Integrated Quantifier Elimination

Andreas Dolzmann

February 3, 2006

Abstract

In this survey paper on our work in the area of quantifier elimination we present integrated quantifier elimination. We discuss three new methods introduced by us for making the three most important real quantifier elimination procedures applicable for the practical usage. We emphasize the necessity and advantage of their combination. Additionally we discuss the integration of quantifier elimination into computer algebra systems and the integration of algorithm of the computer algebra into quantifier elimination. We conclude our work with three applications from different areas.

1 Introduction

Consider the real function

$$f(u, v) = (3u + 3uv^2 - u^3, 3v + 3u^2v - v^3, 3u^2 - 3v^2).$$

The image of the real plane under f is geometrically speaking a 2-dimensional surface in 3-dimensional space. Up to a scaling of a factor of 3 it is the Enneper surface, first introduced by Alfred Enneper around 1863. Due to the two parameters u and v of the function f the above representation of the Enneper surface is called a 2-dimensional parametric representation.

By running through a subset of the parameter space one can generate a part of the plot of the surface. However, given a point in real 3-space we cannot easily decide whether the point belongs to the surface or not. This task is necessary for plotting a part of the surface in a given subset of the real 3-space. Let us, for a moment, change the domain and the range of the function of f to the complex numbers. Then we can by using Gröbner bases techniques quite easily compute the implicitization of the surface, i.e. the minimal variety containing the surface. For the Enneper surface this variety is given by the equation $p = 0$ where

$$\begin{aligned} p \;=\; & 19683x^6 - 59049x^4y^2 + 10935x^4z^3 + 118098x^4z^2 - 59049x^4z + \\ & 59049x^2y^4 + 56862x^2y^2z^3 + 118098x^2y^2z + 1296x^2z^6 + \\ & 34992x^2z^5 + 174960x^2z^4 - 314928x^2z^3 - 19683y^6 + 10935y^4z^3 - \\ & 118098y^4z^2 - 59049y^4z - 1296y^2z^6 + 34992y^2z^5 - 174960y^2z^4 - \\ & 314928y^2z^3 - 64z^9 + 10368z^7 - 419904z^5. \end{aligned}$$

For the complex counterpart of the Enneper surface it can be proved that the variety of p and the image of f are identical. Of course, this result cannot be easily transferred to the reals. As a counterexample consider

$$g(x) = x^2 \quad \text{and} \quad 0 = 0.$$

For the complex numbers, both, the image of g and the variety given by the equation $0 = 0$ are the whole complex plane. For the reals the image of g is only the set of all non-negative numbers whereas the real variety is the entire real line.

Let us switch back to the reals, to the Enneper surface and its known complex implicitization. We want to either prove or disprove that for all real values x, y, and z there are real values u and v with $f(u, v) = (x, y, z)$ if and only if $p = 0$.[1] Using the well-known symbols we can express this by

$$\forall x \forall y \forall z \Big(\exists u \exists v \big(f(u, v) = (x, y, z) \big) \longleftrightarrow p = 0 \Big).$$

In the sense of logic this is a real decision problem and can be theoretically solved by a decision method or a quantifier elimination procedure.

For defining quantifier elimination we need first-order formulas. A first-order formula is constructed from atomic formulas by combining them with the Boolean operators "$\neg$" (not), "$\wedge$" (and), "$\vee$" (or), "$\longrightarrow$" (implication) and "$\longleftrightarrow$" (equivalence). Additionally there are the two operators "$\forall x$" (for all) and "$\exists x$" (there is) ranging over all domain elements.

The construction of the atomic formulas depends on the chosen language which fixes admissible function symbols and relation symbols. The semantics of the symbols are determined by the underlying domain. For the reals we use the language of ordered rings, i.e., addition, additive inverse, and multiplication as functions, 0 and 1 as constants, equality, and the natural order as relations. By using integers as an abbreviation for sums of 1, and allowing the commonly used ordering relation symbols atomic formulas have the form

$$p \, \varrho \, q, \quad \text{where} \quad p, q \in \mathbb{Z}[X_1, \ldots, X_n], \quad \text{and} \quad \varrho \in \{<, \leqslant, =, \neq, \geqslant, >\}.$$

As an example, consider the formula $\exists x(mx + b = 0)$ expressing that a parametrically given line has a real zero.

A quantifier elimination procedure computes for a given first-order formula an equivalent one which contains no quantifiers. In our example such a quantifier-free equivalent is

$$m \neq 0 \vee b = 0.$$

The question whether a given domain for a given language provides quantifier elimination is an important topic in model theory. The reals together with language of ordered rings above allows quantifier elimination. The same language for the domain of integers does not. In one of our applications we will use discretely valued fields with an

[1] This problem was given to us by Eberhard Becker (Dortmund) in 1994 as an open problem.

appropriate language. For all other parts of this paper we use the reals together with the language of ordered rings.

The first mathematical results for real quantifier elimination were by Alfred Tarski around 1935. He published the first real quantifier elimination procedure in [Tar51]. The very fact that there is a quantifier elimination procedure is a very important result for model theory and logic. In 1975 Collins published a new elimination algorithm using his cylindrical algebraic decomposition (CAD) [Col75]. Arnon, a student of Collins, had implemented the quantifier elimination by cylindrical algebraic decomposition [ACM84a, ACM84b] around 1981. Since then, there are continuing developments of algorithms, implementations, and applications of CAD [CH91, Bro98, Bro01, SS03, Sei04]. Starting with its first implementation quantifier elimination have become a practical product ant not only a theoretical result.

