to solve these problems sequentially and even a carry propagation tree of this size is not affordable. Our solution is to partition the accumulator into words and to associate with each word information which makes it possible to handle the problems sufficiently fast with a moderately sized circuit (cf. [3]). The circuit which we have developed using a semi custom design system with 1.2 μm CMOS technology has a size of 8.8 mm \times 8.75 mm and consists of approximately 5000 standard cells (60 000 transistors), 4 K bit RAM and 7 K bit ROM (cf. [4]). Assuming typical conditions (supply voltage 5.0 V, temperature 25°C) our simulations yield that the addition of a floating-point summand takes about 575 ns and the rounding and outputting of the result takes about 850 ns. Note that our summands are exact products of double precision floating-point numbers; a Transputer T800-30 for example needs 667 ns for a double precision multiplication.

References

- [1] U. Kulisch and W. L. Miranker, *Computer Arithmetic in Theory and Practice*, Academic Press, New York, 1981
- [2] U. Kulisch and W. L. Miranker (eds.), *A New Approach to Scientific Computation*, Academic Press, New York, 1983

Work of our group;

- [3] M. Müller, Ch. Rüb, W. Rülling, Exact Accumulation of Floating-Point Numbers, *Proceedings of the 10th IEEE Symposium on Computer Arithmetic*, pp. 64–69, 1991

- [4] M. Müller, *Entwurf eines Chips für auslöschungsfreie Summation von Gleitkommazahlen*, Ph.D. Thesis, Universität des Saarlandes, 1993

3.2 Computational Geometry

3.2.1 Proximity Problems

Investigators: Sunil Arya, Mordecai Golin, Sanjiv Kapoor, Hans-Peter Lenhof, Rajeev Raman, Christian Schwarz, Michiel Smid

Proximity problems are among the fundamental problems in computational geometry. A large part of our work on these problems can be found in the Ph.D. theses of Hans-Peter Lenhof [6] and Christian Schwarz [8]. The above mentioned investigators worked—in close collaboration—on the following problems.

The closest pair problem: Given a set S of n points in \mathbb{R}^d , the goal is to find a closest pair in S , i.e., points $P, Q \in S$ such that $d(P, Q) = \min\{d(p, q) : p, q \in S, p \neq q\}$. Here, $d(p, q)$ denotes the Minkowski L_t -distance between p and q for a fixed $1 \leq t \leq \infty$. Although this problem has been solved optimally already in 1976, we developed a new optimal algorithm: In [4], a very simple randomized algorithm is presented, having an expected running time of $O(n)$. (Provided the floor-function is available at unit-cost.) This algorithm can also be implemented in the algebraic decision tree model of computation. Then the expected running time becomes $O(n \log n)$, which is optimal in this model. Both implementations are the most simple algorithms among all known optimal closest pair algorithms.

In [4], a more general problem is solved optimally: Given k , $1 \leq k \leq \binom{n}{2}$, the algorithm finds the k closest pairs, i.e., the k smallest among all $\binom{n}{2}$ distances, in $O(n \log n + k)$ time. This problem arises e.g. in molecular biology: Atoms that are close together interact more than atoms that are far apart. Hence, it is important to identify the atoms that are close. Our algorithm is sufficiently simple that it can be implemented. (At this moment, a student implements it. Even for higher dimensions the algorithm performs very well.)

In the dynamic closest pair problem, we have to maintain the closest pair if points are inserted and/or deleted in the set S . Intuitively, insertions are easier to handle than deletions: If a point is inserted, we only have to check the neighborhood of this new point. On the other hand, if we delete a point that is part of the closest pair, then we have to find the new closest pair. Indeed, we obtained an optimal solution for the case where only insertions have to be supported: In [9], a data structure of size $O(n)$ is given that maintains the closest pair in $O(\log n)$ time per insertion.

For the fully dynamic problem, no optimal solutions are known, although we designed

near-optimal solutions. In [10], a data structure of size $O(n(\log n)^d)$ is given that maintains the closest pair in $O((\log n)^d \log \log n)$ time per insertion and deletion. This was the first closest pair data structure having polylogarithmic update time. Recently, this result was improved in [5]: For $d \geq 3$, the closest pair can be maintained in $O((\log n)^{d-1} \log \log n)$ time per update using $O(n)$ space. In the planar case, there is a data structure of size $O(n)$ that maintains the closest pair in $O((\log n)^2 / (\log \log n)^k)$ time per update. Here, k is an arbitrary fixed integer.

The above mentioned dynamic data structures are all deterministic. In [3], a randomized data structure is given that maintains—for any dimension $d \geq 2$ —the closest pair in $O((\log n)^2)$ expected time per insertion and deletion. This structure has $O(n)$ expected size.

Clustering problems: Again, a set S of n points in \mathbb{R}^d is given. In addition, we are given an integer k , $2 \leq k \leq n$. The goal is to find k points that are as close as possible. This problem has many applications, e.g. in pattern recognition and statistics. It turns out that different closeness measures give rise to different algorithms. In [2], we give a general transformation: Given any algorithm that solves the k -point clustering problem, the transformation produces another algorithm that solves the same problem by reducing it to $O(n/k)$ clustering problems, each for only $O(k)$ points. Using this transformation, we obtained the currently best known results for these problems.

The post-office problem: In this problem, we want to store a set $S \subseteq \mathbb{R}^d$ of n points (= post-offices) in a data structure such that for any query point $q \in \mathbb{R}^d$ we can efficiently find a point in S that is closest to q . Besides of theoretical interest, this problem is important in areas such as pattern recognition, data compression and speech processing. Researchers from the latter area are interested in solutions for moderate dimensions d .

In the plane, the post-office problem can be solved optimally by means of Voronoi diagrams. In higher dimensions, however, this problem is very difficult to solve. In view of this, we studied the approximate post-office problem: Given a query point $q \in \mathbb{R}^d$, find a point $p \in S$ such that the euclidean distance between q and p is at most $1 + \epsilon$ times the distance between q and its true neighbor. Note that for most practical applications, a solution to this version of the problem suffices.

In [1], a static optimal solution to this approximation problem is given: The data structure has size $O(n)$ and a query time of $O(\log n)$. In [5], a dynamic data structure is given having size $O(n(\log n)^{d-1})$ that solves queries in $O((\log n)^{d-1} \log \log n)$ time. In this data structure, points can be inserted and deleted in $O((\log n)^{d-1} \log \log n)$ time.

References

- [1] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A. Wu. *An optimal algorithm for approximate nearest neighbor searching*. To appear in Proc. 5th SODA, 1994.

Work of our group:

- [2] A. Datta, H.-P. Lenhof, C. Schwarz, M. Smid. *Static and dynamic algorithms for k -point clustering problems*. Proceedings 3rd (WADS), LNCS Vol. 709, Springer-Verlag, 1993, 265–276.
- [3] M. Golin, R. Raman, C. Schwarz and M. Smid. *Randomized data structures for the dynamic closest-pair problem*. Proc. 4th SODA, 1993, 301–310.
- [4] M. Golin, R. Raman, C. Schwarz and M. Smid. *Simple randomized algorithms for closest pair problems*. Proc. 5th Canadian Conf. Computational Geometry, 1993, 246–251.
- [5] S. Kapoor and M. Smid. *New techniques for exact and approximate dynamic closest-point problems*. In preparation.
- [6] H.-P. Lenhof. *Distanz- und Suchprobleme in der algorithmischen Geometrie und Anwendungen in der Bioinformatik*. Ph.D. Thesis. Universität des Saarlandes, Saarbrücken, 1993.
- [7] H.-P. Lenhof and M. Smid. *Enumerating the k closest pairs optimally*. Proceedings 33rd FOCS, 1992, 380–386.
- [8] C. Schwarz. *Data structures and algorithms for the dynamic closest pair problem*. Ph.D. Thesis. Universität des Saarlandes, Saarbrücken, 1993.
- [9] C. Schwarz, M. Smid and J. Snoeyink. *An optimal algorithm for the on-line closest pair problem*. Proc. 8th ACM Symp. on Computational Geometry, 1992, 330–336.
- [10] M. Smid. *Maintaining the minimal distance of a point set in polylogarithmic time*. Discrete Comput. Geom. 7 (1992), 415–431.

3.2.2 Randomized Incremental Constructions (RICs)

Investigators: Kurt Mehlhorn, Stefan Meiser (till Aug 93), Michael Müller, Ronald Rasch, Joachim Ziegler

Incremental construction, i.e., to solve a problem of size n by first solving a problem of size $n-1$ and then extending the solution to the full size, is a basic algorithmic paradigm. Clarkson and Shor [CS89] have shown that in the geometric setting adding the elements in random order frequently yields algorithms with optimal expected running time. They have demonstrated this for such diverse problems as convex hulls, Voronoi diagrams, line segment intersections, and intersections of spheres. They have also given a general but quite complicated and unintuitive analysis of the expected running time of RICs.

In [CMS92] we give a simple expected case analysis using and extending the backwards analysis technique of R. Seidel [Sei91]. We can treat not only insertions but also deletions. An alternative and equally simple analysis was given by Mulmuley [Mul90]. We also give a tail estimate for the space complexity of RICs. In [MSW92] we extend the tail estimate technique to the time complexity of RICs for line segment intersection. A general tail estimate for time is still an unresolved problem. In [AGK⁺92] we treat the related question of optimum stopping rules for randomized algorithms.

We applied RICs to Abstract Voronoi diagrams, cf. section 3.2.3, and to the convex hull problem in arbitrary dimensions. For the latter problem we obtained an optimal data structure for convex hulls under random insertions and deletions [CMS92]. Müller and Ziegler [MZ93] describe an implementation of the convex hull algorithm.

References

- [CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Journal of Discrete and Computational Geometry*, pages 387–421, 1989.
- [Mul90] K. Mulmuley. A fast planar partition algorithm, I. *J. Symbolic Computation*, pages 253–280, 1990.
- [Sei91] R. Seidel. Backwards analysis of randomized geometric algorithms. *New Trends in Discrete and Computational Geometry*, J. Pach (ed.), Springer Verlag, pages 37–67, 1993.

Work of our group:

- [AGK⁺92] H. Alt, L. Guibas, R. Karp, K. Mehlhorn, and A. Widgerson. A method for obtaining randomized algorithms with small tail probabilities. *Rep. Max-Planck-Institut für Informatik*, (MPI-I-92-110), 1992.
- [CMS92] Kenneth L. Clarkson, Kurt Mehlhorn, and Raimund Seidel. Four results on randomized incremental constructions. In *Computational Geometry: Theory*

and Applications, volume 3, pages 185–212, 1993. Full version available as MPI-report MPI-I-92-112.

- [MSW92] K. Mehlhorn, M. Sharir, and E. Welzl. Tail estimates for the space complexity of randomized incremental constructions. In *ACM-SIAM Symposium on Discrete Algorithms*, volume 3, pages 89–93, 1992. Full version available as MPI-report MPI-I-91-113.
- [MZ93] M. Müller and J. Ziegler. An implementation of a convex hull algorithm. manuscript, 1993.

3.2.3 Abstract Voronoi Diagrams

Investigators: Kurt Mehlhorn, Stefan Meiser (till Aug 1993), Ronald Rasch, Michael Seel, Nicole Zimmer

Since 1975, when Shamos and Hoey ([SH75]) discovered Voronoi diagrams for computer science, Voronoi diagrams are 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. For a survey on the topic we refer to [Aur91], [LS86], [Oka92], [Kle89] or [Mei93], which provide exhaustive collections of examples.

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 site among the elements of S . Different types of Voronoi diagrams are obtained by varying the space, frequently $M = \mathbb{R}^2$, the shape of the sites, such as points, spheres, polyhedra, and the distance function, e.g., L_p -norms, convex distance functions, weighted distance functions.

With the notion of abstract Voronoi diagrams Klein [Kle89] has proposed a unifying approach to Voronoi diagrams for the important case $M = \mathbb{R}^2$. Klein's approach covers the most important types of 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 abstract Voronoi diagrams. From the algorithmic point of view the interest was focused on the design of an algorithm for the construction of all types of Voronoi diagrams included in the abstract Voronoi diagram model. Early results [MMD91] led to an $O(n \log n)$ randomized algorithm where $n = |S|$. This algorithm was the first algorithm computing abstract Voronoi diagrams in their full generality provided a certain general position assumption is satisfied. The key idea consists in reducing the dependency from the particular kind of Voronoi diagram to a single basic operation, namely the construction of a Voronoi

diagram of five sites. Later, Klein, Mehlhorn, and Meiser [KMM90, KMM93, Mei93] removed the general position assumption. Further improvements in the computation of abstract Voronoi diagrams benefit from new results in the theory of randomized incremental constructions [BDS⁺92, CMS92].

Instances of the algorithm have been implemented by Zimmer [Zim92] and Seel [See93] for the following types of Voronoi diagrams:

1. point sites under L_1 - and L_2 -metric
2. Power diagrams
3. line segments under L_2 -metric

In contrast to an earlier version Seel's implementation also checks the legality of the input without violating the asymptotic time bound.

In respect to the mathematical treatment of abstract Voronoi diagrams Meiser [Mei93] contributed a simplified characterization of abstract Voronoi diagrams and significantly enlarged the collection of Voronoi diagrams known to be enclosed by the abstract Voronoi diagram model.

Recent results show that the abstract Voronoi diagram approach even allows to consider higher order Voronoi diagrams. In particular, Mehlhorn, Meiser, and Rasch [MMR92] investigate furthest site abstract Voronoi diagrams and give an description of the main properties of these diagrams. Furthest site abstract Voronoi diagrams are shown to have tree structure and linear complexity. Additionally, the work includes an $O(n \log n)$ randomized algorithm for the computation of these diagrams which preserves the features of the algorithm calculating nearest site abstract Voronoi diagrams.

References

- [Aur91] F. Aurenhammer. Voronoi diagrams — a survey. *ACM Computing Surveys*, 23(3):395–405, 1991.
- [BDS⁺92] J. D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete & Comp. Geometry*, 8(1):51–72, 1992.
- [Kle89] R. Klein. *Concrete and Abstract Voronoi Diagrams*. LNCS 400, Springer Verlag, 1989.
- [LS86] D. Leven and M. Sharir. *Intersection and Proximity Problems and Voronoi Diagrams*, pages 187–228. J. Schwartz and C. K. Yap (Eds.), *Advances in Robotics*, Vol. 1, Lawrence Erlbaum, 1986.

- [Oka92] A. Okabe, B. Boots, and K. Sugihara. *Spatial tessellations: concepts and applications of Voronoi diagrams*. Wiley, New York, 1992.
- [SH75] M. I. Shamos and D. Hoey. Closest point problems. In *Proc. 16th IEEE Symp. on Foundations of Computer Science*, pages 151–162, 1975.

Work of our group:

- [CMS92] K. L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. In *Computational Geometry: Theory and Applications*, 3:185–212, 1993.
- [KMM90] R. Klein, K. Mehlhorn, and S. Meiser. On the construction of abstract Voronoi diagrams, part II. In *SIGAL Symp. on Algorithms, Tokyo, LNCS 450*, pages 138–154, 1990.
- [KMM93] R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract Voronoi diagrams. In *Computational Geometry: Theory and Applications*, 3:157–184, 1993.
- [Mei93] S. Meiser. *Zur Konstruktion abstrakter Voronoidiagramme*. PhD thesis, Universität des Saarlandes, 1993.
- [MMD91] K. Mehlhorn, S. Meiser, and C. Ó’ Dúnlaing. On the construction of abstract Voronoi diagrams. *Discrete & Computational Geometry*, 6:211–224, 1991.
- [MMR92] K. Mehlhorn, S. Meiser, and R. Rasch. Furthest Site Abstract Voronoi Diagrams. Technical Report MPI-I-92-135, Max-Planck-Institut für Informatik, Saarbrücken, 1992.
- [See93] M. Seel. Eine Implementierung Abstrakter Voronoidiagramme. Master’s thesis, Universität des Saarlandes, 1993. in preparation.
- [Zim92] N. Zimmer. Die Konstruktion abstrakter Voronoidiagramme. Master’s thesis, Universität des Saarlandes, 1992.

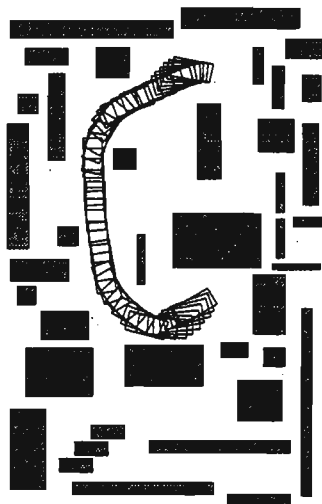
3.2.4 Motion Planning

Investigator: Stefan Schirra

Here we consider the geometric version of a fundamental problem in robotics:

Given a geometric description of a robot and its environment and two positions of the robot, is there a collision-free path between these positions?

If yes, we would also like to get such a path. This problem is known as the *mover's problem*. For this problem many exact algorithms have been developed in the past decade, see [3, 4]. Measured in the size of the environment, e.g. the number of polygon corners, they have polynomial worst case time complexity. However, the degree of the polynomials that give the bounds on the running time, is quite high and excludes these algorithms from being used in practice. We have shown that approximate algorithms can result in much better running times in many practical applications. This was already known, in fact, the first algorithms for the mover's problem have been approximate.



In contrast to former approaches our algorithms come with a bound on the approximation and are much faster for “easy problems”. Here “easy” reflects the human intuition of the difficulty of a motion planning problem. In Figure 1, the problems in the left and the right example are easy.

The problems are obviously solvable and unsolvable respectively. In an intuitive sense, a solvable problem is easy if a much larger object could be moved as well, e.g. moving a pencil through an office door. An unsolvable problem is easy if a much smaller object cannot be moved either, e.g. moving a car through an office door. If the state of the problem switches from solvable to unsolvable or vice versa by varying the size of the object slightly, then the problem is intuitively difficult, e.g. moving a desk through an office door. So far, the time bounds of motion planning algorithms have been given with respect to the size of the environment (number of polygonal corners) exclusively. In Figure 1 however, the polygonal environment is the same in all three cases. The complexity of our algorithms depends on the intuitive notion of the difficulty of a motion planning problem, which we call the *tightness* of the problem, as well as on the size of the environment.

Let us be more precise. In [5] we started with an approximate algorithm for moving a rectangle R with sides of length a and b between polygons. Let \mathcal{P} be a motion planning problem for rectangle R . For a real number $\alpha > 0$ we use αR to denote the rectangle with sides αa and αb and \mathcal{P}_α the problem \mathcal{P} with R replaced by αR . The tightness $\varepsilon_{\text{crit}}$

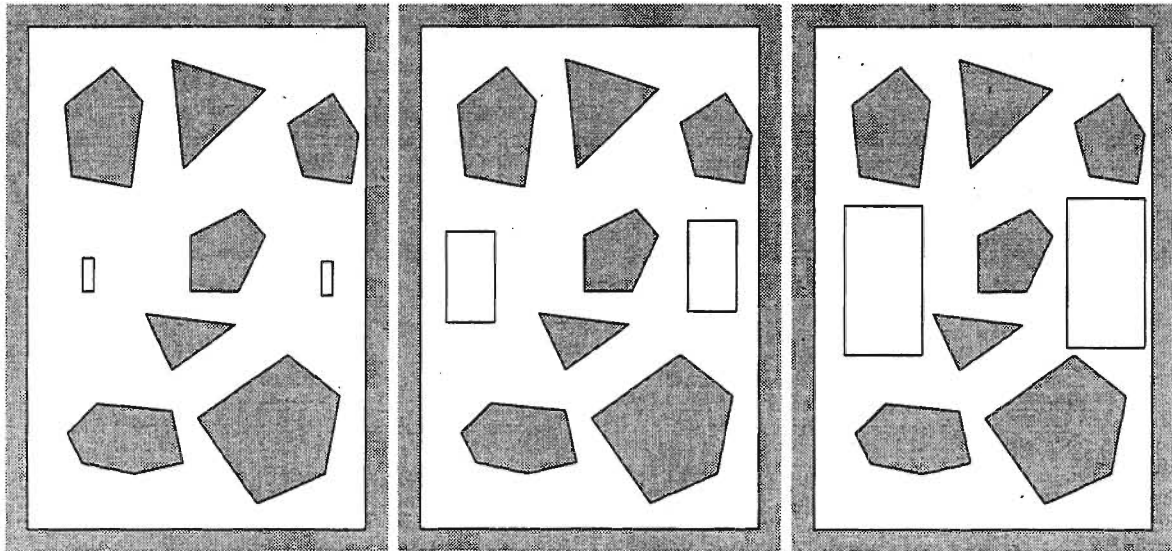


Figure 1: Motion planning problems with different tightness

of \mathcal{P} is given as follows:

- If \mathcal{P} is solvable then $\varepsilon_{\text{crit}} = \inf\{\varepsilon; \mathcal{P}_{1+\varepsilon} \text{ is unsolvable}\}$.
- If \mathcal{P} is unsolvable then $\varepsilon_{\text{crit}} = \inf\{\varepsilon; \mathcal{P}_{1/(1+\varepsilon)} \text{ is solvable}\}$.

We show in [5] that the motion planning problem for a rectangle amidst polygonal obstacles can be solved in time $O((\frac{1}{\varepsilon_{\text{crit}}} + 1)n(\log n)^2)$ where n is the number of corners of the polygons and $\varepsilon_{\text{crit}}$ the tightness of the problem. In [6] this has been improved to $O((\frac{1}{\varepsilon_{\text{crit}}} + 1)n \log n)$. The bounds for the best known exact algorithms for this problem are in $\Omega(n^2)$ [1, 2].

In [6] the basic idea has been generalized to more complex motion planning problems like chains of polygons with joints moving amidst polygonal obstacles in the plane or polyhedra moving amidst polyhedral obstacles in 3-space.

References

- [1] L.P. Chew and K. Kedem. High-clearance motion planning for a convex polygon among polygonal obstacles. Technical Report 184/90, Tel Aviv University, 1990.
- [2] K. Kedem and M. Sharir. An efficient motion planning algorithm for a convex polygonal object in 2-dimensional space. *Discrete and Computational Geometry*, 5:43–75, 1990.
- [3] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
- [4] J.T. Schwartz and M. Sharir. Algorithmic motion planning in robotics. In *Handbook of Theoretical Computer Science Vol. A: Algorithms and Complexity*. Elsevier, 1990.

Work of our group:

- [5] H. Alt, R. Fleischer, M. Kaufmann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig. Approximate motion planning and the complexity of the boundary of the union of simple geometric figures. *Algorithmica*, 8:391–408, 1992.
- [6] S. Schirra. *Approximative Bewegungsplanungsverfahren*. PhD thesis, Universität des Saarlandes, 1992.

3.2.5 Approximate Decision Algorithms for Geometric Pattern Matching

Investigator: Stefan Schirra

We investigate algorithms that decide whether two geometric objects resemble each other to a given degree after applying a transformation from a set of allowed transformations \mathcal{T} . The practical motivation behind our study comes from computer vision, where an observed image is compared to a hypothesized model. We have been mainly interested in comparing point sets.

Perfect resemblance of two geometric patterns means geometrically *congruence*. Due to inaccuracies in the input data and rounding-errors in the computations exact congruence is of theoretical interest only, see [4]. More realistic is *approximate congruence*, i.e. congruence with a tolerable error bound. We say that point sets A and B are *congruent with tolerance ε* , or ε -congruent, if there exists an isometry I and a bijection $\ell : A \rightarrow B$ such that $\text{dist}(I(a), \ell(a)) \leq \varepsilon$, for all $a \in A$, where $\text{dist}(\cdot, \cdot)$ is the distance function for our chosen metric.

Several researchers have studied approximate congruence, notably Baird [1] and Alt, et al. [4]. The distinguishing feature of their decision algorithms is the high run-time: no algorithm is known for $\mathcal{T} =$ set of translations with $o(n^6)$ run-time, and for $\mathcal{T} =$ set of rigid motions, the best known bound is $\Theta(n^8)$. For models with a large number of points, such performance is unacceptable. Therefore we developed *approximate decision algorithms*: For a specified metric and isometry class, let $\varepsilon_{\text{opt}}(A, B)$ denote the minimum value of ε such that A and B are ε -congruent. We call decision algorithms which always return a correct answer *complete decision algorithms*, while an algorithm which either returns a correct answer or chooses not to answer we call an *approximate decision algorithm*. An approximate decision algorithm is called (α, β) -approximate [7], if, for any $\varepsilon \notin [\varepsilon_{\text{opt}}(A, B) - \alpha, \varepsilon_{\text{opt}}(A, B) + \beta]$, it correctly answers a query, and for $\varepsilon \in [\varepsilon_{\text{opt}}(A, B) - \alpha, \varepsilon_{\text{opt}}(A, B) + \beta]$, it either answers correctly or chooses not to answer. We call $[\varepsilon_{\text{opt}}(A, B) - \alpha, \varepsilon_{\text{opt}}(A, B) + \beta]$ the *indecision interval*. An (α, β) -approximate

algorithm has the desirable property that it will not return an incorrect answer; if it is not sure, it will simply say that it does not know the answer.

Let n be the cardinality of A and B . In [7] we presented (γ, γ) -approximate algorithms for approximate congruence with tolerance ε with running time $O((\varepsilon/\gamma)^2 n^{2.5})$ for translations and running time $O((\varepsilon/\gamma)^2 n^4)$ for rigid motions. In [6] we improved these bounds to $O((\varepsilon/\gamma)^4 n^{1.5})$ and $O((\varepsilon/\gamma)^5 n^{2.5})$ respectively. So for rigid motions we have a speed-up of $\Omega(n^{5.5})$ if $\gamma = \varepsilon/c$ for some constant $c \geq 1$.

It is possible to remove the indecision from our approximate decision algorithms, and obtain complete decision algorithms whose time-complexity is dependent on the difficulty of the problem instance. Specifically, if we think of $K_\varepsilon = |\varepsilon_{opt}(A, B) - \varepsilon|$ as the “difficulty parameter”, then each of our approximate decision algorithms can be transformed into a complete decision algorithm, with K_ε replacing γ in the time bound.

Recently we investigated approximate decision algorithms for Hausdorff-distance of point sets [5]. In some sense Hausdorff-distance of point sets under (a subset of) rigid motions is a relaxation of approximate congruence. Point sets A and B are called ε -close if their Hausdorff-distance

$$h(A, B) := \max\left(\max_{a \in A} \min_{b \in B} \text{dist}(a, b), \max_{b \in B} \min_{a \in A} \text{dist}(b, a)\right)$$

is at most ε . Here it is sufficient that every point has an ε -close point in the other point set. A matching (and hence equal cardinality) is not required. In [5] we present algorithms with indecision interval $[\delta - \gamma, \delta + \gamma]$ where δ is the Hausdorff-distance. For point sets of cardinality n and m resp. we get running time $O((\varepsilon/\gamma)^2 (n+m) \log(n+m))$ for Hausdorff-distance under translations and $O((\varepsilon/\gamma)^2 nm \log(nm))$ for Hausdorff-distance under rigid motions. So far the best (complete) decision algorithm for Hausdorff-distance under translation has running time $O(nm(n+m) \log(nm))$ [3]. For rigid motions the best bound is $O(m^3 n^2 \log(nm))$ [2].

References

- [1] H.S. Baird. *Model-Based Image Matching Using Location*. MIT Press, 1984.
- [2] L.P. Chew, M.T. Goodrich, D.P. Huttenlocher, K. Kedem, J.M. Kleinberg, and D. Kravets. Geometric pattern matching under Euclidean motion. In *Proc. of the 5th Canadian Conference on Computational Geometry*, pages 151–156, 1993.
- [3] D.P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of Voronoi-surfaces and its applications. In *Proc. of the 7th ACM Comp. Geom. Conference*, pages 194–203, 1991.

Work of our group:

- [4] H. Alt, K. Mehlhorn, H. Wagnen, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete and Computational Geometry*, 3:237–256, 1988.
- [5] L.P. Chew, K. Kedem, and S. Schirra. Approximate decision algorithms for Hausdorff-distance. in preparation, 1993.
- [6] P.J. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. In *Proc. of the 8th ACM Symp. on Computational Geometry*, pages 93–101, 1992.
- [7] S. Schirra. Approximate decision algorithms for approximate congruence. *Information Processing Letters*, 43:29–34, 1992.

3.3 Data Structures and Graph Algorithms**3.3.1 Competitive Analysis of On-Line Algorithms**

Investigators: Susanne Albers and Boris Teia

Many on-line problems can be formulated as follows. An on-line algorithm is presented with a sequence of *requests* that must be served in their order of occurrence. In particular, the on-line algorithm must satisfy each request without knowledge of any future requests. The processing of requests incurs cost and the goal is to minimize the cost incurred on the entire request sequence. *Competitive analysis* [ST85] is a powerful means to analyze the performance of on-line algorithms. An on-line algorithm A is called c -competitive if there exists a constant a such that for every request sequence, the cost incurred by A is at most a plus c times the cost incurred by the optimal off-line algorithm. The optimal off-line algorithm knows the entire request sequence in advance and pays only the minimum cost. The *competitive factor* of A is the infimum of all c such that A is c -competitive.

We have investigated the problem of *lookahead* in on-line algorithms: What improvement can be achieved in terms of competitiveness if an on-line algorithm knows not only the present request to be served but also some future requests. In many applications, requests do not necessarily arrive one after the other but rather in blocks of possibly variable size. It may also be possible to delay the service of requests so as to wait for some incoming requests. In general, an interesting question is, what is it worth to know a part of the future. The two main problems that we have studied with respect to lookahead are paging and the list-update problem. We have also addressed caching, the k -server problem and metrical task systems.

First, we have introduced a new model of lookahead. Consider the intuitive model of lookahead which we call *weak lookahead*. An algorithm is on-line with a weak lookahead of size l if it always sees the present request and exactly l future requests. This model of lookahead is usually of little or no advantage. The reason is that an adversary can replicate requests in the lookahead, thereby weakening the effect of lookahead. The new model of lookahead that we have defined is called *strong lookahead*. An algorithm is on-line with a strong lookahead of size l if it sees the present request and a sequence of future requests. This sequence contains l pairwise distinct requests which also differ from the present request. Our definition is motivated by an analysis of request sequences that occur in practice: Subsequences of consecutive requests generally contain a number of distinct requests. From a theoretical point of view, we require an adversary to reveal some really significant information on future requests. Our main results can be summarized as follows.

Paging: The paging problem is to decide which pages to store in a small fast memory so as to minimize the number of page faults. Let k be number of pages that can be held in fast memory. The competitive factors of optimal deterministic and randomized on-line paging algorithms without lookahead equal k and $H(k)$, respectively, [ST85, FKLMSY91, MS91]. Here $H(k)$ denotes the k -th harmonic number. It is well known that weak lookahead cannot improve these factors at all. Strong lookahead is the first model of lookahead that can reduce the competitive factors. We have developed a variant of the algorithm LRU (Least Recently Used) that, given a strong lookahead of size l , is $(k - l)$ -competitive. The algorithm is optimal because we can also show that no deterministic on-line algorithm with strong lookahead l can be better than $(k - l)$ -competitive. In the area of randomized algorithms we have developed a variant of the MARKING algorithm [FKLMSY91] that is $2H(k - l)$ -competitive if a strong lookahead l is given. This competitiveness is nearly optimal since no randomized on-line algorithm with strong lookahead l can be better than $H(k - l)$ -competitive. These bounds hold against the oblivious adversary. We have also developed a number of nearly optimal on-line algorithms which make only little use of lookahead.

The list-update problem: The problem consists of maintaining a list of n items as an unsorted linear list. Each request is an access to an item in the list, where accessing the i -th item in the list incurs a cost of i . The competitive factor of optimal deterministic on-line algorithms without lookahead equals 2 [ST85, KR90]. We have derived lower bounds on the competitive factors of on-line algorithms if a strong or weak lookahead of size l is given. We have shown that an on-line algorithm requires a strong lookahead of size $\Omega(n)$ in order to be better than 2-competitive. If an on-line algorithm is given a weak lookahead, the situation is worse. A lookahead of size $\Omega(n^2)$ is necessary to asymptotically beat the competitive factor of 2. However, the constants hidden in the Ω -notation are very small. For this reason, we have also developed competitive on-line

algorithms for both models of lookahead.

See [A92, A93a] for more detailed results on lookahead in on-line algorithms.

Another problem that we have addressed is the k -server problem [MMS88]. The problem consists of scheduling the motion of k mobile servers which cover points in a metric space M . A request sequence consisting of points in M must be served. In response to each request, a server must be moved to the requested point, unless a server is already present. The goal is to minimize the total distance traveled by the servers. Manasse *et al.* [MMS88] demonstrated that no deterministic on-line algorithm for the k -server problem can be better than k -competitive. They also conjectured that for every k there exists a k -competitive deterministic on-line k -server algorithm. This conjecture is still open. So far, k -competitive algorithms are known only for the following special cases: (a) $k = 2$ [MMS88]; (b) the total number of points in the metric space equals $k + 1$ [MMS88]; (c) the metric space is isomorphic to a line or tree [CKPV90, CL91]; (d) the uniform metric space (paging) [ST85]. However, the proposed algorithms differ in each case. Coppersmith *et al.* [CDRS90] have presented a more general approach. They gave a randomized on-line algorithm that is k -competitive for resistive spaces. In fact, all the four cases mentioned above are examples of resistive spaces. We have developed a deterministic k -server algorithm, called HANDICAP, that is k -competitive in the cases (a)–(d), see [T93]. Furthermore, HANDICAP is k -competitive against the *lazy adversary*. An adversary is lazy if it always poses the next request at a point at which it has a server but the on-line algorithm has not (provided that there is such a point). Finally we have proved that HANDICAP is 157-competitive if $k = 3$. The analysis of that proof is quite loose and we conjecture that HANDICAP is 3-competitive for $k = 3$.

A third problem that we have studied is the on-line replication problem. Consider a net of processors, each of which has its local memory. These local memories store pages that are assumed to be read-only. Suppose a processor p wants to read an information from page B . If B is stored in p 's local memory, then this read operation can be satisfied at zero cost. Otherwise, p has to access the local memory of the closest processor q with the page and incurs a cost equal to the distance from p to q . If p has to read page B frequently, it might be worthwhile to replicate B to p 's local memory. However, such a replication incurs a high cost equal to β times the distance from p to q , where β is the page size factor. The replication problem is to decide which pages should be replicated to which processors. Black and Sleator [BS89] considered that case that the processor net forms a tree and presented a deterministic on-line replication algorithm that achieves an optimal competitive factor of 2. Recently, Koga [K93] gave a randomized 1.71-competitive algorithm for trees and a 4-competitive algorithm for circles. We have developed a randomized replication algorithm for trees which achieves a competitive factor of $(\frac{e}{e-1}) \approx 1.58$. The algorithm is optimal because we can show that no randomized

on-line replication algorithm can be better than $(\frac{e}{e-1})$ -competitive. Furthermore, we have developed a technique for transforming a large class of c -competitive algorithms for trees into $2c$ -competitive algorithms for circles. As a result, we obtain a randomized $(\frac{2e}{e-1})$ -competitive algorithm for circles. We also derived two 4-competitive algorithms for circles that are either memoryless or use only one random number during an initialization phase and run completely deterministically thereafter. See [A93b] for details.

References

- [BS89] D.L. Black and D.D. Sleator. Competitive algorithms for replication and migration problems. Technical Report Carnegie Mellon University, CMU-CS-89-201, 1989.
- [CKPV90] M. Chrobak, H. Karloff, T. Payne and S. Vishwanathan. New results on server problems. In *Proc. 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 291-300, 1990.
- [CL91] M. Chrobak and L.L. Larmore. An optimal on-line algorithm for k servers on trees. *SIAM Journal on Computing*, 20(1):144-148, 1991.
- [CDRS90] D. Coppersmith, P. Doyle, P. Raghavan and M. Snir. Random walks on weighted graphs, and applications to on-line algorithms. In *Proc. 22nd Annual ACM Symposium on Theory of Computing*, pages 369-378, 1990.
- [FKLMSY91] A. Fiat, R.M. Karp, L.A. McGeoch, D.D. Sleator and N.E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12:685-699, 1991.
- [IRWS91] S. Irani, N. Reingold, J. Westbrook and D.D. Sleator. Randomized competitive algorithms for the list update problem. In *Proc. 2nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 251-260, 1991.
- [KR90] R. Karp and P. Raghavan. Personal communication, transmitted through [IRWS91].
- [K93] H. Koga. Randomized on-line algorithms for the page replication problem. To appear in *Proc. 4th International Annual Symposium on Algorithms and Complexity*, 1993.
- [MMS88] M.S. Manasse, L.A. McGeoch and D.D. Sleator. Competitive algorithms for on-line problems. In *Proc. 20th Annual ACM Symposium on Theory of Computing*, pages 322-333, 1988.

[MS91] L.A. McGeoch and D.D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6:816-825, 1991.

[ST85] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, 28:202-208, 1985.

Work of our group:

[A92] S. Albers. The influence of lookahead in competitive on-line algorithms. Technical Report Max-Planck-Institut für Informatik, MPI-I-92-143, 1993.

[A93a] S. Albers. The influence of lookahead in competitive paging algorithms. In *Proc. 1st European Symposium on Algorithms*, pages 1-12, 1993.

[A93b] S. Albers. Improved randomized algorithms for the replication problem. Manuscript, November 1993.

[T93] B. Teia. Ein Beitrag zum k -Server Problem. Ph.D. Thesis, Max-Planck-Institut für Informatik, 1993.

3.3.2 Network Flow Algorithms

Investigators: J. Cheriyan (till Nov 1990), T. Hagerup, K. Mehlhorn

Given a directed graph G , a source vertex and a sink vertex, and a capacity function on the edges, the problem is to compute a maximal flow from the source to the sink. This problem has a long history and is one of the basic problems of combinatorial optimization. Previous to our work, the best algorithms had running time $O(n^3)$ [Kar74] and $O(nm \log(n^2/m))$ [GT88].

Cheriyan and Hagerup [CH89] introduced the idea of randomization into network flow algorithm. They achieve a running time of $O(nm + n^2(\log n)^3)$ thus reaching the long-standing goal of $O(nm)$ running time for all non-sparse graphs. In [CHM91], we simplify and refine their approach and add a bit-compression technique to speed up the search for the so-called "eligible edges". We achieve a bound of $O(\min\{nm, n^3/\log n\})$. For dense graphs, this breaks the $O(n^3)$ barrier. In [CM91], we show the usefulness of the bit-compression technique for other graph problems. In particular, we show how to solve the maximal matching problem in bipartite graphs in time $O(n^{2.5}/\log n)$. The latter result was also obtained by Motwani and Feder [FM91] using completely different

techniques.

All our network flow algorithms were derandomized by Alon and King, Rao, and Tarjan, and Philipps and Westbrook [Alo90, KRT92, PW93].

References

- [Alo90] N. Alon. Generating pseudo-random permutations and maximum flow algorithms. *IPL*, 35:201–204, 1990.
- [FM91] T. Feder and R. Motwani. Clique partitions, graph compression and speeding-up algorithms. *23rd Annual ACM Symposium on Theory of Computing*, pages 123–133, 1991.
- [GT88] A.V. Goldberg and R.E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35:921–940, 1988.
- [Kar74] A.V. Karzanov. Determining the maximal flow in a network by the method of preflows. *Soviet Math. Dokl.*, 15:434–437, 1974.
- [KRT92] V. King, S. Rao, and R.E. Tarjan. A faster deterministic max-flow algorithm. *ACM-SIAM Symposium on Discrete Algorithms*, 3:157–164, 1992.
- [PW93] S. Philipps and J. Westbrook. Online load balancing and network flow. *25th Annual ACM Symposium on Theory of Computing*, pages 402–411, 1993.

Work of our group:

- [CH89] J. Cheriyan and T. Hagerup. A randomized maximum flow algorithm. *IEEE Symposium on Foundations of Computer Science*, 30:118–123, 1989. Full version available as Tech. Rep. no 988, School of Operations Research and Industrial Engineering, Cornell University, October 1991.
- [CHM91] J. Cheriyan, T. Hagerup, and K. Mehlhorn. An $o(n^3)$ maximum-flow algorithm. Technical Report MPI-I-91-120, Max-Planck-Institut für Informatik, 1991. A preliminary version was published in ICALP 90.
- [CM91] J. Cheriyan and K. Mehlhorn. Algorithms on dense graphs. Technical Report MPI-I-91-114, Max-Planck-Institut für Informatik, 1991.

3.3.3 Data Structures for Sets and Sequences

Investigators: Paul Dietz, Rudolf Fleischer, Kurt Mehlhorn, Rajeev Raman, Rajamani Sundar, Christian Uhrig

One of the most common (and most important) data structures used in efficient algorithms are balanced search trees. A great variety of them can be found in the literature. Basically, they all store a set of n keys such that location, insertion and deletion of keys can be accomplished in $O(\log n)$ worst case time.

In general, updates (insertions or deletions) are done in the following way: First, locate the place in the tree where the change has to be made; second, perform the actual update; and third, rebalance the tree to guarantee that future query times are in $O(\log n)$. The second step usually takes only $O(1)$ time, whereas steps 1 and 3 both need $O(\log n)$ time. However, there are applications which do not need the first step because it is already known where the key has to be inserted or deleted in the tree. In these cases we would like to have a data structure which can do the rebalancing step as fast as the actual update, i.e. in constant time.

It has been well known for a long time that some of the standard balanced search trees can achieve $O(1)$ amortized update time once the position of the key is known ([O82]). However, for the worst case update time, the best known method had been a complicated $O(\log^* n)$ algorithm by Harel ([H79]). Recently Levkopoulos and Overmars came up with an algorithm achieving optimal $O(1)$ update time ([LO88]). They use the *bucketing technique* of [O82]: Rather than storing single keys in the leaves of the search tree, each leaf (*bucket*) can store up to $O(\log n)$ keys. In fact, the buckets in [LO88] have size $O(\log^2 n)$; therefore, [LO88] also need a 2-level hierarchy of lists to guarantee $O(\log n)$ query time within the buckets.

In [F92], we simplify their approach considerably and reduce the bucket size to $2 \log n$, which means that we need only an ordered list to store the elements of a bucket. The analysis of our algorithm seems simpler and more natural than in [LO88].

The main disadvantage of the bucketing technique used in our data structure as well as in [LO88] is the fact that finger searches can not be supported efficiently. Only a non-combinatorial data structure (using bit manipulations and precomputed tables) is known which achieves constant update time and efficient finger searches ([DR90]).

In many applications the implementation of some additional operations concerning sets (besides insertions and deletions) can be of interest (for instance for the implementation of high-level languages such as SETL ([S74]) and Hermes ([SBGLYY91]), where sets and sequences are supported as primitive data types).

Suppose we want to maintain an initially empty family \mathcal{F} of sets S over a universe U and allow such operations as *Insert*(x, S) (insert the element $x \in U$ into set $S \in \mathcal{F}$), *Delete*(x, S), *Member*(x, S) (is x in S ?), *Equal*(S_1, S_2) (is S_1 equal to S_2 ?), *Subset*(S_1, S_2)

(is S_1 a subset of S_2 ?), and $Intersection(S_1, S_2)$ (compute and report the set $S_1 \cap S_2$). Yellin ([Yel92]) designed a data structure that supports all these operations and achieves a running time of $O(\sqrt{n}(n+q) \log n)$ for processing an intermixed sequence of n updates (*Insert*, *Delete*) and q queries (*Member*, *Subset*, *Equal*, *Intersection*), when n and q are not known in advance. In [DMRU93], we present a simpler solution improving the running time to $O((q+n\sqrt{q}) \log n)$. We also give a lower bound proof for the arithmetic monoid model ([Fre81], [Yao85]) and show that our upper bound, up to factors being polylog in n , is tight in this model.

Furthermore, we consider a variant of the problem mentioned above, where we only allow a limited number of memory cells, say m . For this variant we present an algorithm with running time $O(n^2 \log n/m^{1/3})$ for processing a sequence of n operations (updates as well as queries). Again we prove a matching lower bound (up to polylog in n factors).

Another interesting problem is the maintenance of an initially empty family \mathcal{F} of sequences s over a universe U where the operations $Makesequence(s, a)$ (create the new sequence s consisting of $a \in U$), $Concatenate(s_1, s_2, s_3)$ (create the new sequence s_3 by concatenating the existing sequences s_1 and s_2 without destroying them), $Split(s_1, s_2, s_3, i)$ (create the new sequences s_1 and s_2 by splitting s_3 at position i without destroying it) and $Equal(s_1, s_2)$ (is s_1 equal to s_2 ?) have to be supported efficiently. We are especially interested in solutions where equality tests can be performed in constant time. Pugh ([Pugh88]), and Pugh and Teitelbaum ([PT89]) gave a randomized representation of sequences that supports updates in $O(\log n)$ expected time, where n is the total length of the sequences involved. However, their solution only handles sequences where no duplicate elements occur. Sundar and Tarjan ([ST90]) presented a deterministic solution and achieved $O(\sqrt{n \log m} + \log m)$ amortized time for an update operation using $O(\sqrt{n})$ amortized space. Here, m denotes the total number of operations performed so far. In [MSU93], we present a randomized solution that handles the general case and needs $O(\log^2 n)$ expected time and space. We also give a deterministic data structure which is essentially a derandomization of the randomized one. It supports update operations in time $O(\log n(\log m \log^* m + \log n))$ using $O(\log n(\log n + \log^* m))$ space.

References

- [DR90] P.F. Dietz, R. Raman. A Constant Update Time Finger Search Tree. *Advances in Computing and Information — ICCI '90, LNCS, Vol. 468, Springer*, 100–109, 1990.
- [Fre81] M. L. Fredman. A Lower Bound on the Complexity of Orthogonal Range Queries. *Journal of the ACM*, 28: 696–705, 1981.

- [H79] D. Harel. Fast Updates with a Guaranteed Time Bound per Update. Technical Report, Dept. of ICS, University of California at Irvine, 1979.
- [LO88] C. Levkopoulos, M.H. Overmars. A Balanced Search Tree with $O(1)$ Worst-Case Update Time. *Acta Informatica* **26**, 269–277, 1988.
- [O82] M.H. Overmars. A $O(1)$ Average Time Update Scheme for Balanced Search Trees. *Bull. EATCS*, 18: 27–29, 1982.
- [O83] M.H. Overmars. The Design of Dynamic Data Structures. *Lecture Notes in Computer Science, Vol. 156*, Springer, 1983.
- [Pugh88] W. Pugh. Incremental Computation and the Incremental Evaluation of Functional Programming. *Ph.D. Thesis, Cornell University*, 1988.
- [PT89] W. Pugh, and T. Teitelbaum. Incremental Computation via Function Caching. In *Proc. 16th ACM POPL*, 315–328, 1989.
- [S74] J.T. Schwartz. On Programming: An Interim Report on the SETL Project, Installments I and II. *CIMS, New York University*, 1974.
- [SBGLYY91] R.E. Strom, D.F. Bacon, A.P. Goldberg, A. Lowry, D.M. Yellin, and S. Yemini. Hermes: A Language for Distributed Computing. *Prentice-Hall, Englewood Cliffs, NJ*, 1991.
- [ST90] R. Sundar, and R.E. Tarjan. Unique Binary Search Tree Representation and Equality-Testing of Sets and Sequences. In *Proc. 22nd ACM STOC*, 18–25, 1990.
- [Yao85] A. C. Yao. On the Complexity of Maintaining Partial Sums. *SIAM Journal on Computing*, 14: 277–288, 1985.
- [Yel90] D. Yellin. Representing Sets with Constant Time Equality Testing. *Journal of Algorithms*, 13: 353–373, 1992.
- [Yel92] D. Yellin. Data Structures for Set Equality-Testing. In *Proc. 3rd Annual ACM-SIAM SODA*, 386–392, 1992.

Work of our group:

- [DMRU93] P. Dietz, K. Mehlhorn, R. Raman, and C. Uhrig. Lower Bounds for Set Intersection Queries. In *Proc. 4th Annual ACM-SIAM SODA*, 194–201, 1993. To appear in *Algorithmica*.

- [F92] R. Fleischer. A Simple Balanced Search Tree with $O(1)$ Worst-Case Update Time. Technical Report MPI-I-92-101, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, January 1992. To appear in ISAAC '93.
- [MSU93] K. Mehlhorn, R. Sundar, and C. Uhrig Maintaining Dynamic Sequences under Equality Tests in Polylogarithmic Time. Technical Report MPI-I-93-128, July 1993. To appear in *Proc. 5th Annual ACM-SIAM SODA*, 1994.

3.3.4 Bottom-Up-Heapsort

Investigators : Rudolf Fleischer, Christian Uhrig, Bhabani Sinha

Bottom-Up-Heapsort is a variant of the classical Heapsort algorithm due to Williams ([Wi]) and Floyd ([F64]). The input to both algorithms is an array $a[1..n]$ of n elements from an ordered set S which are to be sorted. We will measure the complexity of the algorithms in terms of number of comparisons.

First the elements will be arranged in form of a heap with the biggest element at the root. This means that the array is considered as a binary tree where node i has children $2i$ and $2i + 1$, and that a parent node contains a bigger element than its children. This requires $O(n)$ time ([Wi]). Then follows the *Selection Phase* which consists of n *Rearrangement Steps*. In each *Rearrangement Step*, the root element changes place with the last element in the array; then the heap is rearranged with respect to the remaining elements. So the size of the heap decreases by one in each *Rearrangement Step*. Since the root always contains the biggest heap element, the array will be filled step by step from the end with elements in decreasing order.

The classical rearrangement procedure works as follows. At the beginning, the root contains a former leaf element (the last array element is always a leaf). This element is repeatedly swapped with the bigger one of its children until it is bigger than both of its children or it is a leaf. At each level two comparisons are made. Hence the total complexity of the Selection Phase might be as big as $2n \log n$.

In Bottom-Up-Heapsort, the rearrangement procedure is changed in the following way. We first compute the *special path* ([We]) which is the path on which the leaf element would sink in the classical rearrangement procedure. This is the unique path with the property that any node on it (except the root) is bigger than its sibling, and costs only one comparison per level. Then we let our leaf element climb the special path up to its destination node at the additional cost of one comparison per level.

This algorithm tries to make use of the intuitive idea that leaf elements are likely to sink back down almost to the bottom of the heap, so one can expect climbing up to

be cheaper than sinking down. In fact, Wegener ([We]) showed an upper bound of $\frac{3}{2}n \log n + O(n)$ for Bottom-Up-Heapsort. He also conjectured a tighter upper bound of $n \log n + o(n \log n)$, but we could construct a heap with an asymptotic lower bound of $\frac{5}{4}n \log n - O(n \log \log n)$ comparisons ([FSU]) and later even a heap with asymptotic $\frac{3}{2}n \log n - O(n \log \log n)$ comparisons, matching the upper bound ([F91]). This bound also implies an asymptotic upper bound of $n \log n + O(n \log \log n)$ for the best case of the classical Heapsort algorithm, as has been suspected for many years. This conjecture has been proven at the same time but independently by [SS] using very similar methods.

References

- [F64] R.W. Floyd. *Algorithm 245 : Treesort 3*. *Communications of the ACM* 7 (1964), pp. 701.
- [SS] R. Schaffer, R. Sedgewick. *The analysis of heapsort*. Technical Report CS-TR-330-91, Princeton University, January 1991.
- [We] I. Wegener. *BOTTOM-UP-HEAPSORT, a new variant of HEAPSORT beating on average QUICKSORT (if n is not very small)*. *Proc. MFCS '90, Lecture Notes in Computer Science*, Vol. 452, Springer 1990, pp. 516–522.
- [Wi] J.W.J. Williams. *Algorithm 232 : Heapsort*. *Communications of the ACM* 7 (1964), pp. 347–348.

Work of our group:

- [F91] R. Fleischer. *A tight lower bound for the worst case of bottom-up-heapsort*. *Proc. 2nd International Symposium on Algorithms 1991, Lecture Notes in Computer Science*, Vol. 557, Springer 1991, pp. 251–262.
- [FSU] R. Fleischer, B. Sinha, C. Uhrig. *A lower bound for the worst case of bottom-up-heapsort*. *Information and Computation* 102/2 (1993), pp. 263–279.

3.4 Realization and the LEDA Project

3.4.1 The LEDA Platform for Combinatorial and Geometric Computing

Investigators: Kurt Mehlhorn and Stefan Näher

One of the major differences between combinatorial or geometric computing and other areas of computing such as statistics, numerical analysis and linear programming is the

use of complex data types. Whilst the built-in types, such as integers, reals, vectors, and matrices, usually suffice in the other areas, combinatorial and geometric computing relies heavily on types like stacks, queues, dictionaries, sorted sequences, priority queues, graphs, points, segments, In the fall of 1988, we started a project (called LEDA for Library of Efficient Data types and Algorithms) to build a growing library of data types and algorithms. We hope that it will narrow the gap between algorithms research, teaching, and implementation. The main features of LEDA are:

- LEDA provides a sizable collection of data types and algorithms in a form which allows them to be used by non-experts. In the current version, this collection includes most of the data types and algorithms described in the text books of the area.
- LEDA gives a precise and readable specification for each of the data types and algorithms mentioned above. The specifications are short (typically, not more than a page), general (so as to allow several implementations), and abstract (so as to hide all details of the implementation). For many efficient data structures access by position is important. In LEDA, we use an item concept to cast positions into an abstract form. Most of the specifications given in the LEDA manual [8] use this concept, i.e., the concept is adequate for the description of many data types.
- LEDA contains efficient implementations for each of the data types, e.g., Fibonacci heaps for priority queues, skip lists and dynamic perfect hashing for dictionaries, and a mechanism based on multiple inheritance and dynamic binding that allows users to choose easily among different implementations for the same data type.
- LEDA contains a comfortable data type *graph*. It offers the standard iterations such as “for all nodes v of a graph do” or “for all neighbor nodes of v do”, it allows to add and delete nodes and edges and it offers arrays and matrices indexed by nodes and edges. The data type graph supports the implementation of graph algorithms in a form close to the typical text book presentation.

LEDA is implemented as a C++ class library, is available by anonymous ftp and can be used freely for research and teaching. The main concepts and some implementation details of LEDA are described in [7] and [9]. The user manual ([8]) lists the specifications of all data types and algorithms contained in the current version (3.0) of the library and gives many example programs.

LEDA is used as a basis for software construction world wide by several hundred sites including universities and industrial software companies. In particular, there is an intensive cooperation with the Siemens AG ([3]). Other projects with similar goals are described in [1], [2], [4], and [5]. However, all of these projects settle for a considerably smaller collection of data types and algorithms than LEDA does.

Current and future work includes the incorporation of various new data structures and algorithms from the area of computational geometry into LEDA including

- arbitrary precision integer and floating point data types
- basic geometric data structures for higher-dimensional geometry ([10])
- robust algorithms for the construction of different kinds of Voronoi diagrams ([6])
- augmented tree data structures based on skip lists and randomized search trees ([11])

References

- [1] G. Booch, *Software Components with Ada*, Benjamin/Cummings Publ. Company, 1987
- [2] K.E. Gorlen, S.M. Orlow, P.S. Plexico, *Data Abstraction and Object-Oriented Programming in C++*, John Wiley and Sons Publishing Company, 1990
- [3] U. Lauther, *A Fast Planning Tool for Routing and Scheduling of Cargo Trains*, ALCOM Workshop "Algorithms: Implementation, Libraries, and Use", 1993.
- [4] C. Lins, *The Modula-2 Software Component Library*, Springer Publishing Company, 1989
- [5] J. Soukup, *Organized C*, Typescript, 1988

Work of our group:

- [6] C. Burnikel, K. Mehlhorn, S. Schirra, *On Degeneracy in Geometric Computations*, Technical Report, Max-Planck-Institut für Informatik, Saarbrücken, 1993
- [7] K. Mehlhorn, S. Näher, *LEDA, a Library of Efficient Data Types and Algorithms*, Communications of the ACM, to appear
- [8] S. Näher, *LEDA User Manual Version 3.0*, Technical Report, Max-Planck Institut für Informatik, Saarbrücken, 1992
- [9] S. Näher, *Parameterized Data Types in LEDA*, in preparation
- [10] M. Neukirch, *Grundlegende geometrische Datenstrukturen und Algorithmen für LEDA*, Diplomarbeit, Max-Planck-Institut für Informatik, Saarbrücken, 1993

- [11] M. Paul, *Augmented Tree Data Structures based on Skiplists and Randomized Search Trees*, Ph.D. Thesis (in preparation), Max-Planck-Institut für Informatik, Saarbrücken, 1993.

3.4.2 Precision and Degeneracy in Geometric Computations

Investigators: Christoph Burnikel, Kurt Mehlhorn, Stefan Meiser (till Aug 1993), Stefan Näher, Stefan Schirra, Erik Schwarzenecker

The implementation of geometric algorithms is a notoriously difficult task. The collection of available correct implementations is still small, where correct means that they come with a precise description of the class of inputs for which they work. Why is the implementation so difficult? Most of the difficulties can be traced to the following sentence that appears in nearly every paper on computational geometry:

Throughout this paper we assume *exact real arithmetic* and the input to be in *general position* (no three points on a line, no four points on a circle, ...).

In general, however, geometric algorithms which are provably correct for the model of real numbers with real arithmetic will fail when executed with floating point arithmetic. In practice, such non-robustness of algorithms is frequently corrected using some ad-hoc method like “epsilon tweaking”. An epsilon parameter is added to the code, such that, for example, two objects that have “distance” less than epsilon are considered to be incident. This principle of “if it’s close to zero then it is zero” is again a common source of failure, because it gives rise to inconsistent decisions. For example, consider Fig. 2. We might detect that p_1 and p_2 are both incident to lines ℓ_1 and ℓ_2 and hence

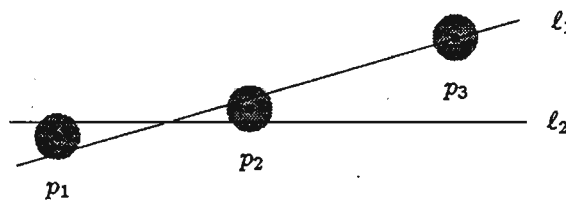


Figure 2: Inconsistent decisions

conclude that these lines are equal. Testing p_1 and p_3 , we would conclude that the lines are different. At best, these ad-hoc approaches reduce failure probability. Therefore exact arithmetic should be used.

We have been mainly interested in the computation of Voronoi-diagrams for line segments and have implemented Yap’s algorithm [6] and the randomized incremental algorithm

for abstract Voronoi-diagrams [10]. For Voronoi-diagrams of points some work has been done with respect to robustness, see e.g. [4], and robust implementations exist [5].

For line segments, however, the situation is different. While in a Voronoi-diagram for point sets all Voronoi-vertices have rational coordinates, they might have non-rational algebraic coordinates in a Voronoi-diagram for line segments. So exact rational arithmetic is not sufficient. Fortunately, computer algebra shows a way to compute exactly with algebraic numbers. The general theory, however, gives quite weak bounds. For the Voronoi diagram of line segments, we can show that if the coordinates of the endpoints of the segments are k bit integers, then precision $48k$ suffices in the computations. Moreover we do not need to resort to general techniques for dealing with algebraic numbers but only need repeated squaring. The general theorems of computer algebra give only about $1000k$.

It is folklore that degeneracies are a curse in the implementation of geometric algorithms, "because" they require many case distinctions and therefore result in lengthy error prone code. Fortunately, there is a general technique for coping with degeneracies: the *perturbation technique* [2, 7, 3]. In this technique, the input is perturbed symbolically and the computation which has to be carried out exactly, is carried out on the perturbed input. It can be shown that the perturbation schemes remove geometric degeneracies, e.g., collinearity of three points, at only a constant factor increase in running time. So perturbation seems to be the perfect solution for the problem of degeneracy. As Yap [7] puts it: the perturbation technique is "*the theoretical paradise in which degeneracies are abolished*".