Today we can, in practice, solve the questions arising from the study of the real Enneper surface above by using quantifier elimination [H3, H5]. However, not with Tarskis method and not with only one quantifier elimination algorithm. We have computed the solution that the implicitization and the Enneper surface are identical by combining three completely different real quantifier elimination algorithms and other tools of computer algebra and computer logic. We turn back to this solution in Section 7.

Our claim is that we need more than one quantifier elimination algorithm implemented for solving problems. This requires in turn one *integrated* system. Such a system allows on the one hand a manual combination as it was successful for solving the Enneper implicitization problem. On the other hand it allows the implementation of automatic strategies for combining the methods as our fall-back quantifier elimination as introduced in Section 5. One system providing more than one quantifier elimination procedure is our first aspect of integrated quantifier elimination.

All our quantifier elimination procedures rely heavily of the algebraic algorithms of the underlying computer algebra system. For example the Hermitian quantifier elimination relies on efficient Gröbner system computations as well as fast arithmetic in residue class algebras. This *integration* of algebraical into logical algorithms is our second aspect of integrated quantifier elimination.

Finally, our third aspect of integrated quantifier elimination is the *integration* of quantifier elimination into algebraic algorithms provided by computer algebra systems. In this way we have used logical simplifications and complex quantifier elimination for an efficient and fast computation of Gröbner systems.

All these three aspects of integration demonstrate that we have to consider not longer computer algebra systems and implemented quantifier elimination procedures separately but that we have to develop integrated systems providing, both, algebraic and logical computation capabilities.

The plan of this paper is as follows: In the next three sections we discuss our new aspects for the three quantifier elimination procedures considered by us: In Section 2 we introduce our greedy projection [H6] which computes a suitable projection order for CAD based quantifier elimination. In Section 3 we discuss our parallelization [H2] of the virtual substitution methods for three different parallel environments. In Section 4

we discuss our generic Hermitian quantifier elimination [H7] together with some general optimizations. Our integrated system REDLOG is presented in Section 5. A detailed presentation of REDLOG is published by the author together with Seidl in [H5]. In the last four sections 6–9 we present applications of quantifier elimination: We introduce our concept of guarded expressions as an advanced user interface for computer algebra systems as presented in [H1]. We give some remarks on the solution of the real implicitization of the Enneper surface firstly published by the author in [H3]. As a practical application we summarize our application of quantifier elimination to the motion planning of several robots [H8]. As a final application we discuss our parametric solution of systems of linear integer congruences [H4]. This example demonstrates that our concept of integration, introduced for the real quantifier elimination, is also applicable for other domains. Additionally this approach uses quantifier elimination for discretely valued fields demonstrating that even an integration of different domains leads, in some cases, to the solution. Finally, we conclude our results in Section 10.

2 Quantifier Elimination by Cylindrical Algebraic Decomposition

Quantifier elimination by cylindrical algebraic decomposition (CAD) is clearly one of the most important ones. The first version was published by Collins 1975 [Col75]. With his method Collins could dramatically lower the worst-case complexity of a real quantifier elimination method. While Tarskis method is non-elementary recursive, Collins' method is doubly exponential in the number of all variables occurring in the input formula. During the past 30 years there have been considerable research and publications on optimizing CAD. For the application to real quantifier elimination, the introduction of *partial* CAD (PCAD) [CH91] has been one major progress, which affects the extension phase. The latest version QEPCAD B including his remarkable results on computing minimal decompositions [Bro98] are maintained by Brown [Bro03].

All these improvements address the practical applicability in contrast to the theoretical worst-case complexity. As a rule, CAD, provides short formulas containing simple polynomials. The main drawback is its sensitivity to the total number of variables in the input.

The vast majority of improvements are extremely focused on improving the projection phase [McC84, McC88, Hon90, Laz94, Bro01, SS03]. Most surprisingly, all these contributions concentrating on improved projection operators have never examined the relevance of the *order* in which the variables are projected. If one is only interested in pure CAD, then this order can be chosen completely arbitrarily. In the case of quantifier elimination we have to obey some restrictions: We have to project unquantified variables at last and we are not allowed to interchange $\exists$ with $\forall$. There is, however, still a considerable degree of freedom.

The projection phase starts out with the original set $A_r = A$ of polynomials in r variables. In $r-1$ projection steps there are $r-1$ projection sets $A_{r-1}, \ldots, A_1$ of polynomials with one fewer variable in each step generated.

Our results presented here were published in [P6, H6]. The main result is the introduction of the greedy projection, an algorithm for computing a sufficiently good projection order and the corresponding projection set. The quality of a projection order can be only measured in the number of cells of the resulting CAD or the time for constructing the CAD or the time for quantifier elimination. Even so we can perform our complete computation in the first phase, i.e. the projection phase, without actually constructing any CAD.