In [8] we argue against this belief and put forward the claim that it is simpler (in terms of programming effort) and more efficient (in terms of running time) to avoid the perturbation technique and to deal directly with degenerate inputs. Our argument rests on the observations that on degenerate inputs the perturbation schemes may incur an arbitrary overhead in running time, that the complexity of the postprocessing required to retrieve the answer for a degenerate input x from the answer to the perturbed input $x(\epsilon)$ is significant, and that for many geometric problems algorithms handling degeneracies directly are only moderately more complex than algorithms assuming non-degenerate inputs. In [8] we substantiate these claims on two basic problems in computational geometry, the line segment intersection problem and the convex hull problem.

For concreteness, let us substantiate the first claim for the *line segment intersection problem*. The input are n line segments in the plane and the output is the planar graph whose vertices are the endpoints and the crossings of the segments and whose edges are the subsegments induced by the vertices. We use m to denote the number of vertices of this planar graph and s to denote the number of pairs of intersecting segments. Note that s might be as large as m^2 , e.g., if all segments pass through the origin. The perturbation technique yields running time $O((n + s) \log n)$ when combined

with Bentley–Ottmann plane-sweep, and time $O(s + n \log n)$ when combined with the optimal deterministic algorithm of Chazelle and Edelsbrunner [1]. We give a variant of the optimal deterministic algorithm of Chazelle and Edelsbrunner, that runs in time $O(m + n \log n)$.

In [9] we describe an algorithm for intersecting a polyhedron with a convex polyhedron, that handles all degenerate cases directly.

We can also offer some experimental evidence. The LEDA-implementation [12] of the plane sweep algorithm for line segment intersection has 581 lines of code out of which about 100 deal with degeneracies (vertical segments and high-degree intersections). The LEDA-implementation [11] of our convex hull algorithm (without deletion) has 1124 lines of code. Out of this 134 lines deal with degeneracies.

References

- [1] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM*, 39:1–54, 1992.
- [2] H. Edelsbrunner and E.P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graphics*, 9(1):67–104, 1990.
- [3] Ionnais Emiris and John Canny. An efficient approach to removing geometric degeneracies. In *Proc. of the 8th Symp. on Computational Geometry*, pages 74–82, 1992.
- [4] S. Fortune. Numerical stability of algorithms for 2D Delaunay triangulations and Voronoi diagrams. In *Proc. of the 8th ACM Symp. on Computational Geometry*, pages 83–92, 1992.
- [5] A. Okabe, B. Boots, and K. Sugihara. *Spatial tessellations: concepts and applications of Voronoi diagrams*. Wiley, New York, 1992.
- [6] C. Yap. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Discrete and Computational Geometry*, 2:365–393, 1987.
- [7] C. K. Yap. Symbolic treatment of geometric degeneracies. *J. Symbolic Comput.*, 10:349–370, 1990.

Work of our group:

- [8] C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. to appear in SODA94.
- [9] K. Dobrindt, K. Mehlhorn, and M. Yvinec. A complete and efficient algorithm for the intersection of a general and a convex polyhedron. In *WADS'93*, pages 314–324, 1993.
- [10] K. Mehlhorn, St. Meiser, and C. Ó'Dunlaing. On the construction of abstract Voronoi diagrams. *Discrete Comput. Geom.*, 6:211–224, 1991.
- [11] M. Müller and J. Ziegler. An implementation of a convex hull algorithm. manuscript, 1993.
- [12] St. Näher. *LEDA Manual*. Max-Planck-Institut für Informatik, 1993.

3.5 Complexity Theory**3.5.1 Circuit Complexity**

Investigator: Vince Grolmusz

The class **ACC** consists of those languages which are accepted by sequences of bounded-depth, polynomial circuits of AND, OR, NOT and MOD m gates, where a MOD m gate outputs 1 if the sum of its inputs is divisible by m , and 0 otherwise. Considerable efforts were done to prove that some restricted versions of **ACC** do not contain several “natural” languages. *Razborov* [4] proved that the MAJORITY function needs exponential size if it is computed by bounded-depth circuits with AND, OR, NOT and MOD 2 gates.

Smolensky [Sm] generalized this result to circuits with MOD p gates instead of MOD 2 ones, where p is a prime or prime-power. The case, where p is a non-prime-power composite number, remained widely open.

Smolensky posed the problem to prove a lower bound for depth-2 circuits with two levels of MOD 6 gates. This problem was solved by *Krause* and *Waack* [2], who proved that any depth-2 circuit with a MOD m gate at the top, and symmetric gates at the bottom needs exponential size to compute the $ID(x,y)$ function. Since the MOD m gates are also symmetric gates, this answered the question of *Smolensky*.

In [6] we attacked the class of depth-3 circuits with a threshold gate at the top, symmetric gates at the next, and MOD m gates at the bottom. With a multi-party communication technique, we proved an exponential lower bound to the size of these circuits, if they

compute the k -wise inner product function of [1]. We also used the assumption that the lower fan-in is at most k .

In [7] we proved that the communication properties of the two-level MOD m and the two-level MOD p circuits are dramatically different: the first needs linearly many bits of communication, while the second can be computed by communicating a constant number of bits.

In [8] we gave a weight-size trade-off for depth-3 circuits computing the inner product of two vectors of length n , with a weighted threshold gate at the top, AND gates on the next, and MOD m gates of arbitrary fan-in on the lowest level. We have proved that

$$\log w \log M \geq \Omega(n \log n),$$

where M is the maximum fan-in on the second level of the circuit, and w is the sum of the absolute values of the weights in the threshold gate at the top. When every weight is 1, then it gives a trade-off between the maximum fan-in on the second and on the top level. The proof of this result uses the known exponential gap between the deterministic communication complexity of the inner product function and the probabilistic communication complexity of the ID function [9, 3].

References

- [1] L. Babai, N. Nisan, M. Szegedy: Multiparty Protocols and Pseudorandom Sequences, Proc. 21st ACM STOC, 1989, pp. 1-11.
- [2] M. Krause, S. Waack: Variation ranks of communication matrices and lower bounds for depth two circuits having symmetric gates with unbounded fan-in, Proc. 32nd IEEE FOCS, 1991.
- [3] Rabin, M. unpublished.
- [4] A. A. Razborov: Lower Bounds on the Size of Bounded Depth Networks Over a Complete Basis with Logical Addition, (in Russian), Mat. Zametki, 41 (1987), 598-607.
- [5] R. Smolensky, Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity, Proc. 19th ACM STOC, pp. 77-82, (1987).

Work of our group:

- [6] V. Grolmusz: Circuits and Multi-Party Protocols, Technical Report No. MPII-1992-104, Max Planck Institute for Computer Science, Saarbruecken, Germany, 199.2
- [7] V. Grolmusz: Separating the communication complexities of MOD m and MOD p circuits, Proc. 33rd IEEE FOCS, 1992, pp. 278-287.
- [8] V. Grolmusz: Mod m Gates do not Help on the Ground Floor, Technical Report No. MPII-1993-142, Max Planck Institute for Computer Science, Saarbruecken, Germany, 199.3
- [9] Mehlhorn, K., Schmidt, E. M.: Las Vegas is better than determinism in VLSI and distributive computing, Proc. 14th ACM STOC, 1982, pp. 330-337.

3.5.2 Communication Complexity

Investigator: Vince Grolmusz

The *multi-party communication game*, defined by *Chandra, Furst* and *Lipton* [2], is an interesting generalization of the 2-party communication game. In this game, k players P_1, P_2, \dots, P_k intend to compute a Boolean function $g(x_1, x_2, \dots, x_n) : \{0, 1\}^n \rightarrow \{0, 1\}$. On set $S = \{x_1, x_2, \dots, x_n\}$ of variables there is a fixed partition into k classes A_1, A_2, \dots, A_k , and player P_i knows every variable, *except* those in A_i , for $i = 1, 2, \dots, k$. The players have unlimited computational power and they communicate with the help of a blackboard visible to all players. The goal is to compute $g(x_1, x_2, \dots, x_n)$, such that at the end of the computation every player knows this value. The cost of the computation is the number of bits written on the blackboard for the given $x = (x_1, x_2, \dots, x_n)$ and $A = (A_1, A_2, \dots, A_k)$. The theory of the 2-party communication games is well developed. The 2-party communication complexity of a function is known to be between the rank and the logarithm of the rank of a $2^n \times 2^n$ matrix, containing the values of f for all possible input allocations. Better upper bounds were given for special classes of functions by *Lovász* and *Saks* [4], using extensively lattice-theory and Moebius functions. For more than two players, no analogue results were known.

In the k -party case we have proved that the BNS-lower bound [1] is almost optimal, developing a protocol for a specific function [6].

For general Boolean functions, we succeeded to establish a close relationship between the L_1 norm of a Boolean function and its multiparty communication complexity [7]. One application of this result: If f is an arbitrary Boolean function with L_1 norm $L_1(f)$,

then there exists a $O(\log L_1(f))$ -party protocol which computes f with $O(\log^3 L_1(f))$ communication. This is a *fundamental* upper bound in the multiparty communication complexity theory.

Moreover, we have shown the surprising result that the existence of a *real function* that approximates f with small error implies the existence of an efficient *discrete* multiparty protocol for f . More exactly, suppose that $g : \{-1, 1\}^n \rightarrow \mathfrak{R}$ satisfies

$$|f(x) - g(x)| < \frac{1}{5}$$

for all $x \in \{-1, 1\}^n$. (It is allowed that g has irrational or even transcendent values.) Then there exists a $O(\log L_1(g))$ -party protocol which computes f with $O(\log^3 L_1(g))$ bits of communication [8]. Note that the parameters of the communication protocol depends *only* on the L_1 norm of the *approximating real-valued* function.

In [8] we have also proved that if the distribution of the Fourier-coefficients of a Boolean function f is uneven, more exactly, if they can be divided into two groups: one with small L_1 norm (say, L), and the other with small L_2 norm (say, ε), then there exists a $O(\log L)$ -party protocol which computes f with $O(\log^3 L)$ communication on an $(1 - \varepsilon)$ fraction of all inputs.

In [9] we have shown that almost all Boolean functions have very high k -party communication complexity. This and the previous result show an interesting spectral property of Boolean functions: almost all Boolean functions do not have real approximating functions of small L_1 norm, or: almost all Boolean functions have exponential L_1 norm, or: almost all Boolean functions have their Fourier-coefficients evenly distributed: they cannot be divided into two classes one with small L_1 , the other with small L_2 norms.

This results show the importance of the L_1 norm "measure" of the hardness of Boolean functions, versus the well-studied "degree" measure [5].

References

- [1] L. Babai, N. Nisan, M. Szegedy: Multiparty Protocols and Pseudorandom Sequences, Proc. 21st ACM STOC, 1989, pp. 1-11.
- [2] A. K. Chandra, M. L. Furst, R. J. Lipton: Multi-party Protocols, Proc. 15th ACM STOC, 1983, pp. 94-99.
- [3] L. Lovász: Communication Complexity: A Survey, Technical Report, CS-TR-204-89, Princeton University, 1989.
- [4] L. Lovász, M. Saks: Lattices, Moebius functions and communication complexity, Proc. 29th IEEE FOCS, pp. 81-90.

- [5] N. Nisan, M. Szegedy: On the degree of Boolean functions as real polynomials, Proc. 24th ACM STOC, 1992, pp. 462–467.

Work of our group:

- [6] V. Grolmusz: The BNS Lower Bound for Multi-Party Protocols is Nearly Optimal, to appear in “Information and Computation”.
- [7] V. Grolmusz: Multiparty Protocols and Spectral Norms, Technical Report No. MPII-1993-132, Max Planck Institute for Computer Science, Saarbruecken, Germany, 1993.
- [8] V. Grolmusz: Harmonic Analysis, Real Approximation and the Communication Complexity of Boolean Functions, submitted, November 1993.
- [9] V. Grolmusz: On Multi-Party Communication Complexity of Random Functions, manuscript, October 1993.

3.5.3 Symbolic Algebra

Investigator: Devdatt Dubhashi

Progress related to symbolic algebra in p -adic field originating from my Ph.D. dissertation at Cornell was reported in the two conferences, [3, 2] and a comprehensive survey appeared in the special issue of the *Computer Journal*, [1], devoted to quantifier elimination.

References

- [1] D.P. Dubhashi : *Quantifier Elimination and Decision Procedures in P-adic Fields*, The Computer Journal, Special Issue on Computational Quantifier Elimination, 36:5, September 1993.
- [2] D.P. Dubhashi : *Quantifier Elimination and Decision Procedures in P-adic Fields*, Symposium on Quantifier Elimination in honour of Prof. George E. Collins, Linz, Austria, October 6–9, 1993.
- [3] D.P. Dubhashi : *Quantifier Elimination and Decision Procedures for Linear Sentences in P-adic Fields*, 19th Latin American Conference on Informatics, 22nd JAIIO, Buenos-Aires, Argentina, Aug 2–6 1993.

4 Dissertations

completed:

Gao, Sh.: Algorithmen über VLSI-Layout. 1991

Stefan Schirra: Approximative Bewegungsplanungsverfahren 1992

Albers S.: The Influence of Lookahead in Competitive On-Line Algorithms. 1993

Fleischer R.: Genaue Analyse einiger kombinatorischer Algorithmen. 1993

Lenhof H.-P.: Distanz- und Suchprobleme in der algorithmischen Geometrie und Anwendungen in der Bioinformatik. 1993 ¹

Meiser St.: Zur Konstruktion abstrakter Voronoi-Diagramme. 1993

Müller M.: Entwurf eines Chips für auslöschungsfreie Summation von Gleitkommazahlen. 1993

Schwarz Ch.: Data Structures and Algorithms for the Dynamic Closest Pair Problem. 1993 ¹

Teia B.: Ein Beitrag zum k -Server Problem. 1993 ¹

Uhrig Ch.: Lower and Upper Bounds for Operations on Sets. 1993 ¹

ongoing:

Burnikel Ch.: Precision and Degeneracy in Geometric Computations. June 1994 ²

Klär G.: Verdrahtungsprobleme auf planaren Graphen. Dec 1993 ²

Lauer Th.: Dynamische Lastbalancierung. Mai 1994 ²

Paul M.: Augmented Tree Data Structures Based on Skiplists and Randomized Search Trees. June 1994. ²

¹Thesis already completed, but degree not yet awarded.

²Expected completion date

Priebe V.: Analysis of randomized combinatorial algorithms. Dec 1994 ²

Rasch R.: Abstrakte inverse Voronoi-Diagramme. Jan 1994 ²

Schilz T.: Verteilte Algorithmen für den Design-Rule-Check von VLSI-Layouts. June 1995 ²

Schwarzenecker E.: Ein NP-vollständiges Problem aus der Kartographie. Oct 94 ²

Thiel Ch.: Schnitt von Polyedern in höheren Dimensionen. Sept 1994 ²

²Expected completion date

5 Visitors

1991

Dr. J. Katajainen	02.04.91 - 07.04.91	University of Lund, Sweden
Prof. Juraj Hromkovic	26.04.91	Gesamthochschule Paderborn
Dr. Frank Dehne	16.05.91	Carleton University, Ottawa, Canada
Prof. Jack Snoeyink	21.05.91 - 25.05.91	University of Utrecht, The Netherlands
Prof. Gaston Gonnet	23.05.91 - 24.05.91	Zürich, Swiss
Prof. R. Seidel	06.06.91 - 09.06.91	University of California, Berkeley, USA
Dr. Amet Henri	10.06.91	France
Dr. Alain Filbois	11.06.91 - 12.06.91	France
Prof. Paul Dietz	29.07.91 - 04.08.91	University of Rochester, USA
Dr. O. Devillers	01.07.91 - 31.07.91	INRIA Valbonne, France
Dr. M. Devillers	01.07.91 - 31.07.91	INRIA Valbonne, France
Jop Sibeyn	31.07.91 - 06.09.91	University of Utrecht, The Netherlands
Prof. Leo Guibas	21.08.91 - 24.08.91	DEC SRC Palo Alto, USA
Dr. Stephen Omohundro	28.10.91	ICSI Berkeley, USA
Dr. Simon Kahan	01.10.91 - 30.06.92	University of Seattle, Washington, USA
Prof. Ian Munro	28.10.91 - 03.11.91	Princeton University, Princeton, USA
Dr. Wolf Zimmermann	14.11.91	GMD Karlsruhe
Prof. Gritzmann	05.12.91	Universität Trier
Monika Rauch	19.12.91	Princeton University, USA

1992

Prof. Asano, Tetsuo	17.08.92 - 12.09.92	Osaka Electro-Communication Univ., Neyagawa, Osaka, Japan
Dietz, Paul	13.07.92 - 14.08.92	University of Rochester, Rochester, USA
Dr. Flammini, M.	27.04.92 - 03.05.92	Universität Rom, Rom, Italien
Dr. Golin, M.	01.06.92 - 31.07.92	INRIA, Le Chesnay, Frankreich
Dr. Kant, Goos	16.06.92 - 17.06.92	Universität Utrecht, Utrecht, Niederlande
Prof. Katoh, Naoki	05.06.92 - 06.06.92	Kobe University of Commerce, Kobe, Japan
Prof. Krithivasan, K.	01.05.92 - 30.06.92	Indian Institute of Technology, Madras, Indien
Dr. Joan Lawry	18.05.92 - 13.06.92	University of Washington, Seattle, Washington, USA
Dr. Matsumoto, T.	01.09.92 - 30.09.92	University of Tokyo, Tokyo, Japan
Prof. Munro, Ian	21.06.92 - 16.08.92	University of Waterloo, Waterloo, Ontario, Canada
Prof. Papadimitriou, C.	08.10.92 - 23.10.92	University of California, La Jolla, Californien, USA
Dr. Papakostas, A.	15.05.92 - 30.07.92	University of Massachusetts Amherst, Massachusetts, USA
Rabinovich, Juri	09.06.92 - 14.06.92	Hebrew University, Jerusalem, Israel
Dr. Ranade, A.	03.08.92 - 15.08.92	Thinking Machines Corp. Cambridge, Massachusetts, USA
Dr. Roos, Thomas	01.09.92 - 30.09.92	ETH Zentrum, Zürich, Schweiz
Prof. Sack, Jörg	01.11.92 - 30.11.92	Carleton University, Ottawa, Ontario, Canada
Prof. Tokuyama, T.	15.06.92 - 17.06.92	IBM - Watson Res. C., Yorktown, New York, USA
Dr. Zelikovsky, A.	15.05.92 - 15.07.92	Institute of Mathematics, Kishinev, USSR

1993

Prof. Bilardi, G.	21.02.93 - 06.03.93	Universität Padua, Padua, Italien
Chlebus, Bogdan	31.05.93 - 29.06.93	Universität Warschau, Warschau, Polen
Dr. Fekete, S.	11.01.93 - 12.01.93	Suny Stony Brook, Stony Brook, New York, USA
Prof. Golin, M.	12.07.93 - 29.07.93	The Hong Kong Univ. of Science and Technology, Hong Kong
Prof. Guibas, L.	28.09.93 - 29.09.93	Stanford University, Stanford, California, USA
Prof. Hartmanis, J.	01.10.93 - 31.05.94	Cornell University, Ithaca, New York, USA
Ivkovic, Zoran	07.06.93 - 25.06.93	University of Delaware, Newark, Delaware, USA
Prof. Janardan, Ravi	17.07.93 - 28.07.93	University of Minnesota, Minneapolis, Minnesota, USA
Prof. Kapoor, Sanjiv	07.06.93 - 15.07.93	Indian Institute of Technology, New Delhi, Indien
Prof. Kedem, Klara	05.07.93 - 25.07.93	Ben-Gurion University Beer-Sheva, Israel
Prof. Kucera, Ludek	05.07.93 - 11.07.93	Charles University Prag, Tschechische Republik
Prof. Leiseron, Ch.	31.08.93 - 03.09.93	Massachusetts Institute of T. Cambridge, Massachusetts, USA
Mackenzie, Phil	28.06.93 - 29.06.93	University of Texas, Texas, USA
Prof. Maheshwari, A.	11.02.93 - 13.03.93	Tata Institute of Fund. Res., Bombay, Indien
Prof. Mannila, Heikki	27.05.93 - 28.05.93	University of Helsinki, Helsinki, Finland
Dr. Matousek, Jirka	06.09.93 - 28.09.93	Charles University, Prag, Tschechische Republik
Dr. Panconesi, A.	06.09.93 - 20.09.93	Universität Rom, Rom, Italien
Dr. C. Pandurangan	01.06.93 - 31.08.93	Indian Institute of Technology, Madras, Indien
Radhakrishnan, J.	12.07.93 - 25.07.93	Japan Adv. Inst. of S. and T., Tatsuuokuchi, Japan

Dr. Raman, Rajeev	05.07.93 - 15.07.93	University of Maryland, Maryland, USA
Rosen, Adi	08.09.93 - 08.09.93	Tel-Aviv University, Tel-Aviv, Israel
Dr. Salowe, Jeffrey	03.06.93 - 30.07.93	University of Virginia, Charlottesville, Virginia, USA
Schieber, Baruch	05.07.93 - 08.07.93	IBM - Watson Res. Center, Yorktown, New York, USA
Dr. Schuster, Assaf	05.04.93 - 08.04.93	Technion, Haifa, Israel
Dr. Schuster, Assaf	08.08.93 - 28.08.93	Technion, Haifa, Israel
Prof. Sen, Sandeep	15.05.93 - 15.07.93	Indian Institute of T. New Delhi, Indien
Prof. Snoeyink, Jack	29.04.93 - 30.04.93	The Univ. of British Col., Vancouver, Canada

6 Teaching Activities

The group contributes intensively to the Computer Science Curriculum of the Universität des Saarlandes. 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. Here comes the list of courses taught in the winter term 1993/94:

Course: Praxis des Programmierens

Course: Datenstrukturen und Algorithmen II

Course: Ausgewählte Kapitel aus Effizienten Algorithmen

Course: Algorithmen zur Bewegungsplanung

Course: Parallele Algorithmen mit sublogarithmischer Laufzeit

Course: Grundlagen der linearen Programmierung

Seminar: Parallele Algorithmen

Seminar: Genetische Algorithmen

Advanced Practical Course: Softwarekonstruktion.

The group advises a number of masters students. Within the last three years 35 master students wrote their masters thesis under our supervision.

7 Organization

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 in recent months were: Approximation Algorithms, Average Case Analysis of Graph Algorithms and Degeneracy in Geometric Computations.

We have elected an executive committee (K. Mehlhorn, T. Hagerup, V. Priebe, G. Barnes, A. Esser, K. Reinert) which makes the day to day decisions concerning the group.

8 Cooperations

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).

We are involved in seven research projects: BMFT-projects PAKAP and SOFTI, SFB 124 VLSI-Entwurfsmethoden und Parallelität, ESPRIT-project ALCOM, EC Cooperative Action IC-1000-project ALTEC, EC-project HC & M, GIF research project: Arrangements in Computational Geometry.

8.1 BMFT-projects

The *Bundesministerium für Forschung und Technologie* (BMFT) supports our work through projects PAKAP and SOFTI. PAKAP supports all of our applied and some of our theoretical work in parallel algorithms, and SOFTI supports our work on implementation of algorithms and computational geometry.

8.2 SFB 124 VLSI-Entwurfsmethoden und Parallelität

The Sonderforschungsbereich (SFB) 124 is a special research effort, sponsored by the DFG (Deutsche Forschungsgemeinschaft) and located at the University of the Saarland (now also at the MPI) and the University of Kaiserslautern. The 10 participating groups work on topics from the areas VLSI design methods and parallelism. The SFB 124 was founded in 1983.

8.3 ALCOM

ALCOM (Algorithms and Complexity) is an Esprit basic research action. It involves 13 partners in 9 EC countries, namely Università di Roma (Italy), Universitat Politècnica de Catalunya (Spain), University of Utrecht (Netherlands), Universität-GH Paderborn (Germany), University of Warwick (Great Britain), Computer Technology Institute of Patras (Greece), INRIA Sophia-Antipolis (France), INRIA Rocquencourt (France), Åarhus University (Denmark), University of Dublin (Irish Republic), EHESS Paris (France), FU Berlin (Germany), Max-Planck-Institut Saarbrücken (Germany). Kurt Mehlhorn and Christoph Storb coordinate the action, Torben Hagerup is the person in charge locally.

8.4 ALTEC

ALTEC (Basic Algorithms for Future Technologies) is the extension of ALCOM to Eastern Europe. It is chaired by Jan van Leeuwen, the former ALCOM coordinator. The partners of ALTEC are Utrecht, Bordeaux, Prague, Bratislava, Budapest, Warsaw, and Saarbrücken.

The purpose of ALCOM and ALTEC is to foster algorithms research within Europe and to stimulate the cooperation within Europe. In its four years of existence ALCOM has been very successful in reaching these goals. It has given the European algorithms community an identity and its own conference ESA and it has led to close collaboration within it. We work mainly with Berlin, Sophia-Antipolis, Paderborn, Dublin, Patras, and Åarhus. ALTEC, which has existed for 12 months, tries to extend this cooperation to Eastern Europe.

8.5 HC & M

Human Capital and Mobility (HC & M) is the post-doc program of the European Community. Starting 1994 we will have one post-doc position for 3 years paid by HC & M.

8.6 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.

8.7 Industry

Our main industrial partner is Siemens AG, Munich. Dr. Hammer from Siemens and T. Hagerup and T. Lauer cooperate on load balancing algorithms for parallel machines (cf. section 3.1.3), and Dr. Lauther from Siemens and St. Näher, K. Mehlhorn, and Ch. Uhrig cooperate on efficient graph algorithms and a planning tool for scheduling cargo trains (cf. section 3.4.1). Ch. Uhrig will spend three months with Lauther's group starting November 15th, 1993, and T. Lauer holds a Siemens Ph.D. scholarship. Kurt Mehlhorn serves on the scientific advisory board of Siemens corporate research.

With the Dillinger Hütte AG we cooperate on a planning tool for cutting steel plates. Through the LEDA project we have loose contacts to several other companies. These relations are of the producer-consumer type, i.e. we receive bug reports and sometimes enthusiastic comments.

9 Technical Reports

1991

- **MPI-I-91-101, to appear in Information and Computation**
Dynamic rectangular point location, with an application to the closest pair problem
Author: Michiel Smid

- **MPI-I-91-102, Algorithms Review (Newsletter of the ESPRIT II AL-COM project) 2, 1991, pp. 77-87**
Range trees with slack parameter
Author: Michiel Smid

- **MPI-I-91-103, Discrete and Computational Geometry, 7, 1992, pp. 415-431**
Maintaining the minimal distance of a point set in polylogarithmic time (revised version)
Author: Michiel Smid

- **MPI-I-91-104, Proc. 2nd Intern. Symp. on Algorithms, 1991, LNCS, Springer-Verlag, Vol. 557, pp. 251-262**
A Tight Lower Bound for the Worst Case of Bottom-Up-Heapsort
Author: Rudolf Fleischer

- **MPI-I-91-105, Proc. 6th Ann. ACM Symp. on Computational Geometry, 1990, pp. 216-224 and Algorithmica 8, 1992, pp. 391-406**
Simultaneous Inner and Outer Approximation of Shapes
Authors: R. Fleischer, K. Mehlhorn, G. Rote, E. Welzl, C. Yap

- **MPI-I-91-106**
Fast Parallel Space Allocation, Estimation an Integer Sorting
Author: Torben Hagerup

- MPI-I-91-107, Proc. 3rd Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA), 1992, pp. 280-285

An $O(n \log n \log \log n)$ algorithm for the on-line closest pair problem (This work was supported by the ESPRIT II Basic Research Actions Program, under contract No. 3075 (project ALCOM).)

Author: Christian Schwarz

- MPI-I-91-110, Proc. 8th ACM Symp. on Computational Geometry, 1992, pp. 93-101

Approximate Decision Algorithms for Point Set Congruence

Authors: Paul J. Heffernan, Stefan Schirra

- MPI-I-91-112, Proc. 2nd Ann. Intern. Symp. on Algorithms, Lecture Notes in Comp. Sci., Vol. 557, Springer-Verlag 1991, pp. 349-363

An Optimal Construction Method for Generalized Convex Layers

Authors: Hans-Peter Lenhof, Michiel Smid

- MPI-I-91-113, *Algorithmica*, 1992

Tail Estimates for the Space Complexity of Randomized Incremental Algorithms

Authors: K. Mehlhorn, M. Sharir, E. Welzl

- MPI-I-91-114

Algorithms for Dense Graphs and Networks

Authors: Joseph Cheriyan, Kurt Mehlhorn

- MPI-I-91-115, IPL 42, 1992, pp. 25-27

A Lower Bound for the Nondeterministic Space Complexity of Contextfree Recognition

Authors: Helmut Alt

Viliam Geffert

Kurt Mehlhorn

- **MPI-I-91-120, Proc. 17th Intern. Coll. on Automata, Languages and Programming (ICALP) 1990, Lect. Notes in Comp. Sci., Springer-Verlag, Vol. 443, pp. 235-248**
An $O(n^3)$ -Time Maximum-Flow Algorithm
Authors: Joseph Cheriyan, Torben Hagerup, Kurt Mehlhorn
- **MPI-I-91-121, IPL 43, 1992, pp. 335-340**
On a Compaction Theorem of Ragde
Author: Torben Hagerup
- **MPI-I-91-122**
On Embeddings In Cycles
Juraj Hromkovi, Vladimír Müller, Ondrej Sýkora and Imrich Vrto
- **MPI-I-91-123, Proc. 8th Ann. ACM Symp. on Computational Geometry, 1992, pp. 330-336 and to appear in Algorithmica**
An optimal algorithm for the on-line closest-pair problem
Christian Schwarz, Michiel Smid and Jack Snoeyink
- **MPI-I-91-124**
On Crossing Numbers of Hypercubes and Cube Connected Cycles
Ondrej Sýkora and Imrich Vrto
- **MPI-I-91-125**
Edge Separators for Graphs of Bounded Genus with Applications
Ondrej Sýkora and Imrich Vrto
- **MPI-I-91-126**
Optimal Embedding of a Toroidal Mesh in a Path
Michael S. Paterson, Heiko Schröder, Ondrej Sýkora and Imrich Vrto

1992

- **MPI-I-92-101**
A simple balanced search tree with $O(1)$ worst-case update time
Author: Rudolf Fleischer

- **MPI-I-92-102, Proc. SWAT'92, Lect. Notes in Comp. Sci., Vol. 621, Springer-Verlag, 1992, pp. 388-398**
Maintaining the Visibility Map of Spheres while Moving the Viewpoint on a Circle at Infinity
Authors: Hans-Peter Lenhof, Michiel Smid

- **MPI-I-92-104, Proc. 28th Ann. Symposium on Foundations of Computer Science, 1992, pp. 278-287**
Circuits and Multi-Party Protocols
Author: Vince Grolmusz

- **MPI-I-92-108, Proc. 4th Canadian Conf. on Computational Geometry, 1992, pp. 115-120**
Computing Intersections and Arrangements for Red-Blue Curve Segments in Parallel
Author: Christine R"ub

- **MPI-I-92-110**
A Method for Obtaining Randomized Algorithms with Small Tail Probabilities
Authors: H. Alt, L. Guibas, K. Mehlhorn, R. Karp, A. Wigderson

- **MPI-I-92-112, STACS 1992**
Four Results on Randomized Incremental Constructions
Authors: K. L. Clarkson, K. Mehlhorn, R. Seidel

- **MPI-I-92-115, Proc. 19th Intern. Coll. on Automata, Languages and Programming, 1992, Lect. Notes in Comp. Sci., Springer-Verlag, Vol. 623, pp. 318-329**
Fast Integer Merging on the EREW PRAM
Authors: Torben Hagerup, M. Kutylowski

- **MPI-I-92-118, Proc. 33rd Ann. IEEE Symp. on Foundations of Comp. Sci. (FOCS), 1992, pp. 380-386**

Enumerating the k closest pairs optimally

Authors: Michiel Smid, Hans-Peter Lenhof

- **MPI-I-92-120**

Separating the Communication Complexities of MOD m and MOD p circuits

Author: Vince Grolmusz

- **MPI-I-92-121**

Minimum Base of Weighted k Polymatroid and Steiner Tree Problem

Author: Alexander Zelikovsky

- **MPI-I-92-122**

A Faster $11/6$ -Approximation Algorithm for the Steiner Tree Problem in Graphs

Author: Alexander Zelikovsky

- **MPI-I-92-123**

The Largest Hyper-Rectangle in a Three Dimensional Orthogonal Polyhedron

Author: Kamala Krithivasan

- **MPI-I-92-125**

A New Lower Bound Technique for Decision Trees

Author: Rudolf Fleischer

- **MPI-I-92-126, SODA 1992**

Dynamic Point Location in General Subdivisions

Authors: Hanna Baumgarten, Hermann Jung, Kurt Mehlhorn

- **MPI-I-92-127, SODA 93**

A Lower Bound for Set Intersection Queries

Authors: Kurt Mehlhorn, Christian Uhrig und Rajeev Raman

- **MPI-I-92-134**
Sequential and parallel algorithms for the k closest pairs problem
Authors: Hans-Peter Lenhof, Michiel Smid
- **MPI-I-92-135**
Furthest Site Abstract Voronoi Diagrams
Authors: Kurt Mehlhorn, Stefan Meiser, Ronald Rasch
- **MPI-I-92-141, Proc. 33rd Ann. Symp. on Foundations of Computer Science (FOCS) 1992, pp. 628-637**
Waste Makes Haste: Tight Bounds for Loose Parallel Sorting
Authors: Torben Hagerup, Rajeev Raman
- **MPI-I-92-143**
The Influence of Lookahead in Competitive On-Line Algorithms
Author: Susanne Albers
- **MPI-I-92-145**
Optimal Generation of Dynamic Random Variables
Authors: Torben Hagerup, Kurt Mehlhorn, Ian Munro
- **MPI-I-92-149, Proc. 4th Ann. ACM-SIAM Symp. on Discrete Algorithms, 1993, to appear**
Fast Deterministic Processor Allocation
Author: Torben Hagerup
- **MPI-I-92-152**
Finding k points with a smallest enclosing square
Author: Michiel Smid
- **MPI-I-92-153**
Christian Schwarz: Semi-dynamic Maintenance of the Width of a Planar Point Set³
- **MPI-I-92-154**
Michiel Smid, Prosenjit Gupta, Ravi Janardan: Further results on generalized intersection searching problems: counting, reporting, and dynamization

- MPI-I-92-155

M. Golin, R. Raman, C. Schwarz, M. Smid: Simple Randomized Algorithms for Closest Pair Problems

1993

- MPI-I-93-101

Abstract and Report not yet published.

- MPI-I-93-102

Randomized data structures for the dynamic closest-pair problem

Authors: Mordecai Golin, Rajeev Raman, Christian Schwarz, Michiel Smid

- MPI-I-93-103

Tail estimates for the efficiency of randomized incremental algorithms for line segment intersection

Authors: Kurt Mehlhorn, Micha Sharir, Emo Welzl

- MPI-I-93-105

Randomized incremental construction of abstract Voronoi diagrams

Authors: Rolf Klein, Kurt Mehlhorn, Stefan Meiser

- MPI-I-93-106

Broadcasting through a noisy one-dimensional network

Author: Luděk Kučera

- MPI-I-93-107

Expected complexity of graph partitioning problems

Author: Luděk Kučera

- MPI-I-93-108

Static and dynamic algorithms for k -point clustering problems

Authors: Amitava Datta, Hans-Peter Lenhof, Christian Schwarz, Michiel Smid

- MPI-I-93-109

LEDA manual version 3.0

Author: Stefan Näher

- **MPI-I-93-110**
Coloring k -colorable graphs in constant expected parallel time
Author: Luděk Kučera

- **MPI-I-93-116**
An $O(n \log n)$ algorithm for finding a k -point subset with minimal L_∞ -diameter
Author: Michiel Smid

- **MPI-I-93-118**
Approximate and exact deterministic parallel selection
Authors: Shiva Chaudhuri, Torben Hagerup, Rajeev Raman

- **MPI-I-93-119**
Generalized topological sorting in linear time
Author: Torben Hagerup

- **MPI-I-93-121**
The circuit subfunction relations are Σ_2^P -complete
Authors: Bernd Borchert, Desh Ranjan

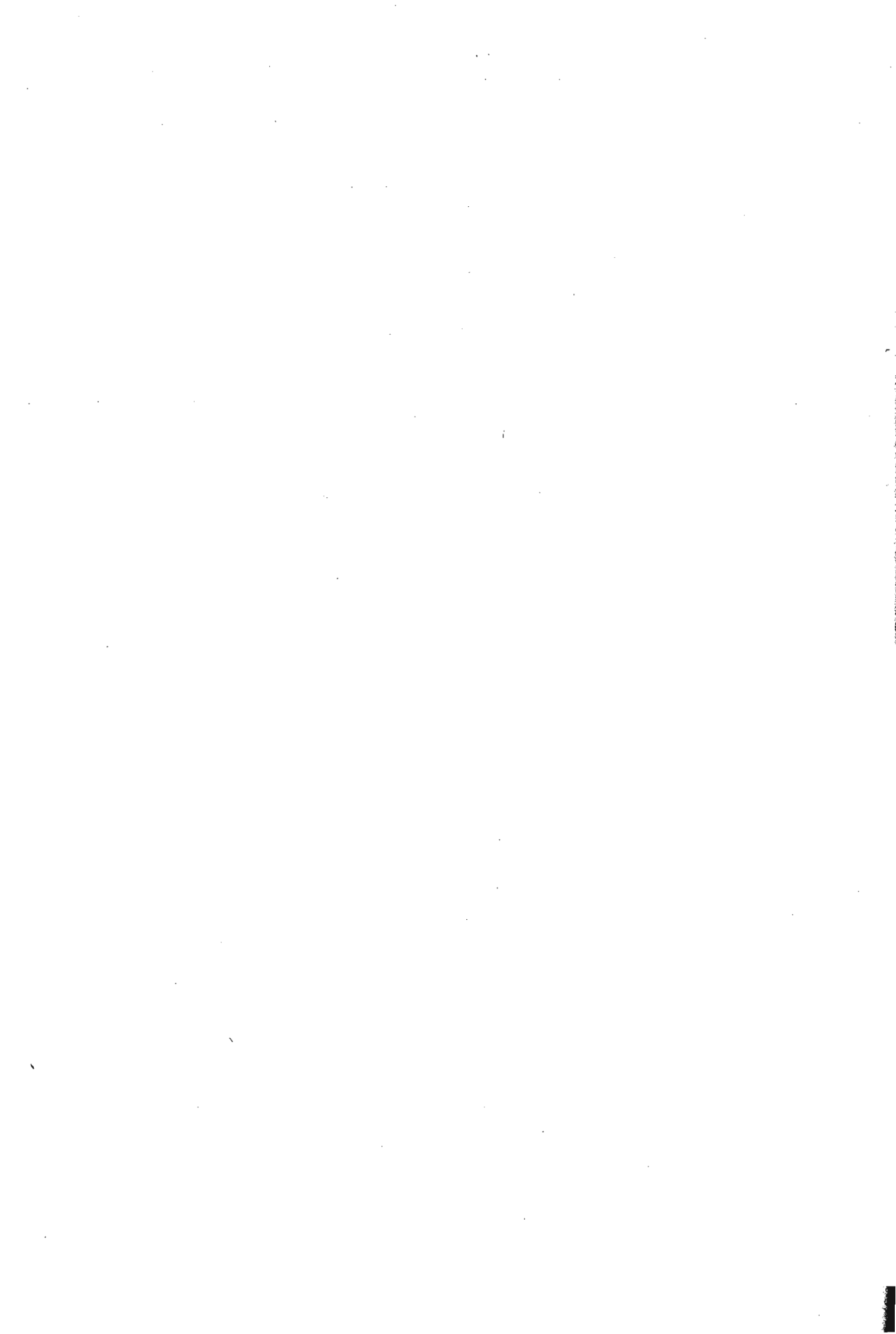
- **MPI-I-93-123**
Fast parallel space allocation, estimation and integer sorting (revised)
Authors: Holger Bast, Torben Hagerup

- **MPI-I-93-124**
On intersection searching problems involving curved objects
Authors: Prosenjit Gupta, Ravi Janardan, Michiel Smid

- **MPI-I-93-128**
Maintaining dynamic sequences under equality-tests in polylogarithmic time
Authors: K. Mehlhorn, R. Sundar, C. Uhrig

- **MPI-I-93-129**
Tights bounds for some problems in computational geometry: the complete sub-logarithmic parallel time range
Author: Sandeep Sen

- **MPI-I-93-132**
Spectral norms and multi-party protocols
Author: Vince Grolmusz
- **MPI-I-93-138**
Routing and Sorting on Circular Arrays
Author: Jop Sibeyn
- **MPI-I-93-140**
A Complete and Efficient Algorithm for the Intersection of a General and a Convex Polyhedron
Authors: Katrin Dobrindt, Kurt Mehlhorn, Mariette Yvinec
- **MPI-I-93-142**
Mod m gates do not help on the ground floor
Author: Vince Grolmusz
- **MPI-I-93-144**
A Lower Bound for Area-Universal Graphs
Authors: Gianfranco Bilardi, Shiva Chaudhuri, Devdatt Dubhashi, Kurt Mehlhorn
- **MPI-I-93-145**
Sensitive Functions and Approximate Problems
Author: Shiva Chaudhuri
- **MPI-I-93-146**
A Lower Bound for Linear Approximate Compaction
Author: Shiva Chaudhuri
- **MPI-I-93-147**
The Complexity of Parallel Prefix Problems on Small Domains
Authors: Shiva P. Chaudhuri, Jaikumar Radhakrishnan



Part II

The Programming Logics Group



1 Members of the working group

Director

Prof. Dr. Harald Ganzinger

Research staff

David Bařın, PhD.	Andreas Nonnengart
Hubert Baumeister	Dr. Hans Jürgen Ohlbach
Dr. Alexander Bockmayr	Renate Schmidt
Detlef Fehrer	Dr. Rolf Socher
Prof. Dr. Harald Ganzinger	Jürgen Stuber
Peter Graf	Andreas Tönne
Dr. Michael Hanus	Uwe Waldmann
Jörn Hopf	Christoph Weidenbach
Ulrich Hustadt	Dr. Yong Fei Han (July 1992 - July 1993)

PhD. Students

Peter Barth
Manfred Jäger
Frank Zartmann

Post-doctoral fellows

Yannis Dimopoulos (July 1992–June 1994)
University of Athens
Viktor Kistlerov (January 1992–December 1993)
Academy of Science, Moscow
Rao Krishna (January 1994–December 1994)
Tata Institute, Bombay
Peter Madden (November 1993–October 1995)
University of Edinburgh
Seán Matthews (March 1992–February 1994)
University of Edinburgh
Emil Weydert (April 1993–August 1994)
IMS Stuttgart

Secretaries

Ellen Fries
Ellen Schreck

Guests

Leo Bachmair (Summer 1991; Summer 1992;
July 1993–July 1994)
SUNY at Stony Brook
Philippe Balbiani (March 1993–May 1993)
IRIT Toulouse
Witold Charatonik (June 1993–August 1993;
October 1993–December 1993)
University of Wroclaw
Evelyne Contejean (November 1992–October 1993)
Université de Paris Sud
Dragan Cvetkovic (Summer 1992–December 1993)
University of Yugoslavia
Dov Gabbay (July 1991–June 1995)
Imperial College, London
Darek Litwinienko (June 1993–August 1993)
University of Wroclaw
Pilar Nivela (March 1992–August 1992)
University of Barcelona
Robert Nieuwenhuis (March 1992–August 1992)
University of Barcelona
Frank Pfenning (Summer 1991)
Carnegie Mellon University, Pittsburgh
David Plaisted (July 1992; August 1993–August 1994)
University of North Carolina
Ian Pratt (October 1992–December 1992)
University of Manchester
Rosa Ruggeri (July 1992–December 1993)
University of Catania
Andrzej Szalas (August 1991–May 1992)
University of Warsaw
Jerzy Witkowski (July 1993–September 1993)
University of Wroclaw
Jan Zatopianski (May 1993–August 1993)
University of Wroclaw

2 Research programme and results

2.1 Introduction

Computer science is about how to solve abstract problems with machines. Faced with some problem, a computer scientist uses tools taken from mathematics to formalize it, and then, using that formalization, to derive a correct and efficient algorithm. This work of problem solving is difficult, so we would like to enlist computers to help us. And if mathematics is the means to solve a particular problem, then the metamathematics, i.e. mathematical logic, is the general theory of how problems are solved, and is what we use to investigate how computers might be used as general tools for problem solving. In the *programming logics* group we are looking at the following areas of logic in computer science, listed in increasing order of abstractness:

- Logic Programming
- Automated Deduction
- Program Synthesis
- Non-monotonic Reasoning and Knowledge Representation
- Logical Frameworks and the Compilation of Logics

In logic programming we study efficient techniques for compiling logic programs and for extending logic programming languages with functions, or constraint domains and solvers.

By automated deduction we mean refutational theorem proving for predicate logic. Given the significance of predicate logic, improving the state of the art in this area is of great importance. Even more so, since we believe that it is possible to compile many non-standard logics 'efficiently' into predicate logic. Despite 25 years of intensive research, the problem of automated theorem proving for predicate logic is still far from solved.

While automated deduction deals with finding solutions to goals in first-order logics, program synthesis attempts to find (semi-automatically) functions, i.e. second-order objects, that satisfy given properties. We follow the proofs-as-programs paradigm and envisage programs constructed as instantiations of unknowns in proofs of equivalence between a specification and an unknown program. Our idea is to apply logical frameworks to develop, and experiment with, various program development calculi.

Classical predicate logic is often not enough for modelling situations in a real and changing world, since we have to be able to treat incomplete knowledge and rely on assumptions that may turn out not to be true. Thus we are studying implementation techniques for well-established such (*non-monotonic*) logics, and are developing new, and more powerful, formalisms.

Research in computer science has produced a vast range of logics, from classical to intuitionistic to modal, from equational to first-order to higher-order, from monotonic to non-monotonic. Systematic methods for implementing languages on a computer, i.e. compiler construction and compiler-compilers, was an important research topic in the sixties and seventies, and in the same way we now try to find the basic principles needed to implement logics on a computer. Like for programming languages, we have two major approaches to the problem: interpretation and transformation. The first proposes the idea of a *framework* theory to formalize, execute, and reason about, logics, while the second explores ways to compile different source logics into a target logic like Horn, or predicate, logic efficiently. Both ideas are being pursued in the programming logics group.

2.2 Integrating functional and logic languages

Functional and logic programming are the most important declarative programming paradigms, and interest in combining them has grown over the last 15 years. Such integrated languages have advantages from both points of view: functional logic languages extend functional languages with facilities like function inversion, partial data structures and logical variables and, since functions evaluate deterministically, provide logic programming languages with more efficient operational behavior. Further, logic programming can avoid some of the impure control features of Prolog, like 'cut', if functions are available. Early research concentrated on the definition and improvement of execution principles for functional logic languages, while more recently efficient implementations of these execution principles have been developed, making the languages more practical to use.

A common way to integrate functions into logic programming languages is to define functions equationally, and in order to do this, resolution has to be extended with some form of equational (or E-) unification. If the equational theory enjoys certain properties (e.g. confluence and termination), then *narrowing* is a universal E-unification procedure. A narrowing step instantiates variables in a subterm by unification, so that the instantiated subterm can be reduced. In its original form, this is extremely inefficient, but many optimizations have been proposed. In [3] we present narrowing strategies for arbitrary confluent and terminating systems and introduce *LSE* narrowing; this is complete and improves all other strategies that are complete for arbitrary confluent and terminating systems. It is also optimal in the sense that two different LSE narrowing derivations cannot generate the same narrowing substitution.

Another optimal narrowing strategy for constructor-based, but not necessarily terminating, systems is *needed* narrowing [1], influenced by the lazy evaluation principle of functional programming. Needed narrowing extends the Huet and Lévy notion of needed reductions, is optimal for the number of distinct steps of a derivation, computes only

independent unifiers, and is efficiently implemented by pattern matching.

Another way to obtain an efficient execution principle for functional logic programs is to simplify goals to normal form between narrowing steps. In [4] we show that such a *normalizing* narrowing strategy improves the operational behavior of logic programs by translating pure logic programs into functional logic programs. In [5] we present a method for implementing the simplification process between narrowing steps in an incremental manner. In [7] we show that the idea of simplification can also be combined with the lazy evaluation principle for narrowing. This can yield a further reduction of the search space.

In [2] we introduce conditional narrowing modulo a set of conditional equations, and give a full proof of its correctness and completeness under general circumstances. This result can be seen as the theoretical foundation of a special form of constraint functional logic programming.

As functional logic languages becoming more efficient and, thus, practical, debugging facilities are needed. We propose, in [8], a debugging model for functional logic programs which combines features of debuggers for pure logic and pure functional programs.

Another way to execute functional logic programs is *Residuation*. Residuation tries to avoid nondeterministic steps when functions are being evaluated by delaying the evaluation of functions until the arguments are sufficiently instantiated. This retains the deterministic nature of functions, but is incomplete: if the variables in a delayed function call are not instantiated by the logic program, the function will never be evaluated and some logical consequences of the program are lost. In order to detect such situations at compile time, we have developed an abstract interpretation algorithm [6] that approximates the possible residuations and instantiation states of variables during program execution. If the algorithm computes an empty residuation set for a goal, a run of the program will not finish with residuations that cannot be evaluated due to insufficient instantiation of argument variables, and can be used to combine the advantages of narrowing (completeness) with residuation (efficiency).

References

- [1] S. Antoy, R. Echahed, and M. Hanus. A needed narrowing strategy. In *Proc. 21st ACM Symp. Principles of Programming Languages*, 1994. (To appear).
- [2] A. Bockmayr. Conditional narrowing modulo a set of equations. *Applicable Algebra in Engineering, Communication and Computing*, 4:147–168, 1993.
- [3] A. Bockmayr, S. Krischer, and A. Werner. Narrowing strategies for arbitrary canonical systems. "MPI-I-93-233", 1993.

- [4] M. Hanus. Improving control of logic programs by using functional logic languages. In *Proc. 4th Int. Symp. Programming Language Implementation and Logic Programming*, 1992.
- [5] M. Hanus. Incremental rewriting in narrowing derivations. In *Proc. 3rd Int. Conf. Algebraic and Logic Programming*, 1992.
- [6] M. Hanus. On the completeness of residuation. In *Proc. 1992 Joint Int. Conf. and Symp. on Logic Programming*, 1992. Also available, in an extended version, as MPI-I-92-217.
- [7] M. Hanus. Lazy unification with inductive simplification. MPI-I-93-215, 1993.
- [8] M. Hanus and B. Josephs. A debugging model for functional logic programs. In *Proc. 5th Int. Symp. Programming Language Implementation and Logic Programming*, 1993. Also available as MPI-I-93-222.

2.3 Constraint logic programming and combinatorial optimization

Constraint logic programming is one of the major recent developments in declarative programming. The idea is to combine a logic programming language similar to Prolog with a constraint solver for some area of computation, such as linear arithmetic, Boolean algebra, finite domains or lists. In [4] we have proposed a new constraint logic programming language, $CLP(\mathcal{PB})$, for logic programming with *pseudo-Boolean* constraints. Pseudo-Boolean constraints combine Boolean algebra with arithmetic: a pseudo-Boolean function is an integer-valued function of 0-1 variables and a pseudo-Boolean constraint is an equation or inequality between pseudo-Boolean functions.

With this language we can use constraint logic programming for combinatorial optimization problems [3]. Consider, for example, a knapsack problem: suppose there is a vessel with capacity w and goods g_i with weight w_i and value v_i for $i = 1, \dots, n$. We introduce 0-1 variables X_i indicating whether g_i is loaded on the vessel. The cargos that do not exceed the capacity can be determined by a constraint rule using a pseudo-Boolean constraint:

$$\begin{aligned} \text{cargo}(X_1, \dots, X_n) \quad \leftarrow \\ w_1 * X_1 + \dots + w_n * X_n \leq w. \end{aligned}$$

If we want to find a most valuable cargo, we can use a metapredicate *max* which optimizes a pseudo-Boolean function by maximizing it, subject to some constraints:

$$\begin{aligned} \text{most-valuable-cargo}(X_1, \dots, X_n) \quad \leftarrow \\ \text{max}(v_1 * X_1 + \dots + v_n * X_n, \text{cargo}(X_1, \dots, X_n)). \end{aligned}$$

Constraint logic programming in its original form does not support optimizations like this, but the problem has received a lot of attention recently.

We are looking at two approaches to handling pseudo-Boolean constraints. In the first, we are developing a symbolic constraint solver for the language $CLP(\mathcal{PB})$ [1]. The basic formulae of this solver are extended clauses $L_1 + \dots + L_n \geq d$ expressing that at least d out of n literals have to be true. We have developed a method for transforming arbitrary linear 0–1 inequalities into equivalent sets of extended clauses [2]. The solved form of a set of extended clauses is an equivalent set of prime extended clauses. Prime extended clauses are similar to prime implicants for classical clauses. Given this solved form, logical entailment and satisfiability can be easily decided.

Our second approach is based on cutting plane techniques from polyhedral combinatorics [5]. A constraint set is simplified by computing strong cutting planes for its linear relaxation. The ideal solved form is a description of the convex hull of the 0–1 solution set by a system of facet-defining inequalities. An interesting feature of this approach is that it can be extended to the case of mixed 0–1 constraints. Thus it provides the basis for a full integration of two of the most important domains in constraint logic programming: linear arithmetic and Boolean algebra.

References

- [1] P. Barth. A complete symbolic 0–1 constraint solver. In *3rd Workshop on Constraint Logic Programming*, 1993.
- [2] P. Barth. Linear 0–1 inequalities and extended clauses. In *Logic Programming and Automated Reasoning '93*, 1993.
- [3] P. Barth and A. Bockmayr. Solving 0–1 problems in $CLP(\mathcal{PB})$. In *Proc. 9th Conf. Artificial Intelligence for Applications*, 1993.
- [4] A. Bockmayr. Logic programming with pseudo-Boolean constraints. In F. Benhamou and A. Colmerauer, editors, *Constraint Logic Programming—Selected Research*. MIT Press, 1993. Also available as MPII-91-227.
- [5] A. Bockmayr. Using strong cutting planes in constraint logic programming. In *Operations Research '93*, 1993. (To appear).

2.4 Analysis of declarative programs

The advantage of functional and logic (i.e. *declarative*) programming languages is that we can state solutions to problems in a very abstract way. But efficient implementations need sophisticated compilation techniques.

Although various new compilation techniques based on abstract machines have been developed, high-performance implementations need knowledge about what the language is

being used for, which is why global analysis at compile time, and abstract interpretation, are widely believed to be important for them.

Abstract interpretation was originally developed as a formal analysis technique for imperative languages, but has been adopted for functional and logic programming. The basic idea is to guess the run-time behavior of the program by executing it with abstract values. If there are only finitely many abstract values, then we can be sure that the abstract execution terminates, and if the abstract values have a clearly defined relationship to the concrete values, the result of the abstract interpretation correctly approximates the real behavior. Therefore the results of abstract interpretation can be used to improve the compiled code.

We have used abstract interpretation to analyze the behavior of logic programs with certain extensions. In [1] we use abstract interpretation to analyze the completeness of the residuation principle for executing functional logic programs (see Section 2.2 for more details). In [2] we use abstract interpretation to analyze logic programs with nonlinear arithmetic constraints. In the constraint logic programming language $\text{CLP}(\mathcal{R})$ only linear constraints are solved as programs are run — nonlinear ones are delayed until they become linear. This method has the disadvantage that sometimes computed answers are unsatisfiable, or infinite loops occur because delayed nonlinear constraints are not satisfiable. These problems can be solved by using a more powerful constraint solver which can deal with nonlinear constraints like in $\text{RISC-CLP}(\text{Real})$. Since such powerful constraint solvers are not very efficient, we propose a compromise: we characterize a class of $\text{CLP}(\mathcal{R})$ programs for which all delayed nonlinear constraints become linear at run time. Programs belonging to this class can be safely executed with $\text{CLP}(\mathcal{R})$ while the remaining programs use a more powerful system. Currently, we are trying to develop general abstract interpretation techniques for analyzing functional logic programs. We have shown the usefulness of run-time information for the efficient implementation of functional logic programs in [3].

Along with abstract interpretation we are also looking at how type systems can be used to analyze the dynamics of declarative programs at compile time. Classical type-systems are weak this way: they have no way to analyze patterns of access to variables — information that could be used, for instance, to predict the life-time of values (and therefore to optimize ‘garbage collection’). We are exploring how Girard’s *linear* logic can make such details explicit. In [4] we present two complementary linearly typed λ -calculi and a typing algorithm, and discuss how ‘linear’ types capture the runtime behavior of declarative programs. We show that programs can have different linear types (in fact, exponentially many), depending on the evaluation strategy, lazy or eager. We plan to implement our calculi and examine the advantages of our approach in practice.

References

- [1] M. Hanus. On the completeness of residuation. In *Proc. 1992 Joint Int. Conf. and Symp. on Logic Programming*, 1992. Also available, in an extended version, as MPI-I-92-217.
- [2] M. Hanus. Analysis of nonlinear constraints in CLP(\mathcal{R}). In *Proc. 10th Int. Conf. Logic Programming*, 1993. Also available as MPI-I-92-251.
- [3] M. Hanus. Towards the global optimization of functional logic programs. In *Proc. Workshop on Global Compilation, Int. Logic Programming Symp., Vancouver*, 1993.
- [4] A. Tönne. Linear logic meets the lambda calculus. MPI-I-93-258, 1993. (To appear).

2.5 Automated deduction

2.5.1 Background

By *automated deduction* here we mean refutational theorem proving for predicate logic, especially using saturation methods. That is, we look at methods that start out from a theory and a negated hypothesis and derive inferences until a contradiction is produced; if this happens we can infer that the hypothesis is a theorem of the theory. Resolution is the classical instance of saturation-based refutational theorem proving. The particular problems we are looking at are as follows.

Local search space reduction. We want to improve inference systems so that fewer consequences can be derived from the same premises without losing completeness. A common problem is that such reductions only come at the expense of a large, sometimes unbearable, increase in complexity for single inferences.

Global search space reduction. In saturation-based refutational theorem proving inferences are calculated by forward chaining from the axioms and the negated hypothesis. This approach has the advantage that we are able to generate and reuse lemmas as we build the proof, but the problem that it is not goal-oriented: we can waste huge amounts of time computing redundant proofs that we do not need. We are trying to reduce the number of these useless proofs. Unfortunately, this is a hard problem: we can only see that proofs are redundant by looking at the state of the theorem prover as a whole, not locally.

Rewrite techniques supply tools for dealing with the problem, such as term orderings and simplification techniques. For instance, Knuth/Bendix-completion usually does not generate the complete equational theory of a set of identities, but a convergent set of rewrite rules, so that forward computation is under better control. An important

question is how we can extend these ideas to the more general case of first-order logic and devise similar rewrite techniques for theories other than equality.