For achieving this result we discovered in a first step heuristically a measure for a projection set to be good: Consider the sum of total degrees of *all* monomials of *all* polynomials in *all* the projection sets $A_r, \ldots, A_1$ computed in the projection phase wrt. to some order X:

$$\mathrm{sotd}(A, X) = \sum_{i=1}^{r} \sum_{f \in A_i} \sigma(f),$$

where, using the convention $\mathbf{e} = (e_1, \ldots, e_r)$,

$$\sigma\Big(\sum_{\mathbf{e} \in E} a_{\mathbf{e}} x_1^{e_1} \cdots x_r^{e_r}\Big) = \sum_{\mathbf{e} \in E} \sum_{i=1}^{r} e_i.$$

Our tests have shown that for a fixed input a projection set with a small sotd leads generally to smaller CADs and faster computations. Finding a projection order with a minimal sotd requires, in general, the computation of all projection sets for all admissible projection orders. This is practically impossible. Thus we have developed the greedy projection to heuristically find a suitable projection order with a low sotd. In the i-th step of the projection phase we compute from the projection set A_{r-i+1} the sets A_{r-i}^x for each remaining variable x. Then we greedily take one of the A_{r-i+1}^x with a minimal sotd as A_{r-i}.

Though, both, the choice of sotd and the greedy algorithm are only heuristics, our results are very impressive. We have tested our algorithm on almost 50 examples, mostly taken from the literature. The most important results were the following:

- The computation overhead introduced by using the greedy projection is completely negligible. There was not a single example where we had a loss of efficiency in our improved algorithm.
- For all computations we had a time limit of 1 hour. For some examples there are several variable orders which does not permit to finish the CAD computation in our time limit, while others do. In these cases our greedy projection has found always a good variable order such that we do not have exceed out time limit. For some examples we had only a probability of 1/4 for finding such a good variable order.
- In all our examples the computed projection order is not significantly worse than the best possible choice.

It should be noticed here that cylindrical algebraic decomposition is also an important and powerful tool in real algebraic geometry and not only for real quantifier elimination; one prominent other application are adjacency algorithms [ACM84b, SS83]. Our results are, in fact, not restricted to the application to quantifier elimination but can be used in general.

3 Quantifier Elimination by Virtual Substitution

Weispfenning published in 1988 an article about the complexity of linear problems in fields [Wei88]. He proved that in the worst case quantifier elimination is doubly exponential only in the number of quantifier blocks, i.e. the number of sequences of like quantifiers. Moreover, has he shown that quantifier elimination is, in the worst case, singly exponential in the number of quantifiers and polynomial in all other parameters. In his paper he gave the first version of quantifier elimination by virtual substitution for linear formulas. This method matches exactly the complexity bounds proved in his paper. The refined method [LW93] was implemented and improved by the author together with Sturm. Meanwhile, the method has been extended to almost linear formulas [Wei94, Wei97a], i.e., in some circumstances we can eliminate some quantifiers binding variables occurring with degrees greater than one. As a rule, quantifier elimination by virtual substitution can handle formulas with many variables but produces output formulas with a large amount of simple atomic formulas.

Since the algorithm was initially implemented in 1992, there were many new ideas and optimizations added. In [Dol00] the author presented a large collection of new ideas leading to quantifier elimination by repeated condensing.

In the following subsections we present a completely different approach to make this method applicable for more input problems. We study the parallelization of this method using different underlying hardware and software concepts as presented in [P2, H2].

Parallelization is at present again an important topic. In the previous years, there was a dramatic drop in price for multi-processor boards such that they can be used in premium PCs. Meanwhile the most important chip manufacturer presents multi-core processors, including parallel computation capabilities on one chip. This increases the interest on threaded versions of virtual substitution as presented in Section 3.3. Besides multi-core and multi-processor machines there are often large networks of PCs available. Such networks can also be used for performing resource intensive computations. Parallelization of quantifier elimination using a networks of workstations are addressed in Section 3.2. Besides such comparable cheap solutions there are, of course, still several computing centers using parallel and massively parallel computers like the successors of the Cray T3D for which we have also studied the parallelization of the virtual substitution as presented in Section 3.1.

Parallelization of quantifier elimination was already considered for the CAD method. Saunders et al. have parallelized in 1989 [SLA89] the CAD method on a Sequent Symmetry shared-memory machine with 8 processors. In 1993 Hong has parallelized his SACLIB based QEPCAD on a workstation network based on an apparently unpublished communication protocol [Hon93]. In 1997 the author took part on a NSF workshop under the leadership of Hong and Steinberg. The participants worked on a parallelization of the CAD method on the IBM RS6000/SP machine of the Maui High Performance Computing Center. Some results of this workshop are published in [HLRS00].

Consider a formula $\varphi = \exists x_n \cdots \exists x_1(\psi)$, where ψ is a quantifier-free first-order formula. Eliminating all existential quantifiers occurring in this input formula corresponds algorithmically to the construction of a tree. This tree is called *elimination tree*. The

elimination result is then basically constructed by combining the leaves. Each inner node of the tree corresponds to eliminating one existential quantifier from a formula, whereas the combination of its children is the result of this elimination. In this sense each node is a subproblem of our input problem. The elimination tree can be constructed in a depth-first manner, in a breadth-first manner, or in any combination of these approaches. Some optimizations of the substitution methods are based on combining nodes of a particular level and are thus only applicable for parts of the tree that is constructed in a breadth-first manner.

Parallelizing the virtual substitution method makes mainly advantage of work parallelism ($\forall$-Parallelism), i.e. constructing the entire tree in parallel. For special types of input problems, e.g. for pure decision problems, we also can make use of search parallelism ($\exists$-Parallelism). In this case, we search the existence of one special node. After finding that node we have computed the result and we can hence stop our computations.

Implementing a parallel variant of the virtual substitution method, we firstly have to decide how to construct the tree. Arbitrary versions between breadth-first manner and depth-first manner can be implemented. Secondly, we have to schedule the computations of the nodes to the threads and/or processors available.

3.1 Parallelization on a parallel multi-processor computer

First, we have studied the parallelization on a traditional massively parallel computer. The implementation and all test computations were done on the T3D of the ZIB in Berlin. Our software solution was based on Remote REDUCE RR [MN95], a parallel REDUCE version, which uses PVM [GBD+94] as the underlying parallelization mechanism. The parallel quantifier elimination is build in REDLOG and still available in the newest version.

After test computations we made the experience that the communication between the processors are both time consuming and sometimes not reliable. We have thus decided to implement an master slave-approach minimizing the communication costs by efficiently using the computation resources. In a first phase we compute on the master processor in a breadth-first-search manner the elimination tree. We stop the first phase if we have computed three times more of subproblems as processors available. Then we change to the second phase. The master processor distributes to each processor one subproblem. On each processor we compute the complete subtree of the elimination tree starting at the node representing the communicated subproblem. Then the slave process sends back the result to the master. If the master has still an unsolved subproblem, it will send it immediately to the process that just sent back one result. After having received all results the master computes in the final third phase the result and terminates.

All example computations here were focused on linear optimization problems. For this kind of problems we have the advantage that the intermediate result of a slave-process is comparable short, i.e., we have only minimal costs for communicating them back to the master. For optimization problems we get generally a speed up of about 3 for 8 processors used. These results are similar to the results of Saunders, parallelizing CAD on a shared memory machine[SLA89].

3.2 Parallelization on a workstation cluster

The second parallel environment we have studied consists of a small network of workstations connected by 10-MBit Ethernet. For our implementation we have used a C implementation of our algorithm using PARSAC-2 including S-Threads [Küc91, Küc95] and a PVM based extension of the S-threads called DTS. The distributed thread system DTS [BHKR95]) allows us to transports fork/join calls across the network.

The S-thread system and DTS allow us to fork essentially arbitrary many threads over the net. Therefore, we can apply the concept of divide-and-conquer, where there is not an a priori bound for the number of concurrent tasks. Divide-and-conquer algorithms distribute the problem over the computational resources in the fastest possible manner.

When the elimination is distributed over several networked hosts, however, each fork of a function causes its arguments to be sent over the net. Therefore, we have to take care of the communication costs. This means that, whenever a host processes some forked function, all the data passed to this function should actually be processed there at least partially. The algorithm "first_process" has been carefully designed to meet this requirement. For a list of nodes it computes all children and then distributes one half to another thread while computing the other half itself.

We observe an initial overhead of about 60% caused by the parallelization environment, i.e. PVM and DTS, and the fact that the distributed version on one host does not construct the tree in a breadth-first manner, and hence cannot identify as many identical nodes as the sequential algorithm. On more than one host, we obtain moderate speed-ups; with three hosts even compared to the sequential version.

3.3 Parallelization on a parallel multi-processor SMP workstation

One of the advantages of the software design of the S-Thread system together with PVM is, that we can reuse our implementation for running on a multi-processor machine instead of a workstation network. On such a shared memory machine a fork does not need to communicate the parameter values across the net. We have thus developed our algorithm "first_divide". In this algorithm we fork away one half of the subproblems we got. This process is recursively repeated until there is only one node remaining. Then we compute the children of this particular node, before we finally call "first_divide" recursively on the list of the obtained children. This behavior speeds up the distribution of the subtasks to threads possibly leading to better utilization of the available processors. Using 4 processors we obtain a speed-up between 2.5 and 3 for our test examples.

Finally, we have combined distribution to processors with parallelization on a network by connecting two 4-processor SPARCs. Since the S-thread system and DTS cooperate perfectly, there is no additional code necessary for doing so, i.e., the same executable as in the previous section is running. Both, an increase of the number of hosts and/or an increase of the number of processors results in a gain of efficiency.

4 Hermitian Quantifier Elimination

Using results form Pederson, Roy, Szpirglas [PRS93] and Becker, Wöhrmann [BW94] and himself [Wei92], Weispfenning published in 1993 a new quantifier elimination procedure [Wei93, Wei98]. The worst-case complexity of this method is again, as Tarskis method, not elementary recursive. The Hermitian quantifier elimination was refined and implemented by the author in 1994 [Dol94]. Meanwhile it was carefully analyzed and using several new algorithmic ideas reimplemented. Some results are published in a diploma thesis supervised by the author [Gil03].

As a rule, this method is applicable for one quantifier block in front of a conjunction containing as many equations as quantifiers and only few other atomic formulas. It produces formulas containing some huge atomic formulas.

For the CAD method we have discussed a general algorithmic idea for optimizing the algorithm. For the virtual substitution method we have used special computing facilities for widening the range of applications. For the Hermitian quantifier elimination we present here, in some sense, an application driven approach. We have generalized the Hermitian quantifier elimination to a generic quantifier elimination procedure. Generic quantifier elimination is a very successful approach for automatic theorem proving in geometry [DSW98].

The Hermitian quantifier elimination consists of five main steps. These are called iteratively and recursively in a very sophisticated way:

1. Computation of a special normal form based on a disjunctive normal form.
2. Computation of a Gröbner System [Wei92]. The Gröbner system represents a finite case distinction such that in each case we can use Gröbner bases based techniques. The number of cases in the Gröbner systems computed for Hermitian quantifier elimination is one of the main parameters determining the running time and output size. By using real methods for generating and simplifying the case conditions the author could relevantly reduce the number of cases.
3. Generation of additional equations. Again, by generating a finite case distinction we add to a conjunction of inequalities a non-trivial equation. As in Step 2 the size of the case distinction is relevant for the running time and output size.
4. Counting real zeroes based on computations in a residue class algebra [PRS93, BW94]. For the arithmetic in the residue class algebra the author has introduced and used a generalization of the technique of combined structure constants [BWK93]. For larger examples this leads to a considerable speed-up while for smaller ones the overhead is negligible.
5. Construction of type formulas. In this step we compute formulas characterizing that special univariate polynomials have as many positive as negative real zeroes.