Structured theories. Theories from which we want to prove theorems are usually not flat, but structured into modules, like a large piece of software. Some of these modules are for ordinary mathematical theories like the numbers, or set theory, but others have been defined for a particular computing application. An ideal theorem prover might collect together the tools of computer algebra, geometry, unification theory etc., inside the general, but inefficient, methods of saturation-based theorem proving. Combinations that have been studied include resolution with E-unification, provers that call computer algebra systems for simplifying complex algebraic formulae, and the introduction of sorts. A lot still needs to be done here: how can computer algebra techniques, such as Gröbner basis computation be combined with superposition-based calculi for equational logic? How can algebraic properties of relational structures be compiled into inference rules based on rewriting? How can constraint techniques be used to delay difficult subproblems, and for representing global information about proofs?

Efficient datastructures and algorithms. The most impressive tool for resolution-based refutational theorem proving available at the moment is the *OTTER* system, which uses a 'brute force' approach, using efficient datastructures and speed, instead of 'intelligence'.

We are looking for a different compromise between, on the one hand, efficiency on the level of single inferences, and, on the other, global redundancy reduction, so that fewer useless inferences are computed. Unfortunately such local and global restrictions of the search space are often hard to compute, e.g. AC-unification is double-exponential, subsumption is NP-complete. We need algorithms that perform well in practice, that work well with the existing techniques for storing large sets of formulae, and don't try to solve hard problems exactly, since approximative solutions are often enough.

Integration with higher-order logic. Effective, automatic theorem proving is, as far as we can see, a fantasy; but systems like *Isabelle*, and *Nuprl* allow real theorems to be proved interactively. In such systems, normally: the logic is designed to be usable by people rather than machines; proofs are *checked*, not automatically constructed; large and extensible collections of *tactics* automate parts of the work of building proofs.

We would like to find ways of integrating automatic first-order techniques into such an environment. As yet not much is known about even whether this is possible in principle, never mind how to do it.

2.5.2 Our results

First-Order Theorem Proving

Saturation methods. In [2] we present refutationally complete calculi for predicate logic in which equality is dealt with by ordered term rewriting. These calculi are parametrized by term orderings, selection functions and simplification techniques. The main result of this work is an abstract notion of redundancy that captures global redundancy of formulas and inferences for the current state of the saturation process. In particular, it allows simplification techniques to be easily tested for admissibility, so that they can be switched on or off as necessary.

These techniques are refined in [3], where we show that rewriting in substitutions that have been generated in previous inferences is not needed. In such ‘basic’ calculi the notion of redundancy is much more involved [3]. Many of these calculi, together with useful simplification techniques, have been implemented in the *SATURATE* system, developed by Nivela and Nieuwenhuis from Barcelona while they were visiting. We also have extended these methods to the case of hierarchic theories such that an arbitrary theorem prover (constraint solver) for the base theory can be utilized for the proof of base formulas [4]. There is a close relationship between consistent extensions of theories and second-order quantification, so these techniques, can also, in certain circumstances, be used to eliminate such quantifiers.

In [6] we show that these general purpose methods perform well on decidable fragments of predicate logic, and, in particular, can be turned into a decision procedure for the monadic class with equality if an appropriate setting of the three parameters is chosen. Recently we have started to look at applying similar rewrite techniques to transitive relations other than equality, e.g. orderings. We have identified commutation between two rewrite systems as the appropriate generalization of the notion of confluence which is the basis of the rewrite techniques for equational case [1]. A prototypical implementation of these ideas in the *SATURATE* system has provided us with promising experimental results.

Efficient Datastructures and Algorithms The performance of a theorem prover depends on the speed of the basic retrieval operations like finding unifiable terms.

Among the known methods for term retrieval in deduction systems *Path-Indexing* shows good performance in general. In [8], we describe an implementation of this method; The software is currently being used in various deduction systems such as *STOP* (here) and *SETHEO* (Technische Universität München). *Path-Indexing* is not, however, a perfect filter; we still have to check, by unification, the candidates it finds, and thus *Discrimination trees* and *abstraction trees*, which are perfect filters, manage sometimes to outperform it. In [9], we describe an improved version of *Path-Indexing* that provides

query trees and the path-index with clash and occur-check information, so that many more terms than with the standard method can be dismissed immediately.

Unification and Constraint Solving

Unification Recently there has been a lot of interest in equational (or E-) unification; both unification algorithms for special equational theories, such as those with associativity and commutativity, and more general methods like *narrowing*, that apply to whole classes of theories, have been developed. We have been looking at both aspects; particularly at narrowing, a general method for the class of equational theories defined by a convergent rewrite system. (There is a detailed account of this work in Section 2.2.)

General E-unification, which is not confined to theories with convergent rewrite systems, is investigated in [11]; we show that Gallier and Snyder's *Lazy Paramodulation*, and Dougherty and Johann's *Relaxed Paramodulation* systems can be improved by a stronger restriction on the applicability of *Lazy Paramodulation*.

In [12], we have investigated unification in the theory of terms with exponents; these allow certain infinite sets of terms to be finitely represented. Possible applications include logic programming and equational logic, where they can be used to avoid certain types of nontermination or divergence. Most inference systems for unification of terms with exponents, such as the algorithms of Comon or Salzer suffer from high technical complexity. In [12], a syntactic generalization of Comon's notion of terms with exponents is introduced together with a comparatively simple inference system for unification.

Constraints Constraint based approaches to automated theorem proving have become increasingly popular. Beside the development of constraint solving procedures for different applications, we have looked at the question of how to integrate constraint solving with logical inference rules such as resolution or superposition [3].

An interesting application of constraints is to orderings, which can be interpreted as recursive or lexicographic path orderings (*RPO* and *LPO*), arbitrary simplification orderings, subterm relations, or embedding relations. The known decision procedures for *LPO* and *RPO* have at least exponential complexity and it seems likely that the constraint satisfaction problem is NP-complete. In contrast, simplification orderings have a polynomial constraint satisfaction problem; [10] gives a constraint satisfaction procedure based on work by Plaisted, for simplification ordering constraints with complexity $O(n^3)$, and shows some modifications of the problem to be NP-complete.

Set constraints describe relations between sets of terms over some vocabulary, and arise naturally when, for instance, the concrete values of program variables, or type inference algorithms, are abstracted. We show in [5] that set constraints are equivalent to the monadic class. From this equivalence it follows that the satisfiability problem for

set constraints is complete for NEXPTIME; in fact the problem has a lower bound of $\text{NTIME}(c^{n/\log n})$. The relationship between set constraints and the monadic class also provides decidability and complexity results for certain practically useful extensions of set constraints, in particular ‘negative’ projections and subterm equality tests.

Order-Sorted Logics Adding sorts to a clause set has proved to be a very effective way of restricting the search space, and so improving the performance, of an automated theorem prover. One-place predicate symbols can usually be transformed into sort symbols subject to certain syntactic constraints. In [13, 7], we develop a system that allows one-place predicate symbols to be used as sort information in arbitrary clause sets. The system consists of an order sorted unification algorithm and a complete calculus for full first-order logic; as a corollary it provides an optimally efficient decision procedure for propositional horn clauses.

In [7], we present several results on unification in order-sorted logic. For quasi-linear signatures we show that unification is decidable, and that the empty-sort problem is undecidable. We also show that the problems of sort intersection, and whether a term has a given sort are, independently, undecidable. Finally we develop a sorted unification procedure that terminates for elementary, semi-linear, quasi-linear, and linear signatures.

References

- [1] L. Bachmair and H. Ganzinger. Rewriting techniques for transitive relations. MPI-I-93-249.
- [2] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 1993. (To appear). Revised version of MPI-I-91-208.
- [3] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation and superposition. *Information & Computation*. (To appear). Also available as MPI-I-93-236; an earlier version appeared in *Proc. CADE'11*.
- [4] L. Bachmair, H. Ganzinger, and U. Waldmann. Theorem proving for hierarchic first-order theories. In H. Kirchner and G. Levi, editors, *Algebraic and Logic Programming*, Springer, LNCS, 1992.
- [5] L. Bachmair, H. Ganzinger, and U. Waldmann. Set constraints are the monadic class. In *Proc. LICS'8*, 1993. Revised version of MPI-I-92-240.
- [6] L. Bachmair, H. Ganzinger, and U. Waldmann. Superposition with simplification as a decision procedure for the monadic class with equality. In *Proc. 3rd Kurt Gödel Colloquium*, 1993. Previous version available as MPI-I-93-204.

- [7] C. Weidenbach. Unification in sort theories and its applications. MPI-I-93-211, 1993.
- [8] P. Graf. Path indexing for term retrieval. MPI-I-92-237, 1992.
- [9] P. Graf. Extended path-indexing. MPI-I-93-253, 1993.
- [10] P. Johann and R. Socher. Solving ordering constraints in polynomial time. MPI-I-93-256, 1993.
- [11] R. Socher-Ambrosius. A refined transformation system for general E-unification. MPI-I-93-237, 1993.
- [12] R. Socher-Ambrosius. Unification of terms with exponents. MPI-I-93-217, 1993.
- [13] C. Weidenbach. A sorted logic using dynamic sorts. MPI-I-91-218, 1991.

2.6 Program synthesis

An important application of logic and theorem proving is program correctness or program synthesis from specification, and many logics have been developed and proposed for it. Implementing these logics is hard work; so is ensuring that they are correct and automating derivations in them. We are looking for a framework in which to address these problems; a way both to implement proposed calculi and to develop new ones.

We have shown how logical frameworks can be used to formalize program synthesis calculi. In our approach we treat a logic, like first-order logic or type theory, as a foundation on which we derive a calculus of rules for program development. We use a logical framework style proof development system that allows us to state and manipulate proof rules without having to worry about whether the logic we use is strong enough, itself, to formalize such concepts.

Normally we show that a program is correct by specifying in the programming logic the relationship between the program and its specification, then building a proof in the logic showing that the program and the specification are, in some sense, the same. However, we can also synthesize a program from a specification in a similar way: if at the start we have only a higher-order metavariable standing in for the program, rather than a complete program, then we can force this to be incrementally instantiated, using resolution, as we apply proof rules to build a complete proof. The resulting derivation provides a program as a solution for the metavariable that makes what we have built a proper proof.

This can be understood by analogy with Prolog, which 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 by a more complicated form of resolution. The problems we tackle are also complicated by the fact that we are trying to build recursive programs.

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. We have used Paulson's *Isabelle* framework to derive calculi for logic program synthesis (based on Wiggin's *Whelk* Calculus) and combinational circuit synthesis (based on Hanna's *Formal Synthesis* calculus) [1, 2]. In both cases, the formalization not only resulted in a machine checked account of the correctness of the calculus but also to simplifications and extensions. We have used these calculi to synthesize formally verified logic programs and circuit descriptions.

Our work has also addressed automation of proof this way. 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 [4, 5, 3].

References

- [1] D. Basin. IsaWhelk: Whelk interpreted in Isabelle. Submitted, 1993.
- [2] D. Basin, A. Bundy, I. Kraan, and S. Matthews. A framework for program development based on schematic proof. In *Proc. 7th Int. Workshop on Software Specification and Design*, 1993. Also available as MPI-I-93-231.
- [3] D. Basin and T. Walsh. Difference unification. In *Proc. IJCAI'93*, 1993. Also available as MPI-I-92-247.
- [4] I. Kraan, D. Basin, and A. Bundy. Logic program synthesis via proof planning. In K.K. Lau and T. Clement, editors, *Logic Program Synthesis and Transformation*. Springer, 1993. Also available as MPI-I-92-244.
- [5] I. Kraan, D. Basin, and A. Bundy. Middle-out reasoning for logic program synthesis. In *10th Int. Conf. Logic Programming*, 1993. Also available as MPI-93-231.

2.7 Transformation of Logical Systems

It is sometimes easier to prove a conjecture after it is transformed into the language of another system; but to do this we need to be sure that if the transformed version is true, then so is the original. We are interested in understanding the nature of such transformations, and in finding general methods, recipes and algorithms for supporting their development. Our main area of investigation is nonclassical logics; particularly

variants of modal logic, but we also look at others, such as intuitionistic and relevance logic. And, since we use predicate logic as a metalogic, the methods we are developing can be applied to problems of pure predicate logic as well.

Consider an example of the kind of transformations we are investigating: given the axioms

$$P(i(i(i(p, q), r), i(i(r, p), i(s, p)))) \\ P(r) \wedge P(i(r, p)) \rightarrow P(p)$$

it follows that $i(x, x)$, but the proof is quite complicated. But Łukasiewicz has shown that these two axioms axiomatize the implicational fragment of propositional logic. Once we can see this, the conjecture $P(i(x, x))$ can be transformed into the predicate logic formula $x \rightarrow x$, which is obviously true.

We have developed a procedure that, given the two axioms as input, is able to generate a model theoretic semantics for the function i , which, in this case, is precisely the semantics of propositional implication. All of the algorithm, apart one step, has been verified; this open step is syntactically trivial, but theoretically it has turned out to be central.

A number of powerful methods and partial results have been developed in the course of this investigation:

- A transformation method in the above sense, for eliminating unwanted properties; the method applied, for example, to a formula axiomatizing a reflexive and transitive relation, transforms it into an axiomatization of a relation which is no longer reflexive or transitive. (Applied to Prolog programs, this might help in eliminating loops caused by the transitivity axiom.)
- A method for eliminating second order quantifiers over predicate variables, to produce an equivalent first-order formula (provided there is one) [1]. Applications of this are, for instance, the automation of correspondence theory in modal logic, or the optimization of circumscription.
- Further transformation methods for binary relations as they occur, for example, in the translation of normal modal systems into predicate logic [3]. These methods are applied, in the *MOTEL* knowledge representation system, to transform epistemic and taxonomic information into Prolog programs.

One of the transformations we have developed [2] is of particular interest in practice, as well as in theory. Unlike the others, it results in pure predicate logic, and, afterwards, it is easy to see that certain axioms, for example irreflexivity of the binary relation, are redundant. This provides a simple proof that these properties cannot be axiomatized in a modal logic, and is a striking example of the use of the transformation of a logical

system to simplify proofs of metatheoretic properties; the technique is not restricted to (first-order multi-) modal logics. It is also useful in areas such as taxonomic reasoning and sorted logic.

We plan to integrate all these techniques into a workbench for supporting the development of transformations of logical systems. A prototype implementation of one of its parts, the quantifier elimination procedure, is already available.

References

- [1] D. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. *South African Computer Journal*, 7, 1992. Also available as MPI-I-92-213.
- [2] A. Nonnengart. First-order modal logic theorem proving and functional simulation. In *Proc. IJCAI'93*, 1993. Also available, in an extended version, as MPI-I-92-228.
- [3] H. J. Ohlbach. Translation methods for non-classical logics—an overview. *Bulletin of the Interest Group in Propositional and Predicate Logic*, 1:69–90, 1993. Also available as MPI-I-92-225.

2.8 Nonmonotonic reasoning

We usually have only incomplete, speculative information about the world, but must, nevertheless, make reasonable decisions about how to act, often to a time limit. To do this, we have to use default knowledge, i.e. reasonable assumptions about relationships in the world (*birds can fly*), in a rational way to draw plausible but preliminary conclusions (*Tweety is a bird, thus he can fly*) that can turn out, in the light of new evidence, to be wrong (*Tweety is a penguin*). Formally modeling such defaults, and the defeasible style of reasoning that goes with them, where conclusions do not necessarily grow monotonically with premises, is the task of nonmonotonic logic.

Many default formalisms, with different strengths and weaknesses, have been proposed. We can distinguish, roughly, between consistency/rule-based approaches like default logics or nonmonotonic inheritance networks, and model-preference-based approaches like circumscription or conditional logic. A lot of work has been done to compare, classify and evaluate the differences among them, and it turns out that most approaches have one or more of the following problems.

- *Lack of expressive power*: In natural language processing and user modeling reasoning about action and time, there is a need for nested default expressions in first-order contexts [5]. But classical accounts often have a propositional character, and do not even allow boolean combinations of defaults.

- *Implausibility*: Much work has been done on what reasonable rationality postulates for object-level default conditionals, or metalevel plausible inference, should look like, but proposed formalisms rarely satisfy them, especially not on both levels.
- *Lack of robustness*: Semantic-based approaches are the most appropriate, but, often, very sensitive to the addition of irrelevant information.
- *Inefficiency*: Until recently, the design, and analysis of the properties, of efficient algorithms had not received much attention.

We are looking at two complementary areas: efficient procedures that we can find, using links with other areas, and new semantic-based formalisms that combine, for particular applications, the strong points of previous approaches.

We have looked at the role of graph theory in nonmonotonic reasoning [1, 3, 2]. The idea here is to represent interactions among rules by means of a graph, a method applicable to logic programs with negation, disjunction-free default theories, and reason maintenance systems. The main advantage of this is that results from pure and algorithmic graph theory can be exploited. It also opens the way to the use of classical methods (e.g. satisfiability algorithms or linear optimization methods).

Gabbay's labelled deductive systems are a possible effective method for capturing the semantics of a reason maintenance system, which we are looking at [4]. This approach works for both justification and assumption based methods, providing the two with unifying semantics, that allows us to characterize systems as a whole, while still maintaining a distinction between the basic logic and the supporting machinery.

Another promising research area is the relationship between logic programming and non-monotonic reasoning, and possible exchanges of techniques. It is, for instance, interesting to see what the various semantics for logic programs with negation can give us when applied to default theories. We have used these connections to implement some forms of defeasible reasoning in the knowledge representation system *MOTEL* (see Section 2.10). By analogy with the Davis-Putnam procedure for checking satisfiability, we have developed a procedure for computing stable models of propositional normal disjunctive logic programs, using simplification and case analysis [6]. The procedure is presented as a set of rules that progressively transform a program, while preserving the meaning, until a set of solved forms is reached. This has been used with great success for unification problems and there is some hope that it can be extended to autoepistemic reasoning and to other semantics of logic programs like well-founded semantics.

Finally, we have proposed an extension to first-order logic with a default quantifier N , where $Nx(A(x, \dots), B(x, \dots))$ should be read as saying that any x verifying $A(x, \dots)$ normally verifies $B(x, \dots)$. This allows us to express nested first-order default knowledge. Our semantics is based on qualitative measures, offering a potential interface to probabilistic reasoning, and has a sound and complete proof theory [7]. Based on previous

research about nonmonotonic entailment for conditional theories [8], we have proposed a semantic-based plausible inference relation suitable for first-order contexts. Among its features are specificity reasoning, defeasible chaining and a defeasible deduction principle that links the object-level default conditional to the defeasible metalevel inference relation. This approach is more powerful than the traditional accounts described in the literature.

References

- [1] Y. Dimopoulos. The computational value of joint consistency. In *Proc. 1st Dutch/German Workshop on Nonmonotonic Reasoning Techniques and their Applications*, 1993. (To appear).
- [2] Y. Dimopoulos and V. Magirou. A graph theoretic approach to default logic. *Information & Computation*. (To appear).
- [3] Y. Dimopoulos, V. Magirou, and C. Papadimitriou. On kernels, defaults and even graphs. MPI-I-93-226, 1993.
- [4] D. Fehrer. A unifying framework for reason maintenance. In *Proc. ECSQARU '93*, 1993.
- [5] U. Hustadt. A multi modal logic for user modeling. (To appear).
- [6] J. Stuber. Computing stable models by program transformation. MPI-I-93-257, 1993. (To appear).
- [7] E. Weydert. Default quantifier logic: About plausible reasoning in first-order contexts. In *Proc. 1st Dutch/German Workshop on Nonmonotonic Reasoning Techniques and Applications*. (To appear).
- [8] E. Weydert. Default quantifier logic: Plausible inference for default conditionals. In *Proc. ECSQARU'93*, 1993.

2.9 Logical frameworks

Many logics, from classical to intuitionistic to modal, from algebraic to first-order to higher-order, have been proposed as tools for computer science. But their range raises a problem even before it solves any: how are we to implement them all? This is why the idea of a *logical framework* was invented, starting with the thought that 'if there are logics for reasoning about parallelism, time and knowledge, why can there not be a logic for reasoning about logics?'

Several logical frameworks have been proposed and implemented, and with most it is an easy matter to implement and use a particular logic (or *object-logic*) in the implemented framework (or *metalogic*). An implementation of an object logic in a framework theory is, essentially, a formal description of the logic itself (or *metatheory*), and we can reason about it in the same way we can reason about objects in other formalized theories. We have been looking at ways to exploit this.

One of the problems we have explored is how to add to a logic, safely. Logic implementors often would like to extend their systems with extra rules that, while theoretically unnecessary, simplify the work of building proofs, but complicated extensions, like rewriting and decision procedures, risk introducing errors, i.e. inconsistency. Our work looks at how metatheoretic extensions to a logic can be made, using the formalized metatheory to check their correctness. We have successfully done this in two different frameworks. We implemented first-order logic in the framework logic FS_0 and proved, as a metatheorem, an equivalence between two classes of formulae (for this experiment we had to formalize reasoning about bound variables and operations on them) [5]. We have also explored a different approach, taking a framework as a class of abstract data types, and looked at how to do proof theory over such a representation [1], and taking advantage of constructive logic to synthesize new proof construction procedures.

Not all the frameworks that have been proposed are suitable for this sort of work, though. Currently we are exploring one in particular, FS_0 , which is explicitly designed for doing general proof theory. For instance we show how to prove a cut elimination theorem for an object theory in [3]. We are implementing this logic, and, inside it, formalizing a general binding and substitution mechanism that we expect to work well both at the object and the framework levels. Further work here involves looking at how effectively we can develop tools that can be used generally for reasoning in metatheory, instead of being restricted to some particular object theory. Also, if we think of using a logical framework as a metatheory, however, the most obvious candidate to be reasoned about is the framework itself, since this is theory we are always, in one way or another, working with. We plan to address the practical problems of doing this for FS_0 sometime in the future, and have already looked at some of the theoretical questions [2, 4].

References

- [1] D. Basin and R. Constable. *Metalogical frameworks*. In G. Huet and G. Plotkin, editors, *Logical Environments*. Cambridge University Press, 1993. Also available as MPI-I-92-205.
- [2] S. Matthews. *Reflection in logical systems*. In *Proc. IMSA '92: Reflection and Meta-level Architecture*, 1992. Also available as MPI-I-92-250.

- [3] S. Matthews. A theory and its metatheory in FS_0 . In D. Gabbay and F. Guentner, editors, *What is a Logical System?* Oxford University Press, 1993. Also available as MPI-I-93-227.
- [4] S. Matthews and A. Simpson. Reflection using the derivability conditions. (forthcoming).
- [5] S. Matthews, A. Smaill, and D. Basin. Experience with FS_0 as a framework theory. In G. Huet and G. Plotkin, editors, *Logical Environments*. Cambridge University Press, 1993. Also available as MPI-I-92-244.

2.10 Knowledge representation using non-classical logics

Since the mid-seventies a variety of knowledge representation systems in the tradition of semantic networks and terminological logics have been proposed, the most famous being *KL-ONE*. All these systems are intended to represent general conceptual information, and they are typically used in the construction of the knowledge base of a single reasoning entity. The language they use is some subset of first-order logic with equality. Beside soundness and completeness, decidability of inferential operations in these systems is a commonly demanded feature.

Developments in natural language processing and planning very soon made it clear that classical knowledge representation systems are not expressive enough for such applications. As a result, various extensions of the traditional systems have been proposed, e.g. terminological logics that are able to represent the knowledge or the beliefs of multiple agents in one knowledge base, defaults, and dynamical changes of the knowledge base. All these extensions have been investigated independently, but no coherent theory for a terminological logic, that is able to provide all the representational and inferential facilities we have mentioned, has been developed.

We have been developing such a coherent that allows not only the knowledge or beliefs of a single agent, but a huge variety of modal operators for multiple agents, and defaults and dynamic changes of the knowledge base to be represented.

Our approach to multi-modal logics [5] is already well developed. We follow Ohlbach's suggestion [8]: to eliminate modal operators in a way that we get standard predicate logic formulae that still represent the modal semantics, using functional simulation to describe accessibility relations and roles.

An advantage of this way of doing things, beyond that we can use a first-order theorem prover to implement a modal logic theorem prover, is that it is independent of the frame axioms (i.e. the axioms describing the properties of the modal operators). However we do have to provide a tool able to translate frame axioms into the corresponding properties of the accessibility relation. *SCAN* [3] is an algorithm which offers a method

for computing these correspondences fully automatically (this work is described in more detail in Section 2.7).

The *MOTEL* system [6] is a prototypical implementation of our approach to knowledge representation for multiple agents. *MOTEL* translates modal terminological logic theories into Prolog logic programs or theories for the theorem prover *SETHEO*. As yet, *SCAN* is not integrated, but is an independent module.

For the target language, Prolog, it has been easy to implement inference facilities:

- In [4], we described 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 at the level of axioms.
- Using the correspondence between default theories and general logic programs at the semantic level, we have been able to integrate default reasoning in *MOTEL*.

As a result, *MOTEL* is, in terms of the expressiveness of the language, and the variety of inferential operations available for knowledge representation for multiple agents, one of the most advanced systems available, and we plan to extend it further; for instance, we are currently also investigating probabilistic reasoning in terminological logics [7] and this should result in an additional component of the system that can deal with uncertain knowledge. The system is currently used in the SFB-Project *PRACMA* ('Processing Arguments between Controversially Minded Actors'). The topic of *PRACMA* is the field of pragmatic dialogue processing, including a deep modeling the underlying concepts of intentions, attitudes, and argumentative behaviour. *MOTEL* has been developed to be well-suited for this purpose.

In the work above we exploit the link between *KL-ONE*-type knowledge representation and modal logic. But *KL-ONE*-type knowledge representation can also be linked to algebra. In [9, 2, 1] we show that terminological representation languages have evolved a semantics effectively described in algebras of sets and relations interacting with each other. In particular we use Boolean modules to accommodate a *KL-ONE*-type terminological language [2] and introduce a new class of algebras, called *Peirce* algebras, able to treat even more expressive languages in [9, 1].

A Peirce algebra is a two-sorted combination of a Boolean and a relation algebra, supplemented with operations (the Peirce product of Boolean modules and a cylindrification operation) through which they interact. Peirce algebras can also usefully formalise other fields in Computer Science beside knowledge representation; they can, for instance, be used in computational linguistics and program semantics. where they can provide elegant and expressive equational axiomatisations. In [10] we relate terminological representation

to the work of P. Suppes in computational linguistics who used Peirce algebras (under a different name) for his semantic analysis of a certain fragment of the English language. The work of Suppes and others provides a formal basis for finding (linguistically) adequate terminological representations for domain knowledge formulated in English, and is now directly accessible (via Peirce algebra) to users and developers working with *KL-ONE*-based systems.

References

- [1] C. Brink, K. Britz, and R. A. Schmidt. Peirce algebras. *Formal Aspects of Computing*, 1994. Also available as MPI-I-92-229.
- [2] C. Brink and R. A. Schmidt. Subsumption computed algebraically. *Computers and Mathematics with Applications*, 1992. Also in Lehmann, F., editor, *Semantic Networks in Artificial Intelligence*. Pergamon Press, 1992.
- [3] D. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. *South African Computer Journal*, 7, 1992. Also available as MPI-I-92-213.
- [4] U. Hustadt. Abductive disjunctive logic programming. In *ICLP'93, Postconf. Workshop on Abductive Reasoning*, 1993.
- [5] U. Hustadt and A. Nonnengart. Modalities in knowledge representation. In *Proc. 6th Australian Joint Conf. AI*, 1993.
- [6] U. Hustadt, A. Nonnengart, R. Schmidt, and J. Timm. *MOTEL* user manual. MPI-I-92-236, 1992.
- [7] M. Jaeger. Probabilistic reasoning in terminological logics (extended abstract). Submitted to KR'94, 1993.
- [8] H. J. Ohlbach. *A Resolution Calculus for Modal Logics*. PhD thesis, Universität Kaiserslautern, 1988.
- [9] R. A. Schmidt. Algebraic terminological representation. Master's thesis, Department of Mathematics, University of Cape Town, South Africa, 1991. Also available as MPI-I-91-216.
- [10] R. A. Schmidt. Terminological representation, natural language & relation algebra. In *Proc. GWAI-92*, 1993. Also available as MPI-I-92-246.

3 Journal and conference activities

3.1 Editorial positions

Prof. Harald Ganzinger has been an editor of *Information Processing Letters* since November 1990, and an editor of *Mathematical Systems Theory* since June 1992.

Dr. Michael Hanus became an editor of the *Electronic Journal on Functional and Logic Programming* this year.

Dr. Hans Jürgen Ohlbach became an editor of the *Bulletin of the Interest Group for Propositional and Predicate Logic* this year.

3.2 Conference positions

3.2.1 Memberships in Organizing Committees

D. Basin Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Marseille, France, Apr 1993

A. Bockmayr Workshop on Artificial Intelligence and Operations Research at the 17th German Conference on Artificial Intelligence, Berlin, Germany, Sep 1993

M. Hanus Workshop Logische Programmierung, Hagen, October 1993

H.J. Ohlbach German Workshop on Artificial Intelligence, Bonn, Germany, Aug 1992
First international conference on Temporal Logic, Bonn, Germany, 1994

3.2.2 Memberships in Program Committees

D. Basin Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Marseille, France, 1993

Third Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Abingdon, UK, 1994

Fifth International Conference on Logic Programming and Automated Reasoning, Crimea, Ukraine, 1994

Twelfth International Conference on Automated Deduction, Nancy, France, 1994

H. Ganzinger Twelfth International Conference on Automated Deduction, Nancy, France, 1994

Third Conference on Rewriting Techniques and Applications, Como, Italy, 1991

Fourth international conference on Logic Programming and Automated Reasoning, St. Petersburg, Russia, 1993

Fifth international conference on Rewriting Techniques and Applications, Montreal, Canada, 1993

Fourth International Conference on the Theory and Practice of Software Development, Orsay, France, 1993

Tenth Annual IEEE Symposium, Logic in Computer Science, Paris, 1994

International Conference on Algebraic and Logic Programming, 1994

M. Hanus International Logic Programming Symposium, Ithaca, NY, 1994

Fifth International Conference on Logic Programming and Automated Reasoning, Crimea, Ukraine, 1994

H.J. Ohlbach Fourth International Conference on Logic Programming and Automated Reasoning, St. Petersburg, Russia, 1993

Twelfth International Conference on Automated Deduction, Nancy, France, 1994

Frühjahrsschule für Künstliche Intelligenz, 1992

Eleventh International Conference on Automated Deduction, Albany, USA, 1992

Second Conference on Principles of Knowledge Representation and Reasoning, Boston, USA, 1992

German Workshop on Artificial Intelligence, Bonn, Germany, 1992

European Conference on Artificial Intelligence, Amsterdam, 1994

First International Conference on Temporal Logic, Bonn, Germany, 1994

Fourth Conference on Principles of Knowledge Representation and Reasoning, 1994

4 Teaching activities

4.1 Lectures and Seminars

If not specified, then at the University of Saarbrücken.