Generic quantifier elimination is a generalization of quantifier elimination. It computes to an input formula φ a quantifier-free formula φ'' and a theory Θ, i.e., a conjunction

of negated equations, such that

$$\Theta \longrightarrow (\varphi \longleftrightarrow \varphi'').$$

The main idea behind this concept is, that we can compute Θ dynamically during the elimination process in such a way that it is faster as and/or the output formula is shorter than in case of the regular quantifier elimination. The concept of generic quantifier was firstly published in [DSW98] for the virtual substitution method.

Although the concept of generic quantifier elimination is known the actual design of a generic quantifier elimination depends heavily on the underlying algorithm. In [P7, H7] we have carefully designed a generic variant of the Hermitian quantifier elimination. In order to decide what to add to Θ we have three restrictions:

1. The size of both Θ and φ'' together should be substantially smaller than the computed quantifier-free equivalent φ' computed by the regular quantifier elimination procedure.
2. The negated equations in Θ should in some sense be meaningful for the application.
3. We should only add a restriction to Θ if it supports the generic quantifier elimination.

All three requirements are perfectly matched by our generic Hermitian quantifier elimination.

We have adapted Step 2 and Step 4 of the five steps mentioned above. Instead of computing a Gröbner system, we compute a generic Gröbner system, which is also introduced in [Wei92]. A generic Gröbner system corresponds to only one case of regular Gröbner system. This leads, generally, both to a dramatic speed-up and much shorter output formulas.

As for the Gröbner system computation, the case distinctions made in step 4 can be reduced for our generic variant. For the regular Hermitian quantifier elimination the number of cases generated here is polynomial in the number of inequalities and in the degrees of the polynomials. For our generic variant it is only polynomial in the number of inequalities. This leads again, generally, to faster running times and shorter output formulas.

The computation of type formulas could, in principle, also be optimized by using generic ideas. However, example computations have shown that this would, in general, increase unacceptably the size of the computed theory Θ.

Generic Hermitian quantifier elimination is well-suited for automatic theorem proving and finding in geometry as shown by example computations. The logical and algebraic structure of the input problems, typically arising in this area, matches perfectly the constraints for a successful application of the generic Hermitian quantifier elimination. In contrast to the generic variant of the virtual substitution method it is not restricted to almost linear formulas. The generic projection operator in conjunction

with the CAD method [SS03] provides another generic quantifier elimination without any degree restriction. The total number of variables occurring in the formulation of geometric theorems inhibits the application of generic CAD to this area in the most cases.

5 REDLOG

The author together with Sturm and Seidl is developing REDLOG [DS97c], a package for the computer algebra system REDUCE. It adds to the algebraic facilities of the computer algebra system REDUCE a logical environment. Since the distribution of REDUCE 3.7 REDLOG is an integral part of REDUCE. An application-oriented view of REDLOG is given in [H5].

All quantifier elimination algorithms including our optimizations are implemented and available in REDLOG. Only the PARSAC based parallel implementation of the virtual substitution method is not available in REDLOG.

The most prominent implementation of the CAD, which is highly tuned and efficiently implemented, is QEPCAD B [Bro03]. It is the current version of Hongs original implementation of his partial cylindrical algebraic decomposition [CH91]. For accessing QEPCAD B from within REDLOG we provide an interface.

The computer algebra system REDUCE together with REDLOG is meanwhile an integrated system as defined in the introduction. It provides the three discussed quantifier elimination algorithms. All quantifier elimination methods strongly rely on the algebraic capabilities and their efficient implementation in REDUCE. But the logical algorithms of REDLOG are used also for the algebraic algorithms of REDUCE. The prominent example for this is the implementation of the comprehensive Gröbner basis and Gröbner systems by the author and Sturm. During the computations we make use of simplification and quantifier elimination for simplifying the comprehensive Gröbner bases and their computations. The algebraic mode extension guarded expressions, presented in Section 6, and the parametric solving of congruences, presented in Section 9, are two further examples. Note that the Gröbner system computation is also used for the Hermitian quantifier elimination. Thus REDUCE makes use of REDLOG and REDLOG make use of REDUCE. Hence we cannot any longer separate both components into layers.

In an integrated system a user can manually combine the quantifier elimination procedures available together with other tools for handling first-order formulas in order to compute his solutions. In contrast to such an manual combination the author has introduced the fall-back quantifier elimination. This combines, in an automatic fashion, the virtual substitution method with the CAD method. Given a prenex first-order formula, we start with eliminating quantifier by quantifier, beginning with the innermost one, using the virtual substitution method. If it happens that one quantifier cannot be eliminated due to the degree restriction of the method, we generate an input for the CAD method, and then we try to eliminate all the remaining quantifiers using REDLOGs CAD. Because the CAD is a complete quantifier elimination procedure, this can

only fail due to memory or time restrictions. For our test example suite this fall-back quantifier elimination is always faster or at least as fast as the pure CAD method. Note, that we do not enter CAD if the virtual substitution method succeeds with eliminating all quantifiers. The gain of efficiency is caused by reducing the numbers of quantifiers in the input of the CAD method by eliminating some quantifiers with the virtual substitution method.