Key: L – Lectures, LE – Lectures and exercises, S – Seminar,
FoPra – Project class.

Winter Semester 1991/1992

Programmieren In Logik A. Bockmayr – LE

Lambda-Kalkül H. Ganzinger – L

Nichtklassische Logik H. J. Ohlbach – L

Summer Semester 1992

Logik Und Entscheidbarkeit A. Bockmayr – LE

Implementierung Logischer Programmiersprachen M. Hanus – L

Logische Grundlagen des Automatischen Beweisens

R. Socher-Ambrosius – L (at Univ. Kaiserslautern)

Lineare Logik H. Ganzinger, A. Bockmayr – S

Winter Semester 1992/1993

0-1 Optimierung A. Bockmayr – LE

Übersetzerbau H. Ganzinger, R. Wilhelm – LE

Spezielle Verfahren des Automatischen Beweisens

R. Socher-Ambrosius – L (at Univ. Kaiserslautern)

Metatheory and Extensible Systems D. Basin, S. Matthews – S

Summer Semester 1993

Termersetzungssysteme A. Bockmayr – LE

Lambda-Kalkül und Typ-Theorie H. Ganzinger, D. Basin – LE

Genetische Algorithmen und Simulated Annealing

H. J. Ohlbach – L

Nichtklassische Logiken R. Socher-Ambrosius – L

(at Univ. Kaiserslautern)

Logisch-Funktionale Programmiersprachen M. Hanus – S

Effiziente Algorithmen für Aussagenlogische Erfüllbarkeitsprobleme

H. J. Ohlbach, D. Cvetkovic, P. Graf, C. Weidenbach – FoPra

Winter Semester 1993/1994

Foundations of Program Verification Calculi

D. Basin, S. Matthews – LE

Constraintlösungstechniken A. Bockmayr – LE

Rechnergestütztes Beweisen H. Ganzinger, D. Basin – LE

Deduktion in klassischen und nichtklassischen Logiken

H. J. Ohlbach – LE (at Univ. Darmstadt)

Synthesis using Higher-Order Unification

H. Ganzinger, D. Basin – FoPra

4.2 Doctorates awarded

None.

4.3 Habilitations

H. J. Ohlbach, 1993.

M. Hanus (in progress).

4.4 Masters theses in progress

Ayari Abdelwaheb: *Program Synthesis with Higher-Order Unification* under D. Basin.

Alexander Bach: *Implementierung und Analyse einer lineartypischen funktionalen Programmiersprache* under D. Basin and A. Tönne.

Joachim Becker: *Effiziente Subsumption in Deduktionssystemen* under Peter Graf.

Jörg Becker: *Lernen von Grammatiken mit genetischen Algorithmen* under H. J. Ohlbach

Christoph Meyer: *Verteilte Hyperresolution* (begins in Dec. 1993) under Peter Graf

Thomas Schane: *Lernen von Spielstrategien mit genetischen Algorithmen* under H. J. Ohlbach.

5 Grants

5.1 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

<i>Starting date:</i>	July 24, 1992
<i>Duration:</i>	3 years
<i>Funding:</i>	ESPRIT Basic Research Action
<i>Staff at MPI f. Informatik:</i>	Peter Barth Alexander Bockmayr Harald Ganzinger Michael Hanus Uwe Waldmann

Partners

Cosytec, Paris (contact: Helmut Simonis)
DFKI, Saarbrücken (contact: Gert Smolka)
INRIA-Lorraine, Nancy (contact: Claude Kirchner)
TU München (contact: Tobias Nipkow)
CIS, Univ. München (contact: Klaus U. Schulz)
LRI, Université Paris-Sud (contact: Jean-Pierre Jouannaud)
Universidad Complutense de Madrid (contact: Mario Rodríguez-Artalejo)
Universitat Politècnica de Catalunya (contact: Fernando Orejas)

5.2 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.

The group at the MPI contributes to reaching the objectives in the area of modular combination of logics and constraint theorem proving. They try to extend the techniques they have developed in the past to modular presentations of first-order theories with respect to constrained inference systems for first-order logic with equality. This includes theoretical research on the combination of theorem provers for particular theories.

Work on theorem proving also includes the extension of the first-order theorem proving methods to formulas with quantifiers without employing skolemization so as to not add junk to given algebras. That way, theorem proving will actually mean program synthesis. In addition the group works on the notion of observability, both on the conceptual and on the proof-theoretic side.

Technical Data

<i>Starting date:</i>	1992
<i>Duration:</i>	3 years
<i>Funding:</i>	ESPRIT Basic Research Working Group
<i>Staff at MPI f. Informatik:</i>	Harald Ganzinger Hubert Baumeister

Partners

Åarhus Universitet, Prof. Dr. Peter Mosses

Universitat Politècnica de Catalunya, Barcelona, Prof. Dr. Fernando Orejas

Technische Universität Berlin, Prof. Dr. Hartmut Ehrig
 Technische Universität Braunschweig, Dr. Martin Gogolla
 Universität Bremen, Prof. Dr. Bernd Krieg-Brückner
 Technische Universität Dresden, Prof. Dr. Horst Reichel
 University of Edinburgh, Prof. Dr. Don Sannella
 Università di Genova, Prof. Dr. Egidio Astesiano
 Università degli Studi de L'Aquila, Prof. Dr. F. Parisi-Presicce
 INESC, Lisboa, Prof. Dr. Amilcar Sernadas
 Technische Universität München, Prof. Dr. Manfred Broy
 CRIN, Nancy, Dr. Hélène Kirchner, Prof. Dr. Pierre Lescanne
 University of Nijmegen, Prof. Dr. Hartmut Partsch
 University of Oslo, Prof. Dr. Ole-Johan Dahl
 Oxford Universtiy, Prof. Dr. Joseph Amadee Goguen
 CNRS - Université de Paris-Sud, Prof. Dr. Marie-Claude Gaudel
 Ecole Normale Supérieure/CNRS, Paris, Dr. Michel Bidoit
 Ludwig-Maximilians-Universität, München, Prof. Dr. Martin Wirsing
 MPI für Informatik, Saarbrücken, Prof. Dr. Harald Ganzinger

5.3 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

<i>Starting date:</i>	24 July 1992
<i>Duration:</i>	3 years (1.5 years approved so far)
<i>Funding:</i>	ESPRIT Basic Research Action 6471
<i>Staff at MPI f. Informatik:</i>	Hans Jürgen Ohlbach Christoph Weidenbach

Partners

RISC-Linz, Austria (contact: Jochen Pfalzgraph)
 Technische Hochschule Darmstadt, Germany (contact: Wolfgang Bibel)
 Universität München, Germany (contact: Bertram Fronhöfer)
 Max-Planck-Institut für Informatik Saarbrücken, Germany (contact: Hans Jürgen Ohlbach)
 ONERA-CERT, Toulouse, France (contact: Robert Demolombe)
 Institut Nationale Polytechnique de Grenoble, France (contact: Ricardo Caferra)
 Universite Paul Sabatier, Toulouse, France (contact: Luis Fariñas del Cerro)
 Universita di Torino, Italy (contact: Alberto Martelli)
 University of Oslo, Norway (contact: Andrew Jones)
 The Imperial College of Science Technology and Medicine, London, United Kingdom
 (contact: Dov Gabbay)
 ICL-International Computers LTD, London, United Kingdom (contact: Malcom Rigg).

5.4 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 losing 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 redundancy 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

<i>Starting date:</i>	1.7.1992
<i>Duration:</i>	2 years (possibly 4 years more)
<i>Funding:</i>	DFG Schwerpunkt Deduktion
<i>Staff at MPI f. Informatik:</i>	Harald Ganzinger Uwe Waldmann

Partners

The project is a part of the 'Schwerpunktprogramm Deduktion' (Az. 322698). Partners in this program include:

J. Avenhaus, K. Madlener (Universität Kaiserslautern),

W. Bibel (Technische Hochschule Darmstadt),
M. Broy, T. Nipkow (Technische Universität München),
U. Furbach (Universität Koblenz-Landau),
H. Ganzinger (Max-Planck-Institut für Informatik, Saarbrücken),
J. Gehne (Humboldt-Universität Berlin),
S. Hölldobler (Technische Hochschule Darmstadt),
S. Jaehnichen (Technische Universität Berlin),
E. Jessen, B. Fronhöfer, R. Letz (Technische Universität München),
H. Kleine Büning, B. Monien (Universität-GH Paderborn),
C. Kreitz (Technische Hochschule Darmstadt),
W. Küchlin, R. Bündgen (Universität Tübingen),
H. Leiß, J. Hudelmaier (Ludwigs-Maximilians-Universität München),
W. Menzel, W. Reif, W. Stephan (Universität Karlsruhe),
H. J. Ohlbach (Max-Planck-Institut für Informatik, Saarbrücken),
U. Petermann (Universität Leipzig),
J. Siekmann (Universität Saarbrücken),
G. Snelting (Technische Universität Braunschweig),
P. H. Schmitt (Universität Karlsruhe),
P. Schroeder-Heister, S. Keronen (Universität Tübingen),
H. Schwichtenberg (Universität München),
C. Walther (Technische Hochschule Darmstadt).

5.5 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 synthesizing 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: 3 years
Funding: BMFT
Staff at MPI f. Informatik: David Basin
Harald Ganzinger
Ullrich Hustadt
Hans Jürgen Ohlbach
Andreas Tönne

5.6 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 will address 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 (with possible 1 year extension)
Grant: DAAD/British Council Academic Research Collaboration Grant

Partners (group leaders)

Dr. David Basin, MPI für Informatik, Saarbücken
Prof. Alan Bundy, University of Edinburgh
Dr. Dieter Hutter, Universität Saarbrücken
Dr. Andrew Stevens, Oxford University
Prof. Christoph Walther, University of Darmstadt

5.7 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.

The Max-Planck-Institute is involved in this project as a subcontractor of the University of Aix-Marseille (A. Colmerauer) and develops efficient techniques for handling pseudo-Boolean constraints.

Technical Data

<i>Starting date:</i>	1 September 1992
<i>Duration:</i>	3 years
<i>Fundation:</i>	ESPRIT Basic Research Action 7195
<i>Staff at MPI f. Informatik:</i>	Peter Barth Alexander Bockmayr

Partners

Swedish Institute of Computer Science (SICS), S. Haridi

DEC Paris Research Laboratory (PRL), H. Aït-Kaci

Deutsches Forschungsinstitut für Künstliche Intelligenz (DFKI), G. Smolka

Institut National de Recherche en Informatique et en Automatique (INRIA), P. Codognet
 Katholieke Universiteit Leuven (KUL), B. Demoen
 Universitat Politécnica de Madrid (UPM), M. Hermenegildo
 Università di Pisa, U. Montanari
 Université d'Aix-Marseille II (UML), A. Colmerauer
 Research Institute for Symbolic Computation (RISC), H. Hong

5.8 MInd IndUS Collaboration on Proof by Mathematical Induction

Description

Grant provided travel funds for participants to hold US/European workshops on inductive theorem proving.

Technical Data

<i>Starting date:</i>	1 June 1992
<i>Duration:</i>	1 year
<i>Fundation:</i>	Esprit/NSF Cooperative Grant
<i>Staff at MPI f. Informatik:</i>	David Basin Harald Ganzinger Sean Matthews

Partners

Partners and group leaders of the MInd Consortium

University of Cambridge, M. Gordon
 Universität Darmstadt, C. Walther
 DFKI Saarbrücken, J. Siekmann
 University of Edinburgh, A. Bundy
 Universität Kaiserslautern, K. Madlener
 INRIA, Nancy, M. Rusinowitch
 Universität Tübingen, W. Kuechlin
 University of Oxford, J. Goguen

Partners and group leaders of the IndUS Consortium

University of Texas at Austin, B. Boyer
 University of Illinois, N. Dershowitz
 University of Rensselaer, NY, D. Musser
 SUNY at Albany, D. Kapur
 SUNY at Stony Brook, J. Hsiang

Ohio State University, W. Kuechlin
 MIT, D. McAllester
 Stanford University, R. Waldinger
 Cornell University, R. L. Constable
 University of North Carolina, D. Plaisted
 University of Iowa, H. Zhang

5.9 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

<i>Starting date:</i>	January 1993
<i>Duration:</i>	3 years
<i>Fundation:</i>	PROCOPE Programme, Deutscher Akademischer Austauschdienst
<i>Staff at MPI f. Informatik:</i>	Hans Jürgen Ohlbach Andreas Nonnengart Ullrich Hustadt Christoph Weidenbach Renate Schmidt Emil Weydert

Partners

Université Paul Sabatier, Toulouse
 Institut de Recherche CNRS (contact: Luis Fariñas del Cerro)

5.10 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 (Uni Saarbrücken) and DISCOUNT (Uni Kaiserslautern).

Technical Data

<i>Starting date:</i>	1 June 1992
<i>Duration:</i>	3 years (possibly 1 year more)
<i>Fundation:</i>	DFG Schwerpunkt Deduktion
<i>Staff at MPI f. Informatik:</i>	Hans Jürgen Ohlbach Peter Graf

Partners

The project is a part of the 'Schwerpunktprogramm Deduktion' (Az. 322698). Partners in this program include:

- J. Avenhaus und K. Madlener, Universität Kaiserslautern
- W. Bibel, Technische Hochschule Darmstadt
- M. Broy und T. Nipkow, Technische Universität München
- U. Furbach, Universität Koblenz-Landau
- H. Ganzinger, Max-Planck-Institut für Informatik, Saarbrücken
- J. Gehne, Humboldt-Universität Berlin
- S. Hölldobler, Technische Hochschule Darmstadt
- S. Jaehnichen, Technische Universität Berlin
- E. Jessen, B. Fronhöfer und R. Letz, Technische Universität München
- H. Kleine Büning und B. Monien, Universität Paderborn
- C. Kreitz, Technische Hochschule Darmstadt
- W. Küchlin und R. Bündgen, Universität Tübingen
- H. Leiß und J. Hudelmaier, Ludwig-Maximilians-Universität München
- W. Menzel, W. Reif und W. Stephan, Universität Karlsruhe

U. Petermann, Universität Leipzig
J. Siekmann, Universität des Saarlandes, Saarbrücken
G. Snelting, Technische Universität Braunschweig
P. H. Schmitt, Universität Karlsruhe
P. Schroeder-Heister und S. Keronen, Universität Tübingen
H. Schwichtenberg, Universität München
C. Walther, Technische Hochschule Darmstadt

5.11 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

<i>Starting date:</i>	September 1, 1991
<i>Duration:</i>	3 years
<i>Funding:</i>	German Ministry for Research and Technology (BMFT)
<i>Staff at MPI f. Informatik:</i>	Hubert Baumeister Alexander Bockmayr Harald Ganzinger Michael Hanus Rolf Socher Jürgen Stuber

6 Recent Publications

S. ANTOY, R. ECHAHED AND M. HANUS, 1993. A Needed Narrowing Strategy. Technical Report MPI-I-93-243, Max-Planck-Institut für Informatik, Saarbrücken. Short version to appear in Proc. 21st ACM Symposium on Principles of Programming Languages.

cf. section II.2.2, page 82

L. BACHMAIR AND H. GANZINGER, 1991. Completion of First-Order Clauses with Equality by Strict Superposition. In St. Kaplan, M. Okada, editors, *Proc. 2nd Int'l Workshop on Conditional and Typed Rewriting*, Montreal, Lecture Notes in Computer Science, vol. 516, pp. 162–180, Berlin. Springer-Verlag.

Abstract

We have previously shown that strict superposition together with merging paramodulation is refutationally complete for first-order clauses with equality. This paper improves these results by considering a more powerful framework for simplification and elimination of clauses. The framework gives general criteria under which simplification and elimination do not destroy the refutation completeness of the superposition calculus. One application is a proof of the refutation completeness for alternative superposition strategies with arbitrary selection functions for negative literals. With these powerful simplification mechanisms it is often possible to compute the closure of nontrivial sets of clauses under superposition in a finite number of steps. Refutation or solving of *goals* for such closed or *complete* sets of clauses is simpler than for arbitrary sets of clauses. The results in this paper contain as special cases or generalize many known results about ordered Knuth-Bendix-like completion of equations, of Horn clauses, of Horn clauses over built-in Booleans, about completion of first-order clauses by clausal rewriting, and inductive theorem proving for Horn clauses.

L. BACHMAIR AND H. GANZINGER, 1991. Perfect Model Semantics for Logic Programs with Equality. In *Proc. International Conference on Logic Programming '91*, pp. 645–659. MIT Press.

Abstract

We develop a perfect model semantics for logic programs with negation and equality. Our approach is based on ordered rewriting, a fundamental technique

used in equational programming. A logic program in our sense is a set of first-order clauses with equality together with a well-founded ordering on terms and atoms. We show that any consistent logic program has a unique perfect model, provided the ordering is total on ground expressions. The key to this result is a notion of saturation of a set of formulas (under certain inference rules) together with a related concept of redundancy. Our techniques can be applied to Prolog-programs (without equality), in which case a class of programs can be characterized via the notion of stratification up to redundancy for which unique perfect models exist. This extends previous results on (local and weak) stratification.

L. BACHMAIR AND H. GANZINGER, 1991. Rewrite-Based Equational Theorem Proving with Selection and Simplification. Technical Report MPI-I-91-208, Max-Planck-Institut für Informatik, Saarbrücken. Revised version to appear in the Journal of Logic and Computation.

cf. section II.2.5, page 89

L. BACHMAIR AND H. GANZINGER, 1992. Non-Clausal Resolution and Superposition with Selection and Redundancy Criteria. In A. Voronkov, editor, *Logic Programming and Automated Reasoning*, Lecture Notes in Computer Science, vol. 624, pp. 273–284, Berlin. Springer-Verlag.

Abstract

We extend previous results about resolution and superposition with ordering constraints and selection functions to the case of general (quantifier-free) first-order formulas with equality. The refutation completeness of our calculi is compatible with a general and powerful redundancy criterion which includes most (if not all) techniques for simplifying and deleting formulas. The spectrum of first-order theorem proving techniques covered by our results includes ordered resolution, positive resolution, hyper-resolution, semantic resolution, set-of-support resolution, and Knuth/Bendix completion, as well as their extension to general first-order formulas. An additional feature in the latter case is our efficient handling of equivalences as equalities on the level of formulas. Furthermore, our approach applies to constraint theorem proving, including constrained resolution and theory resolution.

L. BACHMAIR AND H. GANZINGER, 1993. Associative-Commutative Superposition. Technical Report MPI-I-93-250, Max-Planck-Institut für Informatik, Saarbrücken. To appear.

L. BACHMAIR AND H. GANZINGER, 1993. Ordered Chaining for Total Orderings. Technical Report MPI-I-93-251, Max-Planck-Institut für Informatik, Saarbrücken. To appear.

L. BACHMAIR AND H. GANZINGER, 1993. Rewrite-Based Equational Theorem Proving with Selection and Simplification. *Journal of Logic and Computation*. Revised version of Technical Report MPI-I-91-208. To appear.

cf. section II.2.5, page 89

L. BACHMAIR AND H. GANZINGER, 1993. Rewriting Techniques for Transitive Relations. Technical Report MPI-I-93-249, Max-Planck-Institut für Informatik, Saarbrücken.

cf. section II.2.5, page 89

L. BACHMAIR, H. GANZINGER, CHR. LYNCH AND W. SNYDER, 1992. Basic Paramodulation and Superposition. In D. Kapur, editor, *Automated Deduction — CADE'11*, Lecture Notes in Computer Science, vol. 607, pp. 462–476, Berlin. Springer-Verlag.

cf. section II.2.5, page 89

L. BACHMAIR, H. GANZINGER, CHR. LYNCH AND W. SNYDER, 1993. Basic Paramodulation. Technical Report MPI-I-93-236, Max-Planck-Institut für Informatik, Saarbrücken. To appear in *Information and Computation*.

Abstract

We introduce a class of restrictions for the ordered paramodulation and superposition calculi (inspired by the *basic* strategy for narrowing), in which paramodulation inferences are forbidden at terms introduced by substitutions from previous inference steps. In addition we introduce restrictions based on term selection rules and redex orderings, which are general criteria for delimiting the terms which are available for inferences. These refinements are compatible with standard ordering restrictions and are complete without paramodulation into variables or using functional reflexivity axioms. We prove refutational completeness in the context of deletion rules, such as simplification by rewriting (demodulation) and subsumption, and of techniques for eliminating redundant inferences.

L. BACHMAIR, H. GANZINGER AND U. WALDMANN, 1992. Theorem Proving for Hierarchic First-Order Theories. In H. Kirchner, G. Levi, editors, *Algebraic and Logic Programming*, Lecture Notes in Computer Science, vol. 632, pp. 420–445, Berlin. Springer-Verlag. Revised version to appear in *AAECC*.

cf. section II.2.5, page 89

L. BACHMAIR, H. GANZINGER AND U. WALDMANN, 1993. Set Constraints are the Monadic Class. In *Proc. 8th IEEE Symposium on Logic in Computer Science*, pp. 75–85.

cf. section II.2.5, page 90

L. BACHMAIR, H. GANZINGER AND U. WALDMANN, 1993. Superposition with Simplification as a Decision Procedure for the Monadic Class with Equality. In G. Gottlob, A. Leitsch, D. Mundici, editors, *Proc. of Third Kurt Gödel Colloquium, KGC'93*, Lecture Notes in Computer Science, vol. 713, pp. 83–96, Berlin. Springer-Verlag. Revised version of Technical Report MPI-I-93-204.

cf. section II.2.5, page 89

R. BARNETT, D. A. BASIN AND J. HESKETH, 1992. A Recursion Planning Analysis of Inductive Completion. Technical Report MPI-I-92-230, Max-Planck-Institut für Informatik, Saarbrücken. To appear in the *Annals of Artificial Intelligence and Mathematics*, no. 3–4, vol. 8, 1993.

Abstract

We use the AI proof planning techniques of recursion analysis and rippling as tools to analyze so called inductionless induction proof techniques. Recursion analysis chooses induction schemas and variables and rippling controls rewriting in explicit induction proofs. They provide a basis for explaining the success and failure of inductionless induction both in deduction of critical pairs and in their simplification. Furthermore, these explicit induction techniques motivate and provide insight into advancements in inductive completion algorithms and suggest directions for further improvements. Our study includes an experimental comparison of Clam, an explicit induction theorem prover, with an implementation of Huet and Hullot's inductionless induction.

P. BARTH, 1992. $CLP(\mathcal{PB})$: A Meta-Interpreter in $CLP(\mathcal{R})$. Technical Report MPI-I-92-233, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

Constraint logic programming is one of the most attractive research areas in logic programming. Due to (J. Jaffar, 1987) the theoretical foundation of a general constraint logic programming language scheme $CLP(\mathcal{X})$ is available. Unfortunately, implementing a $CLP(\mathcal{X})$ system for some domain \mathcal{X} is a difficult task. The problematic points are providing a *constraint solver* and ensuring the *incrementality* of the constraint system. We propose here to use an existing CLP system as implementation environment for a new CLP language. We

show that under certain conditions we can use the given constraint solver as constraint solver for the new CLP-language. We focus here on prototyping $CLP(\mathcal{PB})$, where \mathcal{PB} denotes the structure of pseudo-Boolean functions, in $CLP(\mathcal{R})$, where \mathcal{R} denotes the structures of real numbers.

P. BARTH, 1993. A Complete Symbolic 0-1 Constraint Solver. In F. Benhamou, A. Colmerauer, G. Smolka, editors, *3rd Workshop on Constraint Logic Programming, WCLP '93*.

cf. section II.2.3, page 85

P. BARTH, 1993. Linear 0-1 Inequalities and Extended Clauses. In A. Voronkov, editor, *Proc. 4th Intern. Conference on Logic Programming and Automated Reasoning LPAR '93*, Lecture Notes in Computer Science, vol. 698, pp. 40–51. Springer-Verlag. Extended version to appear in *Operations Research '93*, 18th Symposium on Operations Research.

Abstract

Extended clauses are the basic formulas of the 0-1 constraint solver used in the constraint logic programming language $CLP(\mathcal{PB})$. 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. The method relies on cutting planes techniques known from integer programming. We develop special redundancy criteria and can so produce the minimal number of extended clauses. We show how the algorithm can be used to replace the resolution rule in the generalized resolution algorithm for extended clauses. Furthermore the method can be used to obtain all strongest extended cover inequalities of a knapsack inequality.

P. BARTH AND A. BOCKMAYR, 1993. Solving 0-1 Problems in $CLP(\mathcal{PB})$. In *Proc. 9th Conference on Artificial Intelligence for Applications (CAIA)*, pp. 263–269. IEEE.

cf. section II.2.3, page 84

D. A. BASIN, G. M. BROWN AND M. E. LEESER, 1991. Formally Verified Synthesis of Combinational CMOS Circuits. *Integration: The Intern. Journal of VLSI Design*, Vol. 11, pp. 235–250.

Abstract

We present a system for simultaneously synthesizing and proving correct CMOS implementations of combinational circuits. Our system, developed within the Nuprl proof development system, is based on a set of transformation

rules that generate CMOS implementations from their logical specifications. Our research differs from previous work in three important ways: our rules are rigorously proven with respect to a formal transistor model, our transformation rules admit the synthesis of both pass transistor and series/parallel networks, and our implementation produces a human readable proof along with each circuit it synthesizes.

D. A. BASIN, A. BUNDY, I. KRAAN AND S. MATTHEWS, 1993. A Framework for Program Development Based on Schematic Proof. In *7th Intern. Workshop on Software Specification and Design*. To appear. Also available as Technical Report MPI-I-93-231.

cf. section II.2.6, page 93

D. A. BASIN AND R. CONSTABLE, 1993. Metalogical Frameworks. In G. Huet, G. Plotkin, editors, *Logical Environments*, pp. 1-29. Cambridge University Press. Also available as Technical Report MPI-I-92-205.

cf. section II.2.9, page 98

D. A. BASIN, F. GIUNCHIGLIA AND P. TRAVERSO, 1991. Automating Meta-Theory Creation and System Extension. In *AI*IA-91 (Italian Association for Artificial Intelligence)*, pp. 48-57. Springer-Verlag.

Abstract

In this paper we describe a first experiment with a new approach for building theorem provers that can formalize themselves, reason about themselves, and safely extend themselves with new inference procedures. Within the GETFOL system we have built a pair of functions that operate between the system's implementation and a theory about this implementation. The first function *lifts* the actual inference rules to axioms that comprise a theory of GETFOL's inference capabilities. This allows us to turn the prover upon itself whereby we may formally reason about its inference rules and derive new rules. The second function *flattens* new rules back into the underlying system. This provides a novel means of safe system self-extension and an efficient way of executing derived rules.

D. A. BASIN, R. HÄHNLE, B. FRONHÖFER, J. POSEGGA AND C. SCHWIND, 1993. Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Marseille, France, 1993. Technical Report MPI-I-93-213, Max-Planck-Institut für Informatik, Saarbrücken.

D. A. BASIN AND D. HOWE, 1991. Some Normalization Properties of Martin-Löf's Type Theory, and Applications. In *Theoretical Aspects of Computer Software*, pp. 475-494. Springer-Verlag.

Abstract

For certain kinds of applications of type theories, the faithfulness of formalization in the theory depends on intensional, or structural, properties of objects constructed in the theory. For type theories such as LF, such properties can be established via an analysis of normal forms and types. In type theories such as Nuprl or Martin-Löf's polymorphic type theory, which are much more expressive than LF, the underlying programming language is essentially untyped, and terms proved to be in types do not necessarily have normal forms. Nevertheless, it is possible to show that for Martin-Löf's type theory, and a large class of extensions of it, a sufficient kind of normalization property does in fact hold in certain well-behaved subtheories. Applications of our results include the use of the type theory as a logical framework in the manner of LF, and an extension of the *proofs-as-programs* paradigm to the synthesis of verified computer hardware. For the latter application we point out some advantages to be gained by working in a more expressive type theory.

D. A. BASIN AND S. MATTHEWS, 1993. A Conservative Extension of First-Order Logic and its Applications to Theorem Proving. In *The 13th Conference on Theoretical Computer Science and Foundations of Software Technology*, Lecture Notes in Computer Science. Springer-Verlag. To appear. Also available as Technical Report MPI-I-93-235.

Abstract

We define a weak second-order extension of first-order logic. We prove a second-order cut elimination theorem for this logic and use this to prove a conservativity and a realisability result. We give applications to formal program development and theorem proving, in particular, in modeling techniques in formal metatheory.

D. A. BASIN AND T. WALSH, 1992. Difference Matching. In D. Kapur, editor, *Proc. 11th Intern. Conference on Automated Deduction (CADE-11)*, Lecture Notes in Computer Science, vol. 607, pp. 295-309. Springer-Verlag.

Abstract

Difference matching is a generalization of first-order matching where terms are made identical not only by variable instantiation but also by structure

hiding. After matching, the hidden structure may be removed by a type of controlled rewriting, called rippling, that leaves the rest of the term unaltered. Rippling has proved highly successful in inductive theorem proving. Difference matching allows us to use rippling in other contexts, e.g., equational, inequational, and propositional reasoning. We present a difference matching algorithm, its properties, several applications, and suggest extensions.

D. A. BASIN AND T. WALSH, 1993. Difference Unification. In R. Bajcsy, editor, *Proc. 13th Intern. Joint Conference on Artificial Intelligence (IJCAI '93)*, volume 1, pp. 116–122. Morgan Kaufmann. Also available as Technical Report MPI-I-92-247.

cf. section II.2.6, page 93

H. BAUMEISTER, 1991. Unifying Initial and Loose Semantics of Parameterized Specifications in an Arbitrary Institution. In S. Abramsky, T. S. E. Maibaum, editors, *Proc. TAPSOFT '91-CAAP*, Lecture Notes in Computer Science, vol. 493, pp. 103–120. Springer-Verlag.

Abstract

In this paper we are going to present a theory of parameterized abstract datatypes as the model-theoretic level of parameterized specifications. We will show that parameterized abstract datatypes allow us to model the main approaches to the semantics of parameterized specifications, the loose approach and the free functor semantics, using the same formalism. As a consequence we obtain that, when using data constraints in a specification language, this language is able to cope with both the loose and the free functor semantics at the same time. To be independent of a specific logic this theory is developed in the context of an arbitrary institution.

H. BAUMEISTER, 1992. Parameter Passing in the Typed λ -Calculus Approach to Parameterized Specifications. Submitted for publication in *Proc. 9th WADT/4th COMPASS Workshop*.

Abstract

A problem with parameter passing in the typed λ -calculus approach to parameterized specifications is that when applying a parameterized specification to an actual parameter the part of the signature that was not already present in the formal parameter is forgotten in the result. Thus if the parameterized specification “list of elements” is applied to the specification of the natural numbers it is not allowed to write $head(l_1) + head(l_2)$ in the result, since, although $+$ is an operation on the natural numbers, it is not an operation defined

by the formal parameter of the list specification. In this paper we are going to define a parameter passing mechanism for the typed λ -calculus approach that solves this problem.

H. BERTLING, H. GANZINGER, R. SCHÄFERS, R. NIEUWENHUIS AND F. OREJAS, 1993. Completion Subsystem. In *Program Development by Specification and Transformation, The PROSPECTRA Methodology, Language Family, and System*, Lecture Notes in Computer Science, vol. 680, section 4.3, pp. 460–494. Springer-Verlag, Berlin.

Abstract

The paper describes the Knuth/Bendix-like completion subsystem of the PROSPECTRA programming environment. It consists of two independent completion procedures for conditional equations. The paper introduces some of the underlying theory and contains examples that illustrate the use of these procedures.

A. BOCKMAYR, 1992. Algebraic and Logical Aspects of Unification. In K. U. Schulz, editor, *Proc. 1st Workshop on Word Equations and Related Topics*, Lecture Notes in Computer Science, vol. 572, pp. 171–180. Springer-Verlag.

Abstract

During the last years unification theory has become an important subfield of automated reasoning and logic programming. The aim of the present paper is to relate unification theory to classical work on equation solving in algebra and mathematical logic. We show that many problems in unification theory have their counterpart in classical mathematics and illustrate by various examples how classical results can be used to answer unification-theoretic questions.

A. BOCKMAYR, 1992. Embedding OR Techniques in Constraint Logic Programming. In *Operations Research '92. 17th Symposium on Operations Research*.

A. BOCKMAYR, 1992. Model-Theoretic Aspects of Unification. In K. U. Schulz, editor, *Proc. 1st Workshop on Word Equations and Related Topics*, Lecture Notes in Computer Science, vol. 572, pp. 181–196. Springer Verlag.

Abstract

Unification is a fundamental operation in various areas of computer science, in particular in automated theorem proving and logic programming. In this paper we establish a relation between unification theory and classical model theory. We show how model-theoretic methods can be used to investigate a

generalized form of unification, namely the problem whether, given an equational theory E and a system of equations S , there is an extension of the free algebra in E in which S is solvable.

A. BOCKMAYR, 1992. A Theoretical Basis for Constraint Logic and Functional Programming. In M. Tchuente, editor, *Proc. 1st African Conference on Research in Computer Science*, volume 2, pp. 793–804. INRIA.

A. BOCKMAYR, 1993. 0-1 Constraints and 0-1 Optimization. In F. Benhamou, A. Colmerauer, G. Smolka, editors, *3rd Workshop on Constraint Logic Programming WCLP '93*.

A. BOCKMAYR, 1993. Conditional Narrowing Modulo a Set of Equations. *Applicable Algebra in Engineering, Communication and Computing*, Vol. 4, No. 3, pp. 147–168.

cf. section II.2.2, page 83

A. BOCKMAYR, 1993. Logic Programming with Pseudo-Boolean Constraints. In F. Benhamou, A. Colmerauer, editors, *Constraint Logic Programming—Selected Research*, chapter 18, pp. 327–350. MIT Press. Also available as Technical Report MPI-I-91-227.

cf. section II.2.3, page 84

A. BOCKMAYR, 1993. Using Strong Cutting Planes in Constraint Logic Programming. In *Operations Research '93, 18th Symposium on Operations Research*. To appear.

cf. section II.2.3, page 85

A. BOCKMAYR, C. BRZOSKA, P. DEUSSEN AND I. VARSEK, 1991. KA-Prolog: Erweiterungen einer logischen Programmiersprache und ihre effiziente Implementierung. *Informatik-Forschung und Entwicklung*, Vol. 6, pp. 128–140.

Abstract

Logic programming is one of the main paradigms in the area of declarative programming. Often it is identified with the programming language Prolog. In this paper we discuss a number of extensions of Prolog that have been investigated in the Sonderforschungsbereich 314 “Artificial Intelligence - Knowledge-Based Systems” at the University of Karlsruhe. On the level of unification we extend Prolog’s syntactical unification to order-sorted and Boolean unification, on the level of resolution we generalize Prolog’s Horn clauses and SLD-Resolution to conditional equations and conditional narrowing. In addition to language extensions themselves we present also methods and tools for their efficient implementation.

A. BOCKMAYR AND F. J. RADERMACHER (EDITORS), 1993. Künstliche Intelligenz und Operations Research (Workshop, Berlin, 13. - 14. September 1993) — Extended Abstracts. Technical Report MPI-I-93-234, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

Zwischen Künstlicher Intelligenz und Operations Research bestehen zahlreiche Querverbindungen, die bisher nicht die Beachtung gefunden haben, die sie eigentlich verdienen. Beide Disziplinen haben eine Fülle von Fragestellungen und Anwendungsgebieten gemeinsam. Dazu zählen zum Beispiel

- logische Inferenz
- Constraintlöungsverfahren
- heuristische Suche
- Entscheidungsunterstützung

und vieles andere mehr. Ergebnisse und Methoden aus dem einen Bereich können oft sehr fruchtbar in dem anderen eingesetzt werden. Der Workshop auf der 17. Fachtagung für Künstliche Intelligenz KI '93 in Berlin soll dazu beitragen, von der KI aus eine Brücke zum Operations Research zu schlagen. Sein Ziel ist es, den Austausch zwischen beiden Gebieten zu fördern und Einsatzmöglichkeiten von Begriffen und Methoden der Künstlichen Intelligenz im Operations Research und umgekehrt aufzuzeigen.

A. BOCKMAYR, S. KRISCHER AND A. WERNER, 1993. Narrowing Strategies for Arbitrary Canonical Rewrite Systems. Technical Report MPI-I-93-233, Max-Planck-Institut für Informatik, Saarbrücken. Previous version in Proc. 3rd Intern. Workshop on Conditional Term Rewriting Systems, CTRS-92, editors: M. Rusinowitch and J.-L. Rémy, vol. 656 of Lecture Notes in Computer Science, pp. 483–497, Springer-Verlag.

cf. section II.2.2, page 82

A. BOCKMAYR AND F. J. RADERMACHER, 1993. Künstliche Intelligenz und Operations Research. In *Grundlagen und Anwendungen der Künstlichen Intelligenz. 17. Fachtagung für Künstliche Intelligenz*, Informatik Aktuell, pp. 249–254. Springer-Verlag.

C. BRINK, K. BRITZ AND R. A. SCHMIDT, 1994. Peirce Algebras. *Formal Aspects of Computing*, Vol. 6, pp. 1–20. To appear. Also available as Technical Report MPI-I-92-229. An extended abstract will appear in Algebraic Methodology and Software Technology (AMAST '93): Proc. 3rd Intern. Conference on Algebraic Methodology and Software Technology, editors: M. Nivat, C. Rattray, C. Rus, and G. Scollo, Workshops in Computing Series, pp. 165–168, London. Springer-Verlag.

cf. section II.2.10, page 100

C. BRINK, D. GABBAY AND H. J. OHLBACH, 1993. Towards Automating Duality. *Journal of Computers and Mathematics with Applications*, Special Issue on Automated Reasoning. To appear. Also available as Technical Report MPI-I-93-220.

Abstract

Dualities between different theories occur frequently in mathematics and logic - between syntax and semantics of a logic, between structures and power structures, between relations and relational algebras, to name just a few. In this paper we show for the case of structures and power structures how corresponding properties of the two related structures can be computed fully automatically by means of quantifier elimination algorithms and predicate logic theorem provers. We illustrate the method with a large number of examples and we give enough technical hints to enable the reader who has access to the OTTER theorem prover to experiment herself.

C. BRINK, I. M. REWITZKY AND R. A. SCHMIDT, 1991. Autodescriptivity: Beware! *The Computer Journal*, Vol. 34, No. 4, pp. 380-381.

Abstract

Non-classical logics, and in particular many-valued logics, are increasingly used in the study of formal aspects of computing. For example, a recent paper by P. F. Gibbins in this *Journal* presents a 3-valued propositional logic for VDM. In the use of such logics one naturally relies on earlier work done by logicians, a case in point being Gibbin's use of the concept of autodescriptivity, introduced by N. Rescher. The purpose of this Note is to sound a warning that Rescher's exposition of autodescriptivity is seriously flawed, and to clarify the autodescriptivity of the logic of VDM.

C. BRINK AND R. A. SCHMIDT, 1992. Subsumption Computed Algebraically. *Computers and Mathematics with Applications*, Vol. 23, No. 2-5, pp. 329-342. Also available as Technical Report TR-ARP-3/90, Automated Reasoning Project, Research School of Social Sciences, Australian National University, Canberra, Australia.

cf. section II.2.10, page 100

J. CUNNINGHAM, D. M. GABBAY AND H. J. OHLBACH, 1991. Towards the MED-LAR Framework. In *ESPRIT '91 Conference Proc.*, pp. 822-841, Directorate-General Telecommunications, Information Industries and Innovation. Commission of the European Communities.

Abstract

This is an outline description of work seeking an integrated framework for mechanising nonclassical logics. The particular logics and calculi we are concerned with are structured from the point of view of applications. As a first example for testing our prototype of the general framework, a generalised interpretation of modal logics is presented. Next, we introduce the methodology of *Labelled Deductive Systems*, demonstrating why this approach for a general framework is adequate to integrate various logical systems via a unified methodology. Finally, the need for different operational methods of solving problems in formal logic are briefly discussed in the context of an ambitious example suggested from MEDLAR case studies.

Y. DIMOPOULOS, 1993. The Computational Value of Joint Consistency. In *Proc. 1st Dutch/German Workshop on Nonmonotonic Reasoning Techniques and their Applications*. To appear.

cf. section II.2.8, page 96

Y. DIMOPOULOS, 1993. A Graph Theoretic Approach to Default Logic. *Information & Computation*. To appear.

cf. section II.2.8, page 96

Y. DIMOPOULOS, V. MAGIROU AND C. PAPADIMITRIOU, 1993. On Kernels, Defaults and Even Graphs. Technical Report MPI-I-93-226, Max-Planck-Institut für Informatik, Saarbrücken.

cf. section II.2.8, page 96

H. J. OHLBACH (EDITOR), 1992. Preprints of Proc. of GWAI-92. Technical Report MPI-I-92-232, Max-Planck-Institut für Informatik, Saarbrücken.

N. EISINGER, A. NONNENGART AND A. PRÄCKLEIN, 1992. Termersetzungssysteme. In K. H. Bläsius, H.-J. Bürckert, editors, *Deduktionssysteme*, chapter III.4, pp. 126–149. Verlag Oldenbourg, 2nd edition.

Abstract

Dieses Kapitel dient zur Einführung in das Gebiet der Termersetzungssysteme. Besonderes Augenmerk ist dabei darauf gerichtet, Lösungsverfahren von Gleichheitsproblemen mithilfe von Termersetzungssystemen zu motivieren. Dabei wird nicht nur auf die klassische Vervollständigungsmethode, sondern

auch auf erste Erweiterungen eingegangen, welche es erlauben, das Knuth-Bendix-Verfahren sowohl als Beweisprozedur für Klauselmengen als auch als Induktionsbeweiser zu verwenden.

N. EISINGER AND H. J. OHLBACH, 1993. Deduction Systems Based on Resolution. In D. M. Gabbay, C. J. Hogger, J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume I, Logical Foundations, pp. 184–271. Oxford University Press. Also available as Technical Report MPI-I-91-217.

Abstract

A general theory of deduction systems is presented. The theory is illustrated with deduction systems based on the resolution calculus, in particular with clause graphs. This theory distinguishes four constituents of a deduction system:

- the logic, which establishes a notion of semantic entailment;
- the calculus, whose rules of inference provide the syntactic counterpart of entailment;
- the logical state transition system, which determines the representation of formulae or sets of formulae together with their interrelationships, and also may allow additional operations reducing the search space;
- the control, which comprises the criteria used to choose the most promising from among all applicable inference steps.

Much of the standard material on resolution is presented in this framework. For the last two levels many alternatives are discussed. Appropriately adjusted notions of soundness, completeness, confluence, and Noetherianness are introduced in order to characterize the properties of particular deduction systems. For more complex deduction systems, where logical and topological phenomena interleave, such properties can be far from obvious.

N. EISINGER, H. J. OHLBACH AND A. PRÄCKLEIN, 1991. Reduction Rules for Resolution Based Systems. *Artificial Intelligence*, Vol. 50, pp. 141–181.

Abstract

Inference rules for resolution based systems can be classified into deduction rules, which add new objects, and reduction rules, which remove objects. Traditional reduction rules like subsumption do not actively contribute to a solution, but they help to avoid redundancies in the search space. We present a number of advanced reduction rules, which can cope with high degrees of

redundancy and play a distinctly active part because they find trivial solutions on their own and thus relieve the control component for the deduction rules from low level tasks. We describe how these reduction rules can be implemented with reasonable efficiency in a clause graph resolution system, but they are not restricted to this particular representation.

D. FEHRER, 1993. A Unifying Framework for Reason Maintenance. In M. Clarke, R. Kruse, S. Moral, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty: Proc. Europ. Conference ECSQARU '93*, Lecture Notes in Computer Science, vol. 747, pp. 113–120. Springer-Verlag.

cf. section II.2.8, page 96

I. FRANK, D. A. BASIN AND A. BUNDY, 1992. Finesse: An Adaptation of Proof-Planning to Declarer Play in Bridge. In B. Neuman, editor, *Proc. 10th Europ. Conference on Artificial Intelligence (ECAI-92)*, pp. 72–76. Wiley.

Abstract

We present FINESSE, a system that forms optimal plans for declarer play in the game of Bridge. FINESSE adapts the technique of proof-planning, developed at Edinburgh University in the context of mathematical theorem-proving, to deal with the disjunctive choice encountered when planning under uncertainty, and the context-dependency of actions produced by the presence of an opposition. FINESSE not only demonstrated how the idea of proof-planning could be generalised, but also proved to be a very capable bridge system. In its domain of planning for individual suits, it correctly identified the proper lines of play found in many examples from the Bridge literature, supporting its decisions with probabilistic and qualitative information. Cases were even discovered in which FINESSE revealed errors in the analyses presented by recognized authorities.

D. M. GABBAY, 1992. Temporal Logic: Mathematical Foundations. Technical Report MPI-I-92-213, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

Draft version of the first six chapters of a book which is attempting to supply a comprehensive coverage of the mathematical and computational aspects of temporal logic. The first chapter introduces temporal logic and gives a fairly detailed preview of the issues which will be covered in the rest of the whole book. These include expressive power, fixed point temporal languages and applications in computing. Chapter 2 develops the basic idea of a language built

from connectives whose semantics is appropriate to some class of underlying “models” of time: for example linear or branching time. Chapter 3 introduces Hilbert style axiomatizations of such logics and contains some simple completeness proofs. The incomplete chapter 4 considers the generally incomplete predicate temporal languages and gives examples of some of the variety of choices of language here. In Chapter 5 we debate the merits of using classical first order logic to talk about temporal structures from the “outside” instead of using temporal languages “inside” the structure. We also consider the possibility of using temporal logic itself as a metalanguage. Finally, in chapter 6 we present a general theory of axiomatization of temporal logics. This examines and uses the irreflexivity rule of Gabbay to provide very general techniques.

D. M. GABBAY, 1993. Classical vs Non-Classical Logics, The Universality of Classical Logic. Technical Report MPI-I-93-230, Max-Planck-Institut für Informatik, Saarbrücken. This paper will be published in vol. 2 of the Handbook of Logic in Artificial Intelligence and Logic Programming, Oxford University Press.

Abstract

This report investigates the question of the universality of classical logic. The approach is to show that an almost arbitrary logical system can be translated reasonably intuitively and almost automatically into classical logic. The path leading to this result goes through the analysis of what is a reasonable logic, how to find semantics for it, how to build a labelled deductive system (LDS) for it, how to translate a LDS into classical logic and how to automate the process using SCAN.

D. M. GABBAY, I. M. HODKINSON AND M. A. REYNOLDS, 1992. Temporal Logic: Mathematical Foundations, Part 2. Technical Report MPI-I-92-242, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

This is a preliminary version of chapters 7-13 of volume one of the book “Temporal Logic: Mathematical Foundations and Computational Aspects” which is an attempt to provide a comprehensive coverage of temporal logic as a topic which generates problems of general mathematical interest, which has many practical applications to computer science and linguistics and which is a source of various complex problems of computation and implementation. The report is the continuation of MPI Report MPI-I-92-213 which contains draft versions of chapters 1-6. Chapter 14 on temporalisation and 15 on decidability

as well as perhaps volume two will appear in future reports. In Chapter 7, by looking at some specific examples of axiomatisation and expressive completeness of two and three dimensional logics, we illustrate some of the technical issues involved with many-dimensional logics. Chapter 8 introduces the idea of propositional quantifiers in temporal logic and examines the properties of a very useful fixed point language. In chapter 9 we show how the property of separation is related to expressive completeness so that in chapter 10 we can infer the expressive completeness of languages with until and since over various classes of flows of time. In chapter 11 we use separation to prove the expressive completeness of the Stavi connectives over the class of linear flows. Chapter 12 contains a direct proof of the same result and also considers languages appropriate to flows with "gaps" in them. Chapter 13 is a very comprehensive account of the concepts of H-dimension and the k -variable property which are both concerned with the number of bound variables needed by the monadic language to be fully expressive but have surprising connections with expressive completeness of temporal languages.

D. M. GABBAY AND H. J. OHLBACH, 1992. From a Hilbert Calculus to its Model Theoretic Semantics. In K. Broda, editor, *Proc. 4th UK Conference on Logic Programming*, pp. 218–252. Springer Workshops in Computing Series.

Abstract

There are different ways of constructing a logic. One possibility is to define a Hilbert calculus, i.e. a kind of grammar that produces all formulae to be considered true. A logic can also be defined by a model theoretic semantics for the logical connectives in the language. In this paper a general theory is presented for the transition from a Hilbert calculus to its model theoretic semantics such that soundness and completeness are automatically guaranteed. For a given Hilbert calculus we start with a general neighbourhood semantics for n -place connectives. This semantics does not impose any built-in properties. A quantifier elimination algorithm is used to translate Hilbert axioms and rules into corresponding semantic properties. By proving certain key lemmas from these semantic properties, neighbourhood semantics can be systematically strengthened up to a version of the semantics which has as many Hilbert axioms built in as possible. The work is still incomplete and will be continued.

D. M. GABBAY AND H. J. OHLBACH, 1992. Quantifier Elimination in Second-Order Predicate Logic. *South African Computer Journal*, Vol. 7, pp. 35–43. Also appeared in *Proc. 3rd Intern. Conference on Principles of Knowledge Representation and Reasoning*,

the term structure. As our method also allows to prove sufficient-completeness of function definitions in parallel with proving an inductive theorem we need not distinguish between constructors and defined functions. Our method is linear and refutationally complete with respect to the perfect model, it supports lemmas in a natural way, and it provides for powerful simplification and elimination techniques.

H. GANZINGER AND U. WALDMANN, 1992. Termination Proofs of Well-Moded Logic Programs via Conditional Rewrite Systems. In M. Rusinowitch, J.-L. Rémy, editors, *Proc. Third International Workshop on Conditional Term Rewriting Systems '92*, Lecture Notes in Computer Science, vol. 656, pp. 430–437, Berlin. Springer-Verlag.

Abstract

In this paper, it is shown that a translation from logic programs to conditional rewrite rules can be used in a straightforward way to check (semi-automatically) whether a program is terminating under the prolog selection rule.

P. GRAF, 1992. Path Indexing for Term Retrieval. Technical Report MPI-I-92-237, Max-Planck-Institut für Informatik, Saarbrücken.

cf. section II.2.5, page 89

M. HANUS, 1991. Efficient Implementation of Narrowing and Rewriting. In *Proc. Intern. Workshop on Processing Declarative Knowledge*, Lecture Notes in Artificial Intelligence, vol. 567, pp. 344–365. Springer-Verlag.

Abstract

We present an efficient implementation method for a language that amalgamates functional and logic programming styles. The operational semantics of the language consists of resolution to solve predicates and narrowing and rewriting to evaluate functional expressions. The implementation is based on an extension of the Warren Abstract Machine (WAM). This extension causes no overhead for pure logic programs and allows the execution of functional programs by narrowing and rewriting with the same efficiency as their relational equivalents. Moreover, there are many cases where functional programs are more efficiently executed than their relational equivalents.

M. HANUS, 1991. Horn Clause Programs with Polymorphic Types: Semantics and Resolution. *Theoretical Computer Science*, Vol. 89, pp. 63–106.

editors: B. Nebel, C. Rich and W. Swartout, pp. 425–435, Morgan Kaufmann, 1992. Also available as Technical Report MPI-I-92-213.

cf. section II.2.7, page 94

H. GANZINGER, 1991. A Completion Procedure for Conditional Equations. *Journal of Symbolic Computation*, Vol. 11, pp. 51–81.

Abstract

The paper presents a new completion procedure for conditional equations. The work is based on the notion of reductive conditional rewriting and the procedure has been designed to in particular handle nonreductive equations that are generated during completion. The paper also describes techniques for simplification of conditional equations and rules, so that the procedure terminates on more specifications. The correctness proofs which form a substantial part of this paper employ recursive path orderings on the proof trees of conditional equational logic, an extension of the ideas of Bachmair, Dershowitz and Hsiang to the conditional case.

H. GANZINGER, 1991. Order-Sorted Completion: The Many-Sorted Way. *Theoretical Computer Science*, Vol. 89, pp. 3–32.

Abstract

Order-sorted specifications can be transformed into equivalent many-sorted ones by using injections to implement subsort relations. In this paper we improve previous results and Meseguer about the relation between order-sorted and many-sorted rewriting. We then apply techniques for the completion of many-sorted conditional equations to systems obtained from translating order-sorted conditional equations. Emphasis will be on ways to overcome some of the problems with non-sort-decreasing rules.

H. GANZINGER AND J. STUBER, 1992. Inductive Theorem Proving by Consistency for First-Order Clauses. In J. Buchmann, H. Ganzinger, W.J. Paul, editors, *Informatik — Festschrift zum 60. Geburtstag von Günter Hotz*, pp. 441–462. Teubner-Verlag. Also in Proc. CTRS'92, LNCS 656, pp. 226–241.

Abstract

We show how the method of proof by consistency can be extended to proving properties of the perfect model of a set of first-order clauses with equality. Technically proofs by consistency will be similar to proofs by case analysis over

Abstract

This paper presents a Horn clause logic where functions and predicates are declared with polymorphic types. Types are parameterized with type variables. This leads to an ML-like polymorphic type system. A type declaration of a function or predicate restricts the possible use of this function or predicate so that only certain terms are allowed to be arguments for this function or predicate. The semantic models for polymorphic Horn clause programs are defined and a resolution method for this kind of logic programs is given. It will be shown that several optimizations in the resolution method are possible for specific kinds of programs. Moreover, it is shown that higher-order programming techniques can be applied in our framework.

M. HANUS, 1991. Parametric Order-Sorted Types in Logic Programming. In *Proc. Intern. Joint Conference on Theory and Practice of Software Development, TAPSOFT '91*, Lecture Notes in Computer Science, vol. 494, pp. 181–200. Springer-Verlag.

Abstract

This paper proposes a type system for logic programming where types are structured in two ways. Firstly, functions and predicates may be declared with types containing type parameters which are universally quantified over all types. In this case each instance of the type declaration can be used in the logic program. Secondly, types are related by subset inclusions. In this case a function or predicate can be applied to all subtypes of its declared type. While previous proposals for such type systems have strong restrictions on the subtype relation, we assume that the subtype order is specified by Horn clauses for the subtype relation \leq . This allows the declaration of a lot of interesting type structures, e.g., type constructors which are monotonic as well as anti-monotonic in their arguments. For instance, parametric order-sorted type structures for logic programs with higher-order predicates can be specified in our framework. This paper presents the declarative and operational semantics of the typed logic language. The operational semantics requires a unification procedure on well-typed terms. This unification procedure is described by a set of transformation rules which generate a set of type constraints from a given unification problem. The solvability of these type constraints is decidable for particular type structures.

M. HANUS, 1992. Improving Control of Logic Programs by Using Functional Logic Languages. In M. Bruynooghe, M. Wirsing, editors, *Proc. 4th Intern. Symposium on Programming Language Implementation and Logic Programming*, Lecture Notes in Computer Science, vol. 631, pp. 1–23. Springer-Verlag.

cf. section II.2.2, page 83

M. HANUS, 1992. Incremental Rewriting in Narrowing Derivations. In H. Kirchner, G. Levi, editors, *Proc. 3rd Intern. Conference on Algebraic and Logic Programming*, Lectures Notes in Computer Science, vol. 632, pp. 228–243. Springer-Verlag.

cf. section II.2.2, page 83

M. HANUS, 1992. Logic Programming with Type Specifications. In F. Pfenning, editor, *Types in Logic Programming*, chapter 3, pp. 91–140. MIT Press.

Abstract

In this chapter, we propose a framework for logic programming with different type systems. In this framework a typed logic program consists of a type specification and a Horn clause program which is well-typed with respect to the type specification. The type specification defines all types which can be used in the logic program. Relations between types are expressed by equations on the level of types. This permits the specification of many-sorted, order-sorted, polymorphic and polymorphically order-sorted type systems. We present the declarative semantics of our framework and two proof procedures (deduction and resolution) for typed logic programs. An interesting application is a type system that combines parametric polymorphism with order-sorted typing and permits higher-order logic programming. Moreover, our framework sheds some new light on the rôle of types in logic programming.

M. HANUS, 1992. On the Completeness of Residuation. In K. Apt, editor, *Proc. 1992 Joint Intern. Conference and Symposium on Logic Programming*, pp. 192–206. MIT Press. Extended version: Technical Report MPI-I-92-217.

cf. section II.2.2, page 83

M. HANUS, 1993. Analysis of Nonlinear Constraints in $CLP(\mathcal{R})$. In *Proc. 10th Intern. Conference on Logic Programming (ICLP '93)*, pp. 83–99. MIT Press. Also available as Technical Report MPI-I-92-251.

cf. section II.2.4, page 86

M. HANUS, 1993. Lazy Unification with Inductive Simplification. Technical Report MPI-I-93-215, Max-Planck-Institut für Informatik, Saarbrücken.

cf. section II.2.2, page 83

$S_0 = \{s_1 = t_2, \dots, s_n = t_n\}$, called a unification problem, finding a substitution σ such that $\sigma(s_i)$ and $\sigma(t_i)$ are equivalent under the conversion rules of the calculus for all i , $1 \leq i \leq n$. I present the method as a transformation system, i.e. as a set of schematic rules $U \Rightarrow U'$ such that any unification problem $\delta(U)$ can be transformed into $\delta(U')$ where δ is an instantiation of the meta-level variables in U and U' . By successive use of transformation rules one possibly obtains a solved unification problem with obvious unifier. I show that the transformation system is correct and complete, i.e. if $\delta(U) \Rightarrow \delta(U')$ is an instance of a transformation rule, then the set of all unifiers of $\delta(U')$ is a subset of the set of all unifiers of $\delta(U)$ and if \mathcal{U} is the set of all unification problems that can be obtained from successive applications of transformation rules from an unification problem U , then the union of the set of all unifiers of all unification problems in \mathcal{U} is the set of all unifiers of U . The transformation rules presented here are essentially different from those in Gallier and Snyder (1989) or Nipkow (1990). The correctness and completeness proofs are in lines with those of Gallier and Snyder (1989).

U. HUSTADT, 1992. Unification and Matching in Church's Original Lambda Calculus. Technical Report MPI-I-92-219, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

In current implementations of higher-order logics higher-order unification is used to lift the resolution principle from the first-order case to the higher-order case. Higher-order matching is the core of implementations of higher-order rewriting systems and some systems for program transformation. In this paper I argue that Church's original lambda calculus, called non-forgetful lambda calculus, is an appropriate basis for higher-order matching. I provide two correct and complete algorithms for unification in the non-forgetful lambda calculus. Finally, I show how these unification algorithms can be used for matching in the non-forgetful lambda calculus.