6 Guarded Expressions

We consider guarded expressions [P1, H1] as our first application. It is one example for the advantages of an integrated systems. We use the REDLOG system for extending the algebraic user interface of REDUCE.

It is a well-known fact that evaluations obtained by an interactive use of computer algebra systems (CAS) are not entirely correct in general. Typically, some degenerate cases are dropped. Consider for instance the evaluation

$$\frac{x^2}{x} = x,$$

which is correct only if $x \neq 0$. The problem here is that CAS consider variables to be transcendental elements. The user, in contrast, has in mind variables in the sense of logic. In other words: The user does not think of rational functions but of terms.

Next consider the valid expression

$$\frac{\sqrt{x} + \sqrt{-x}}{x}.$$

It is meaningless over the reals. CAS often offer no choice than to interpret roots over the complex numbers even if they distinguish between a real and a complex mode.

Corless and Jeffrey [CJ92] have examined the behavior of a number of CAS with such input data. They come to the conclusion that simultaneous computation of all cases is exemplary but not feasible due to the combinatorial explosion of cases to be considered. Therefore, they suggest ignoring the degenerate cases but to provide the assumptions to the user on request. We claim, in contrast, that it is in fact feasible to compute all possible cases. In practice, this is achieved by the use of the logical functions provided by REDLOG.

Our setting is as follows: Expressions are evaluated to *guarded expressions* consisting of possibly several conventional expressions guarded by quantifier-free formulas. For the above examples, we would obtain

$$\left[\ x \neq 0 \mid x\ \right], \quad \left[\ \mathbf{false} \mid \frac{\sqrt{x}+\sqrt{-x}}{x}\ \right].$$

As the second example illustrates, we are working in real closed fields. The handling of guarded expressions can, however, be easily transferred to other situations.

Our approach can also deal with redundant guarded expressions, such as

$$\left[\begin{array}{c|c} \textbf{true} & |x| - x \\ x \geqslant 0 & 0 \\ x < 0 & -2x \end{array}\right].$$

We use such redundant guarded expressions for a systematic approach of algebraic simplifications of nested expressions by using logical simplifications on the guards.

We use first-order formulas over the language of ordered rings as guards. This provides powerful tools for heuristically reducing the combinatorial explosion of cases: equivalent, redundant, tautological, and contractive cases can be detected by simplification [DS97d] and quantifier elimination. At the time of the paper [H1] REDLOG provided only a linear quantifier elimination for the reals. The availability of general real quantifier elimination procedures and logical algorithms for other domains today extends the scope of our approach.

Our experiments have shown that guarded expressions can be used to overcome well-known problems with interpreting expressions as terms. They provide also a systematic approach for simplifying nested expressions of piecewise defined functions.

7 The Real Implicitization of the Enneper Surface

In the introduction we have sketched a geometrical problem arising in the investigation of the Enneper surface. A detailed survey of all used algorithm and our successful strategy is given in [P3, H3]. Given the function

$$f(u, v) = (3u + 3uv^2 - u^3, 3v + 3u^2v - v^3, 3u^2 - 3v^2)$$

and the equation $p = 0$ with

$$\begin{aligned} p \;=\; & 19683x^6 - 59049x^4y^2 + 10935x^4z^3 + 118098x^4z^2 - 59049x^4z + \\ & 59049x^2y^4 + 56862x^2y^2z^3 + 118098x^2y^2z + 1296x^2z^6 + \\ & 34992x^2z^5 + 174960x^2z^4 - 314928x^2z^3 - 19683y^6 + 10935y^4z^3 - \\ & 118098y^4z^2 - 59049y^4z - 1296y^2z^6 + 34992y^2z^5 - 174960y^2z^4 - \\ & 314928y^2z^3 - 64z^9 + 10368z^7 - 419904z^5 \end{aligned}$$

we want to solve the following two tasks:

1. Find a representation of the Enneper surface, i.e. the image of $\mathbb{R}^2$ under f, suitable for testing whether a given point belongs to the surface.
2. Prove the equality of the Enneper surface to the real variety defined by $p = 0$.

First, we describe a solution of Task 1. As a quantifier elimination problem, we can solve this task by eliminating the quantifiers occurring in

$$\exists u \exists v \Big(x = 3u + 3uv^2 - u^3 \wedge y = 3v + 3u^2v - v^3 \wedge 3u^2 - 3v^2 \Big).$$

For that elimination we use the Hermitian quantifier elimination of REDLOG. Plugging in a given point we can then evaluate the formula to true or false. Points contained in the Enneper surface yield true, all others yield false. In [H3] we gave for Task 1 a solution using both Hermitian quantifier elimination and the virtual substitution method. Our new strategies, meanwhile implemented in the Hermitian quantifier elimination of REDLOG, allows us to eliminate all quantifier using this implementation. No combination of the virtual substitution method, the CAD of REDLOG, and of QEPCAD are able to eliminate in reasonable time and space the quantifiers from the above given formula. Note that from the view of theoretical complexity theory the Hermitian quantifier elimination is non elementary recursive while both other methods are only doubly exponential in the worst-case. Thus, this example shows that the theoretical worst-case complexity is not suitable for deciding which method to use.