U. HUSTADT, 1993. Abductive Disjunctive Logic Programming. In P. Codognet, P. M. Dung, A. C. Kakas, P. Mancarella, editors, *ICLP '93 Postconference Workshop on Abductive Reasoning*.

cf. section II.2.10, page 100

U. HUSTADT, 1993. Automated Support for the Development of Non-Classical Logics. In H.-J. Bärckert, W. Nutt, editors, *Workshop: Modellierung epistemischer Propositionen, KI '93*, Berlin. To appear as Research Report of DFKI.

M. HANUS, 1993. Towards the Global Optimization of Functional Logic Programs. In *Proc. Workshop on Global Compilation, International Logic Programming Symposium*, pp. 83-97.

cf. section II.2.4, page 86

M. HANUS AND B. JOSEPHS, 1993. A Debugging Model for Functional Logic Programs. In *Proc. 5th Intern. Symposium on Programming Language Implementation and Logic Programming*, Lecture Notes in Computer Science, vol. 714, pp. 28-43. Springer. Also available as Technical Report MPI-I-93-222.

cf. section II.2.2, page 83

J. HOPF AND F. KLAWONN, 1993. Selbstlernende Fuzzy-Controller auf der Basis Genetischer Algorithmen. In *Fuzzy-Systeme '93: Management unsicherer Informationen*, pp. 21-27. Gesellschaft für Informatik.

Abstract

Up to now, with the design of Fuzzy-Controllers two main difficulties have occurred. On the one hand it is the tuning of the membership functions, on the other hand it is setting up an appropriate rule base. This paper deals with the problem of finding out the best possible rules - the basis of a Fuzzy-Controller. Consulting experts still is the usual but time-consuming and therefore rather expensive method. Besides, after having designed the controller, one cannot be sure that the rule base works at its approximative optimum. This paper shows how to reduce significantly the period of development (and the costs) of Fuzzy-Controllers with the help of Genetic Algorithms and, above all, how to engender a rule base which is very close to an optimum solution. By means of a classic controlling function of a Fuzzy-Controller the design of a Genetic Algorithm will be illustrated and the solution will then be described. So this paper does *not* deal with the tuning of an existing Fuzzy-Controller but with the genetic (re-)production of rules, even without the need for experts.

U. HUSTADT, 1992. A Complete Transformation System for Polymorphic Higher-Order Unification. In *Proc. 6th Intern. Workshop on Unification (BUCS Tech Report #93-004, Computer Science Department, Boston University)*, Schloss Dagstuhl, Germany. Also available as Technical Report MPI-I-91-228.

Abstract

Polymorphic higher-order unification is a method for unifying terms in the polymorphically typed λ -calculus, that is, given a set of pairs of terms

Abstract

The most natural means for specifying a non-classical logic is by means of a Hilbert calculus. Usually, the semantics of a non-classical logic is given in terms of possible worlds. Given an axiomatization of a non-classical logic, the *correspondence problem* in these logics is to find for every given Hilbert axiom an equivalent property of the accessibility relation (van Benthem (1984)). For mechanizing deduction in non-classical logics it is very important to find these correspondences (Ohlbach (1991)). So far the method for finding the correspondences was mostly by intuition and the verification required complex proofs (van Benthem (1984)). SCAN is an algorithm which offers a method for computing the correspondences fully automatically. Moreover, since SCAN preserves equivalences, the computed correspondence axioms are *guaranteed to be complete* in the sense that a formula is derivable in the Hilbert calculus if and only if it is valid in the frames which are models of the computed correspondence axiom. In this paper we present the SCAN algorithm and an application of it to the problem of collapsing modalities in multi-modal logics: Given a Hilbert calculus for modalities \Box_{m_1} and \Box_{m_2} we have to ensure that

$$\Box_{m_1} P \Leftrightarrow \Box_{m_2} P$$

doesn't hold for all formulae P , because this is in general an unwanted consequence of the given axiomatization.

U. HUSTADT AND A. NONNENGART, 1993. Modalities in Knowledge Representation. In *Proc. 6th Australian Joint Conference on Artificial Intelligence*. To appear.

cf. section II.2.10, page 99

U. HUSTADT, A. NONNENGART, R. SCHMIDT AND J. TIMM, 1992. MOTEL User Manual. Technical Report MPI-I-92-236, Max-Planck-Institut für Informatik.

cf. section II.2.10, page 100

MANFRED JAEGER, 1993. Circumscription: Completeness Reviewed. *Artificial Intelligence*, Vol. 60, pp. 293–301.

Abstract

In this paper we demonstrate that some results on the completeness of P -defining theories published earlier are incorrect. We point out that by restricting the original propositions to well-founded theories results somewhat weaker than the original ones can be retained. We also present a theorem that

provides some insight into the relation between completeness and reducibility and helps to identify the theories whose minimal models can be adequately handled with circumscription.

P. JOHANN AND R. SOCHER, 1993. Solving Ordering Constraints in Polynomial Time. Technical Report MPI-I-93-256, Max-Planck-Institut für Informatik, Saarbrücken.

cf. section II.2.5, page 90

I. KRAAN, D. BASIN AND A. BUNDY, 1993. Middle-Out Reasoning for Logic Program Synthesis. In *Proc. 10th Intern. Conference on Logic Programming (ICLP '93)*, pp. 441–455. MIT Press. Also available as Technical Report MPI-I-93-214.

cf. section II.2.6, page 93

I. KRAAN, D. A. BASIN AND A. BUNDY, 1993. Logic Program Synthesis via Proof Planning. In K. K. Lau, T. Clement, editors, *Logic Program Synthesis and Transformation*, pp. 1–14. Springer-Verlag. Also available as Technical Report MPI-I-92-244.

cf. section II.2.6, page 93

S. KRISCHER AND A. BOCKMAYR, 1991. Detecting Redundant Narrowing Derivations by the LSE-SL Reducibility Test. In *Proc. Rewriting Techniques and Applications*, Lecture Notes in Computer Science, vol. 488, pp. 74–85. Springer-Verlag.

Abstract

Rewriting and narrowing provide a nice theoretical framework for the integration of logic and functional programming. For practical applications however, narrowing is still much too inefficient. In this paper we show how reducibility tests can be used to detect redundant narrowing derivations. We introduce a new narrowing strategy, LSE-SL left-to-right basic normal narrowing, prove its completeness for arbitrary canonical term rewriting systems, and demonstrate how it increases the efficiency of the narrowing process.

S. MATTHEWS, 1992. Reflection in a Logical System. In A. Yonezawa, B. C. Smith, editors, *Proc. IMSA '92 Workshop on Reflection and Meta-Level Architecture*, pp. 178–183. Also available as Technical Report MPI-I-92-250.

cf. section II.2.9, page 98

S. MATTHEWS, 1992. A Theory and its Metatheory in FS_0 . In D. Gabbay, F. Guenther, editors, *What is a Logical System*. Oxford University Press. To appear. Also available as Technical Report MPI-I-93-227.

cf. section II.2.9, page 98

S. MATTHEWS, A. SMAILL AND D. A. BASIN, 1993. Experience with FS_0 as a Framework Theory. In G. Huet, G. Plotkin, editors, *Logical Environments*, pp. 61–82. Cambridge University Press. Also available as Technical Report MPI-I-92-214.

cf. section II.2.9, page 98

R. NIEUWENHUIS, 1992. A New Ordering Constraint Solving Method and its Applications. Technical Report MPI-I-92-238, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

We show that it is possible to transform any given LPO ordering constraint C into a finite equivalent set of constraints S for which a special kind of solutions can be obtained. This allows to compute the equalities that follow from ordering constraints, and to decide e.g. whether an *ordering constrained equation* is a tautology. Another application we develop here is a method to check ordered rewrite systems for (ground) confluence.

A. NONNENGART, 1992. First-Order Modal Logic Theorem Proving and Standard PROLOG. Technical Report MPI-I-92-228, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

Many attempts have been started to combine logic programming and modal logics. Most of them however, do not use classical PROLOG, but extend the PROLOG idea in order to cope with modal logic formulae directly. These approaches have the disadvantage that for each logic new logic programming systems are to be developed and the knowledge and experience gathered from PROLOG can hardly be utilized. Modal logics based on Kripke-style relational semantics, however, allow a direct translation from modal logic into first-order predicate logic by a straightforward translation of the given relational semantics. Unfortunately such a translation turns out to be rather naive as the size of formulae increases exponentially during the translation. This paper now introduces a translation method which avoids such a representational overhead. Its basic idea relies on the fact that any binary relation can be replaced by equations and inequations which (under certain circumstances) can be eliminated later on by some further transformation. The overall approach thus works essentially for any modal logic having a Kripke-style possible world semantics and first-order describable frame properties. If at all, its application as a pre-processing for PROLOG is limited merely by the possibility of having frame properties which are not Horn or not even first-order describable.

A. NONNENGART, 1993. First-Order Modal Logic Theorem Proving and Functional Simulation. In R. Bajcsy, editor, *Proc. 13th Intern. Joint Conference on Artificial Intelligence (IJCAI '93)*, volume 1, pp. 80–85. Morgan Kaufmann.

cf. section II.2.7, page 94

H. J. OHLBACH, 1991. Semantics Based Translation Methods for Modal Logics. *Journal of Logic and Computation*, Vol. 1, No. 5, pp. 691–746.

Abstract

A general framework for translating logical formulae from one logic into another logic is presented. The framework is instantiated with two different approaches to translating modal logic formulae into predicate logic. The first one, the well known relational translation makes the modal logic's possible worlds structure explicit by introducing a distinguished predicate symbol to represent the accessibility relation. In the second approach, the functional translation method, paths in the possible worlds structure are represented by compositions of functions which map worlds to accessible worlds. On the syntactic level this means that every flexible symbol is parametrized with particular terms denoting whole paths from the initial world to the actual world. The target logic for the translation is a first-order many-sorted logic with built in equality. Therefore the source logic may also be first-order many-sorted with built in equality. Furthermore flexible function symbols are allowed. The modal operators may be parametrized with arbitrary terms and particular properties of the accessibility relation may be specified within the logic itself.

H. J. OHLBACH, 1992. Logic Engineering—Konstruktion von Logiken. *KI*, Vol. 3, pp. 34–38. Special Issue on Logic.

Abstract

In diesem Beitrag wird gezeigt, wie man monotone zweiwertige Logiken mit Hilfe von Hilbertkalkülen spezifizieren und diese Spezifikation automatisch in einem Compiler transformieren kann, der Formeln dieser Logik in normale Prädikatenlogik übersetzt. Es ist damit unnötig geworden, spezielle Kalküle für diese Logiken zu entwickeln. Alle für Prädikatenlogik entwickelten Methoden – Kalküle, automatische Beweiser, logische Programmiersprachen, KL-ONE basierte Wissensrepräsentationssprachen, Defaultmechanismen und so weiter – sind dann auch für die neu definierten Logiken anwendbar. Damit steht eine Methodik zur Verfügung, um komplexe Logiken für Anwendungen in der KI maßzuschneidern, ohne die notwendigen Inferenzverfahren wieder neu erfinden zu müssen.

H. J. OHLBACH, 1993. Ein kurzes Tutorial über funktionale Übersetzung von Modallogik nach Prädikatenlogik. In Alfred Kobsa, editor, *Bericht Nr. 15/93: Recommendations for Extensions to BGP-MS*, pp. 19–26. Univ. Konstanz, FB Informationswissenschaft.

Abstract

In diesem Bericht werden die wesentlichen Punkte geklärt, die für Anwender der funktionalen Übersetzung von Modal- nach Prädikatenlogik wichtig sind.

H. J. OHLBACH, 1993. Translation Methods for Non-Classical Logics—an Overview. *Bulletin of the Interest Group in Propositional and Predicate Logics (IGPL)*, Vol. 1, No. 1, pp. 69–90. A short version appeared in Proc. LPAR '93, vol. 698 of Lecture Notes in Computer Science, pp. 253–264, Springer-Verlag. Also available as Technical Report MPI-I-93-225.

cf. section II.2.7, page 94

H. J. OHLBACH AND F. BAADER, 1993. A Multi-Dimensional Terminological Knowledge Representation Language, Preliminary Version. Technical Report MPI-I-93-212, Max-Planck-Institut für Informatik, Saarbrücken. A short version of this paper is published in Proc. of IJCAI '93, volume 1, pp. 690–695, Morgan Kaufmann.

Abstract

An extension of the concept description language *ALC* used in KL-ONE-like terminological reasoning is presented. The extension includes multi-modal operators that can either stand for the usual role quantifications or for modalities such as belief, time etc. The modal operators can be used at all levels of the concept terms, and they can be used to modify both concepts and roles. This is an instance of a new kind of combination of modal logics where the modal operators of one logic may operate directly on the operators of the other logic.

H. J. OHLBACH AND A. HERZIG, 1991. Parameter Structures for Parametrized Modal Operators. In *Proc. Intern. Conference on Artificial Intelligence '91*, pp. 512–517. Morgan Kaufmann.

Abstract

The parameters of the parametrized modal operators $[p]$ and $\diamond p$ usually represent agents (in the epistemic interpretation) or actions (in the dynamic logic interpretation) or the like. In this paper the application of the idea of

parametrized modal operators is extended in two ways: First of all a modified neighbourhood semantics is defined which permits among others the interpretation of the parameters as probability values. A formula $[.5]F$ may for example express the fact that in at least 50% of all cases (worlds) F holds. These probability values can be numbers, qualitative descriptions and even arbitrary terms. Secondly a general theory of the parameters and in particular of the characteristic operations on the parameters is developed which unifies for example the multiplication of numbers in the probabilistic interpretation of the parameters and the sequencing of actions in the dynamic logic interpretation.

H. J. OHLBACH AND A. NONNENGART, 1992. Modal- und Temporallogik. In K. H. Bläsius, H.-J. Bürckert, editors, *Deduktionssysteme*, chapter VII, pp. 239–285. Verlag Oldenbourg, 2nd edition.

Abstract

In diesem Kapitel wird Modal und Temporallogik eingeführt. Es werden verschiedene Interpretationen wie epistemische Logik, doxastische Logik, Aktionslogik usw. beschrieben. Darüberhinaus werden Übersetzungstechniken vorgestellt, mit denen man Formeln sowie Formelschemata dieser Logik in Prädikatenlogik übersetzen kann.

H. J. OHLBACH AND J. H. SIEKMANN, 1991. The Markgraf Karl Refutation Procedure. In J. L. Lassez, G. Plotkin, editors, *Computational Logic, Essays in Honor of Alan Robinson*, pp. 41–112. MIT Press.

Abstract

The goal of the *MKRP project* is the development of a theorem prover which can be used as an inference engine in various applications, in particular it should be capable of proving significant mathematical theorems. Our first implementation, the *Markgraf Karl Refutation Procedure (MKRP)* realizes some of the ideas we have developed to this end. It is a general purpose resolution based deduction system that exploits the representation of formulae as a graph (clause graph). The main features are its well tailored selection components, heuristics and control mechanisms for guiding the search for a proof. mechanisms for guiding the search for a proof. This paper gives an overview of the system. It summarizes and evaluates our experience with the system in particular, and the logics we use as well as the clause graph approach: as 1990 marks the fifteenth birthday of the system, the time may have come to ask: "Was it worth the effort?"

R. A. SCHMIDT, 1991. Algebraic Terminological Representation. Technical Report MPI-I-91-216, Max-Planck-Institut für Informatik, Saarbrücken. Also available as Thesis-Reprints TR 011, Department of Mathematics, University of Cape Town, South Africa.

cf. section II.2.10, page 100

R. A. SCHMIDT, 1992. Terminological Representation, Natural Language & Relation Algebra. In H. J. Ohlbach, editor, *Proc. 16th German Workshop on Artificial Intelligence (GWAI-92)*, Lecture Notes in Artificial Intelligence, vol. 671, pp. 357–371. Springer-Verlag. Also available as Technical Report MPI-I-92-246.

cf. section II.2.10, page 100

R. SOCHER-AMBROSIUS, 1991. Boolean Algebra Admits no Convergent Term Rewriting System. In R. V. Book, editor, *Proc. 4th Intern. Conference on Rewriting Techniques and Applications*, Lectures Notes in Computer Science, vol. 488, pp. 264–274. Springer-Verlag.

Abstract

Although there exists a normal form for the theory of Boolean Algebra w.r.t. associativity and commutativity, the so called set of prime implicants, there does not exist a convergent equational term rewriting system for the theory of Boolean Algebra modulo AC. The result seems well-known, but no formal proof exists as yet. In this paper a formal proof of this fact is given.

R. SOCHER-AMBROSIUS, 1991. On the Church-Rosser Property in Left-Linear Systems. TR 91/17, SUNY at Stony Brook.

Abstract

In this paper three critical pair conditions are given that are sufficient for a finite, left-linear, but not necessarily terminating term rewriting system to have the Church-Rosser property.

R. SOCHER-AMBROSIUS, 1991. On the Relation Between Completion Based and Resolution Based Theorem Proving. *Journal of Symbolic Computation*, Vol. 11, No. 1 & 2, pp. 129–148.

Abstract

Completion Theorem Proving, as proposed by Hsiang, is based on the observation that proving a first-order formula is equivalent to solving an equational system over a boolean polynomial ring. The latter can be accomplished by completing the set of rewrite rules obtained from the equational system. This method's basic deduction rule is the generation of a new rule from a divergent critical pair obtained by superposition of two rules. This paper relates superposition exactly once-a result which was given by R. Shostak-but admit completion refutations with this property, that is, such a completion refutation is shorter than any resolution refutation can be. Furthermore, we show by means of Shostak's theorem that the language of rings and ideals is well suited for short and elegant proofs of theorems about resolution deductions.

R. SOCHER-AMBROSIUS, 1991. Optimizing the Clausal Normal Form Transformation. *Journal of Automated Reasoning*, Vol. 7, No. 3, pp. 325-336.

Abstract

Resolution based theorem proving systems require the conversion of predicate logic formulae into clausal normal form. The multiplication from disjunctive into conjunctive forms in general produces a lot of tautologous and subsumed clauses, which is relatively hard to recognize in later stages of the proof. In this paper an algorithm is presented that avoids the generation of redundant clauses. It is based on the generation of paths through a matrix and produces the set of prime implicants of the original formula.

R. SOCHER-AMBROSIUS, 1992. Completeness of Resolution and Superposition Calculi. Technical Report MPI-I-92-224, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

We modify Bezem's (Bezem, M: Completeness of Resolution Revisited. *Theoretical Computer Science* 74 (1990) 227-237) completeness proof for ground resolution in order to deal with ordered resolution, redundancy, and equational reasoning in form of superposition. The resulting proof is completely independent of the cardinality of the set of clauses.

R. SOCHER-AMBROSIUS, 1992. A Goal Oriented Strategy Based on Completion. In H. Kirchner, G. Levi, editors, *Proc. 3rd Intern. Conference on Algebraic and Logic Programming*, Lecture Notes in Computer Science, vol. 632, pp. 435-445. Springer-Verlag. Also available as Technical Report MPI-I-92-206.

Abstract

In this paper, a paramodulation calculus for equational reasoning is presented that combines the advantages of both Knuth-Bendix completion and goal directed strategies like the set of support strategy. Its soundness and completeness is proved, and finally the practical aspects of this method are discussed.

R. SOCHER-AMBROSIUS, 1992. How to Avoid the Derivation of Redundant Clauses in Reasoning Systems. *Journal of Automated Reasoning*, Vol. 9, No. 1, pp. 325–336.

Abstract

This paper addresses two problems concerning the issue of redundant information in resolution based reasoning systems. The first one deals with the question how the derivation of redundant clauses, such as duplicates or instances of already retained clauses, can be substantially reduced. The second one asks for a criterion to decide, which clauses need not be tested for redundancy. In this paper we consider a particular kind of redundancy, which we call *ancestor subsumption*, that is the subsumption of a resolvent by one of its ancestors. We give a complete syntactic characterization of clause sets producing ancestor subsumed clauses. This characterization partially answers the two questions. First, if a clause set is known to exclude ancestor subsumption, linear resolution turns out to be a preferable strategy in order to reduce the generation of subsumed clauses. Concerning the second question, this result allows a suitable restriction of the-usually very expensive-subsumption test. Finally, we show that in particular cases those clauses that account for the occurrence of ancestor subsumption can be excluded from the resolution process. SAM's lemma will serve as an example for demonstrating various possibilities to remove redundancy-generating clauses.

R. SOCHER-AMBROSIUS, 1992. Semi-Unification. Technical Report MPI-I-92-207, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

Semi-unifiability is a generalization of both unification and matching. It is used to check nontermination of rewrite rules. In this paper an inference system is presented that decides semi-unifiability of two terms s and t and computes a semi-unifier. In contrast to an algorithm by Kapur, Musser et al, this inference system comes very close to the one for ordinary unification.

R. SOCHER-AMBROSIUS, 1993. Unification in Order-Sorted Logic with Term Declarations. In *Proc. 4th Conference on Logic Programming and Automated Reasoning (LPAR '93)*, Lecture Notes in Computer Science, vol. 698, pp. 301–308. Springer-Verlag.

Abstract

This paper provides two results concerning Order-Sorted Logic with Term Declarations. First, we show that linear term declarations can be transformed conservatively into function declarations, thus yielding elementary signatures. This provides a simple proof of the well known fact that unification in linear signatures is decidable. A similar transformation exists for semi-linear term declarations, resulting in shallow term declarations. Secondly, we provide an inference system transforming sort constraints over an arbitrary signature into almost solved form. The step from almost solved forms to solved forms requires a procedure to decide emptiness of sort intersections, which is not possible in general. This shows that it is the sort intersection problem that accounts for the undecidability of unification in signatures with term declarations.

R. SOCHER-AMBROSIUS, 1993. Unification of Terms with Exponents. Technical Report MPI-I-93-217, Max-Planck-Institut für Informatik, Saarbrücken.

cf. section II.2.5, page 90

ROLF SOCHER-AMBROSIUS, 1993. A Refined Transformation System for General E-Unification. Technical Report MPI-I-93-237, Max-Planck-Institut für Informatik, Saarbrücken.

cf. section II.2.5, page 90

A. SZALAS, 1992. On Correspondence Between Modal and Classical Logic: Automated Approach. Technical Report MPI-I-92-209, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

The current paper is devoted to automated techniques in correspondence theory. The theory we deal with concerns the problem of finding classical first-order axioms corresponding to propositional modal schemas. Given a modal schema and a semantics based method of translating propositional modal formulas into classical first-order ones, we try to derive automatically classical first-order formula characterizing precisely the class of frames validating the schema. The technique we consider can, in many cases, be easily applied even without any computer support.

Although we mainly concentrate on Kripke semantics, the technique we apply is much more general, as it is based on elimination of second-order quantifiers from formulas. We show many examples of application of the method. Those can also serve as new, automated proofs of considered correspondences.

The overloaded approach differs from the many-sorted and the non-overloaded case, in that the overloaded term algebra is not necessarily initial. We give a decidable sufficient criterion for the initiality of the term algebra, which is less restrictive than GJM-regularity as proposed by Goguen, Jouanaud, and Meseguer.

Sort decreasingness is an important property of rewrite system, since it ensures that confluence and Church-Rosser property are equivalent, that the overloaded and non-overloaded rewrite relations agree, and that variable overlaps do not yield critical pairs. We prove that it is decidable whether or not a rewrite rule is sort decreasing, even if the signature is not regular.

Finally we demonstrate that every overloaded completion procedure may also be used in the non-overloaded world, but not conversely, and that specifications exist that can only be completed using the non-overloaded semantics.

C. WEIDENBACH, 1991. A Sorted Logic Using Dynamic Sorts. Technical Report MPI-I-91-218, Max-Planck-Institut für Informatik, Saarbrücken.

cf. section II.2.5, page 91

C. WEIDENBACH, 1992. A New Sorted Logic. In H. J. Ohlbach, editor, *Proc. 16th German Workshop on Artificial Intelligence (GWAI-92)*, Lecture Notes in Artificial Intelligence, vol. 671, pp. 43–54. Springer-Verlag.

Abstract

We present a sound and complete calculus for an expressive sorted first-order logic. Sorts are extended to the semantic and pragmatic use of unary predicates. A sort may denote an empty set and the sort structure can be created by making use of the full first-order language. Technically spoken, we allow sort declarations to be used in the same way than ordinary atoms. Therefore we can compile every first-order logic formula into our logic.

The extended expressivity implies an extended sorted inference machine. We present a new unification algorithm and show that the declarations the unification algorithm is built on have to be changed dynamically during the deduction process. Deductions in the resulting resolution calculus are very efficient compared to deductions in the unsorted resolution calculus. The approach is a conservative extension of the known sorted approaches, as it simplifies to the known sorted calculi if we apply the calculus to the much more restricted input formulas of these calculi.

C. WEIDENBACH, 1993. Extending the Resolution Method with Sorts. In R. Bajcsy, editor, *Proc. 13th Intern. Joint Conference on Artificial Intelligence (IJCAI '93)*, volume 1, pp. 60–65. Morgan Kaufmann.

We essentially strengthen the considered elimination technique. Thus, as a side-effect of this paper we get a stronger elimination based method for proving a subset of second-order logic.

A. SZALAS, 1992. On Natural Deduction in Fixpoint Logics. Technical Report MPI-I-92-203, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

In the current paper we present a powerful technique of obtaining natural deduction (or, in other words, Gentzen-like) proof systems for first-order fixpoint logics. The term “fixpoint logics” refers collectively to a class of logics consisting of modal logics with modalities definable at meta-level by fixpoint equations on formulas. The class was found very interesting as it contains most logics of programs with e.g. dynamic logic, temporal logic and, of course, μ calculus among them.

Fixpoint logics were intensively studied during the last decade. In this paper we are going to present some results concerning deductive systems for first-order fixpoint logics. In particular we shall present some powerful and general technique for obtaining natural deduction (Gentzen-like) systems for fixpoint logics. As those logics are usually totally undecidable, we show how to obtain complete (but infinitary) proof systems as well as relatively complete (finitistic) ones. More precisely, given fixpoint equations on formulas defining nonclassical connectives of a logic, we automatically derive Gentzen-like proof systems for the logic. The discussion of implementation problems is also provided.

U. WALDMANN, 1992. Semantics of Order-Sorted Specifications. *Theoretical Computer Science*, Vol. 94, No. 1, pp. 1-35.

Abstract

Order-sorted specifications (i.e., many-sorted specifications with subsort relations) have been proved to be a useful tool for the description of partially defined functions and error handling in abstract data types.

Several definitions for order-sorted algebras have been proposed. In some papers an operator symbol, which may be multiply declared, is interpreted by a family of functions (“overloaded” algebras), in other papers it is always interpreted by a single function (“non-overloaded” algebras). On the one hand, we try to demonstrate the differences between these two approaches with respect to equality, rewriting, and completion; on the other hand, we prove that in fact both theories can be studied parallelly, provided that certain notions are suitably defined.

Abstract

In this paper I extend the standard first-order resolution method with special reasoning mechanisms for sorts. Sorts are one place predicates. Literals built from one place predicates are called sort literals. Negative sort literals can be compiled into restrictions of the relevant variables to sorts or can be deleted if they fulfill special conditions. Positive sort literals define the sort structure for sorted unification. Sorted unification exploits the sort restrictions of variables. As the occurrence of sort literals is not restricted, it might be necessary to add additional literals to resolvents and factors and to dynamically change the set of positive sort literals used by sorted unification during the deduction process. The calculus I propose thus extends the standard resolution method by sorted unification, residue literals and a dynamic processing of the sort information. I show that this calculus generalizes and improves existing approaches to sorted reasoning. Finally I give some applications to automated theorem proving and abduction.

C. WEIDENBACH, 1993. Unification in Sort Theories and its Applications. MPI-Report MPI-I-93-211, Max-Planck-Institut für Informatik, Saarbrücken.

cf. section II.2.5, page 91

A. WERNER, A. BOCKMAYR AND S. KRISCHER, 1993. A Concept for the Implementation of LSE Narrowing. In *9. Workshop Logische Programmierung*. FU Hagen, Informatik Bericht 146 – 10/1993.

A. WERNER, A. BOCKMAYR AND S. KRISCHER, 1993. How to Realize LSE Narrowing. In *Proc. 2nd Intern. Workshop on Functional/Logic Programming*. LMU München, Techn. Rep. 9311.

U. WERTZ, 1992. First-Order Theorem Proving Modulo Equations. Technical Report MPI-I-92-216, Max-Planck-Institut für Informatik, Saarbrücken.

Abstract

We present refutationally complete calculi for first-order clauses with equality. General paramodulation calculi cannot efficiently deal with equations such as associativity and commutativity axioms. Therefore we will separate a set of equations (called *E*-equations) from a specification and give them a special treatment, avoiding paramodulations with *E*-equations but using *E*-unification for the calculi. Techniques for handling such *E*-equations known in the context of purely equational specifications (e.g. computing critical pairs

with E -equations or introducing extended rules) can be adopted for specifications with full first-order clauses. Methods for proving completeness results are based on the construction of equality Herbrand interpretations for consistent sets of clauses. These interpretations are presented as a set of ground rewrite rules and a set of ground instances of E -equations forming a Church-Rosser system. The construction of such Church-Rosser systems differs from constructions without considering E -equations in a non-trivial way. E -equations influence the ordering involved. Methods for defining E -compatible orderings are discussed. All these aspects are considered especially for the case that E is a set of associativity and commutativity axioms for some operator symbols (then called AC -operators). Some techniques and notions specific to specifications with AC -operators are included.

E. WEYDERT, 1993. About Plausible Reasoning in First-Order Contexts. In *Proc. 1st Dutch/German Workshop on Nonmonotonic Reasoning Techniques and their Applications*. To appear.

cf. section II.2.8, page 96

E. WEYDERT, 1993. Plausible Inference of Default Conditionals. In M. Clarke, R. Kruse, S. Moral, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty: Proc. of Europ. Conference ECSQARU '93*, Lecture Notes in Computer Science, vol. 747, pp. 356–363. Springer-Verlag. Also in *Proc. IJCAI-Workshop: Conditionals in Knowledge Representation*, 1993.

cf. section II.2.8, page 97