The solution of Task 2 is much more difficult than the solution of Task 1. First, we use the result of Task 1. The main ideas of the approach presented in [H3] are the following:

- We split the elimination problem in several subproblems. For this we use normal form computations together with Gröbner bases techniques as described in [DS97d].
- The computed subproblems are either trivial or could be solved by using QEPCAD.

Finally, we have proved that the real variety of p is identical to the Enneper surface, as defined by the image of f. Apparently, there is still no theoretical proof of this fact.

8 Multi Object Motion Planning

In this section based on [H8] we discuss the motion planning for several independent robots. Motion planning using extended quantifier elimination was already discussed in [Wei01a, Wei01b]. We consider, in contrast, a more general framework allowing simultaneous path computations for several independent robots in an environment that changes with time. For computing the relevant data of all paths we use extended quantifier elimination which computes not only a quantifier-free equivalent but in addition sample points for the first block of existentially quantified variables of the input formula [Wei97b].

We first sketch the general features of the situation considered by us: Both, the objects and the environment are semilinear sets described by linear formulas over the reals. The objects are robots and obstacles possibly moving within the environment. In contrast to other approaches we make no convexity assumptions on objects or the environments. All movements of the objects follow piecewise linear paths or trajectories along a finite choice of directions and speed vectors specified in advance. Additionally, the maximal number of linear pieces of a path or trajectory is specified in advance. Other inputs of the motion planning problem are the initial positions of the robots, their final positions, and a time interval for all movements. These additional data may be left parametric. In

this case the output of our algorithms will provide finitely many solutions depending on certain semi-algebraic conditions on the parameters.

As remarked in [Lat91], motion planning for several controllable objects can follow at least two possible approaches: First, one tries to find separate free spaces for the reference point of each object, where the remaining controllable objects are entered in an unspecified parametric position. Second, one gathers all reference points of controllable objects into a single reference point of a composite object of variable shape in a suitable higher dimensional space. We take the second approach. The first one can be also covered by our method but is in our example computations less efficient.

Even in relatively simple cases the multiple object motion planning problem is known to be PSPACE hard. Thus, it is not surprising that our algorithms are exponential in the number of robots and the bound on the number of linear pieces in the trajectories. They are, however, polynomial in all other input data.

For the practical application of our method we have to use an extended quantifier elimination procedure which is able to handle input formulas with numerous variables. This suggests to use the extended quantifier elimination of REDLOG based on the virtual substitution method. While for the CAD method Seidl has presented an extended quantifier elimination [Sei04], there is no extended variant for the Hermitian quantifier elimination.

Recall, that the virtual substitution method is restricted to almost linear formulas. This imposes some of our restrictions, namely the semilinear shape of all objects, the linear segments of the paths, and the predefinitions of the directions and velocities. All these restrictions could be dropped using a general quantifier elimination method. For the applicability of our method it was, however, a crucial result to find linear formulas describing motion planning. The restrictions imposed by the linearity are not significant in practice.

For our sample system we rely heavily on the rather sophisticated linear real quantifier elimination package in REDLOG [DS97c]. It is used in two ways: In a first preprocessing phase, quantifier elimination is used to compute quantifier-free descriptions of the semilinear free spaces. In a second phase, we use then our extended quantifier elimination: Either it outputs false, asserting that there are no solution, or the description of all paths for all robots.

9 Parametric Linear Congruence Solving

As the last application we consider parametric systems of integer congruences. We test multivariate systems of linear congruences for feasibility. In the positive case we obtain at least one sample solution for the system. Parametric solving of linear congruences is outside the scope of the real methods we have discussed. While the application is to the integers our method is based on linear quantifier elimination for discretely valued fields [Stu00, DS99]. This quantifier elimination is implemented in REDLOG, too. Our approach is an example that even the *integration* of different domains is a successful

strategy: We integrate the domain of discretely valued fields into the domain of integers. A detailed discussion of our method for solving parametric system of congruences can be found in [P4, H4].

Our methods allow to prescribe for each constraint a particular modulus in contrast to having only one modulus for the entire system:

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n \equiv b_1 \quad & \mod \quad \mu_1 \\ & \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n \equiv b_m \quad & \mod \quad \mu_m, \end{aligned}$$

where a_{ij}, $b_k \in \mathbb{Z}$ and $\mu_1, \ldots, \mu_m$ are various powers of a fixed prime, a parametrically given prime p, or arbitrary integers.

For given fixed prime numbers p there are efficient linear algebra approaches for the discussed problem available [How86, BN96, SM98]. Our work, in contrast, focuses on *uniform* solutions for a *parametric* prime p.

Our method will reduce the problem of solving such systems to one or several *extended quantifier elimination* problems over the rational numbers with p-adic valuations. The obtained p-adic integer sample solutions are then lifted to the integers. Our methods generally split into two parts: First, finding solutions in suitable rings of p-adic integers $\mathbb{Z}_p$. Second, lifting these solutions to the integers $\mathbb{Z}$. The first part is computationally hard, while the second one is straightforward and efficient. For the special case, where each modulus is some power of a fixed prime, the computationally hard first part can be performed uniformly for all primes. This is the crucial advantage of our approach in contrast to well-known linear algebra methods for concrete fixed primes. For this uniform case, we have developed two methods making the lifting step also as uniform as theoretically possible. These methods can be finally reused for extending our approach to the general case of arbitrary, i.e. not necessarily prime power, moduli.

10 Conclusions

We have briefly sketched the CAD based quantifier elimination, the virtual substitution method, and the Hermitian quantifier elimination together with our optimizations as the three most prominent examples of real quantifier elimination procedures. All methods are highly tuned and efficiently implemented in our integrated system REDLOG. We have presented both an automatic and a manual combination of these methods.

As applications we have presented

- the application of Guarded expressions to a computer algebra system,
- the first solution of the real implicitization of the Enneper surface,
- and motion planning for several controlled objects.

- Besides these applications of the real methods, we discussed the parametric solving of integer congruences as an application of methods from different domains.

All these applications demonstrate that integrated quantifier elimination as presented in this survey article can be successfully applied to problems arising in various areas.

References (Habilitation)

[H1] Andreas Dolzmann and Thomas Sturm. Guarded expressions in practice. In Wolfgang W. Küchlin, editor, *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (ISSAC 97)*, pages 376–383, Maui, HI, July 1997. ACM, ACM Press, New York, 1997.

[P1] Andreas Dolzmann and Thomas Sturm. Guarded expressions in practice. Technical Report MIP-9702, FMI, Universität Passau, D-94030 Passau, Germany, January 1997.

[H2] Andreas Dolzmann, Oliver Gloor, and Thomas Sturm. Approaches to parallel quantifier elimination. In Oliver Gloor, editor, *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation (ISSAC 98)*, pages 88–95, Rostock, Germany, August 1998. ACM, ACM Press, New York, 1998.

[P2] Andreas Dolzmann, Oliver Gloor, and Thomas Sturm. Approaches to parallel quantifier elimination. Technical Report MIP-9803, FMI, Universität Passau, D-94030 Passau, Germany, February 1998.

[H3] Andreas Dolzmann. Solving geometric problems with real quantifier elimination. In Xiao-Shan Gao, Dongming Wang, and Lu Yang, editors, *Automated Deduction in Geometry*, volume 1669 of *Lecture Notes in Artificial Intelligence (Subseries of LNCS)*, pages 14–29. Springer-Verlag, Berlin Heidelberg, 1999.

[P3] Andreas Dolzmann. Solving geometric problems with real quantifier elimination. Technical Report MIP-9903, FMI, Universität Passau, D-94030 Passau, Germany, January 1999.

[H4] Andreas Dolzmann and Thomas Sturm. Parametric systems of linear congruences. In V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing. Proceedings of the CASC 2001*, pages 149–166. Springer, Berlin, 2001.

[P4] Andreas Dolzmann and Thomas Sturm. Solving systems of linear congruences. Technical Report MIP-0101, FMI, Universität Passau, D-94030 Passau, Germany, February 2001.

[H5] Andreas Dolzmann and Andreas Seidl. REDLOG – first-order logic for the masses. *Journal of Japan Society for Symbolic and Algebraic Computation*, 10(1):23–33, 2003.

[H6] Andreas Dolzmann, Andreas Seidl, and Thomas Sturm. Efficient projection orders for CAD. In Jaime Gutierrez, editor, *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation (ISSAC 2004)*, Santander, Spain, July 2004. ACM.

[P6] Andreas Dolzmann, Andreas Seidl, and Thomas Sturm. Efficient projection orders for CAD. Technical Report MIP-0401, FMI, Universität Passau, D-94030 Passau, Germany, January 2003.

[H7] Andreas Dolzmann and Lorenz A. Gilch. Generic Hermitian quantifier elimination. In John A. Campbell Bruno Buchberger, editor, *Artificial Intelligence and Symbolic Computation: 7th International Conference, AISC 2004, Linz, Austria*, volume 3249 of *Lecture Notes in Computer Science*, pages 80–93. Springer-Verlag, Berlin, Heidelberg, 2004.

[P7] Andreas Dolzmann and Lorenz A. Gilch. Generic Hermitian quantifier elimination. Technical Report MIP-0408, FMI, Universität Passau, D-94030 Passau, Germany, July 2004.

[H8] Andreas Dolzmann and Volker Weispfenning. Multiple object semilinear motion planning. To appear in the Journal of Symbolic Computation.

References (Other)

[ACM84a] Dennis S. Arnon, George E. Collins, and Scott McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal on Computing*, 13(4):865–877, November 1984.

[ACM84b] Dennis S. Arnon, George E. Collins, and Scott McCallum. Cylindrical algebraic decomposition II: An adjacency algorithm for the plane. *SIAM Journal on Computing*, 13(4):878–889, November 1984.

[BHKR95] Tilmann Bubeck, Martin Hiller, Wolfgang Küchlin, and Wolfgang Rosenstiel. Distributed symbolic computation with DTS. In Afonso Ferreira and José Rolim, editors, *Parallel Algorithms for Irregularly Structured Problems, 2nd International Workshop, IRREGULAR'95*, volume 980 of *Lecture Notes in Computer Science*, pages 231–248, Lyon, France, September 1995, 1995. Springer-Verlag, Berlin, Heidelberg, New York.

[BN96] Johannes Buchmann and Stefan Neis. Algorithms for linear algebra problems over principal ideal rings. Techincal Report TI-7/96, Technische Hochschule Darmstadt, Fachbereich Informatik, D-64283 Darmstadt, Germany, November 1996.

[Bro98] Christopher W. Brown. Simplification of truth-invariant cylindrical algebraic decompositions. In Oliver Gloor, editor, *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation (ISSAC 98)*, pages 295–301, Rostock, Germany, Aug 1998. ACM, ACM Press, New York.

[Bro99] Christopher W. Brown. Guaranteed solution formula construction. In Sam Dooley, editor, *Proceedings of the ISSAC 99*, pages 137–144. ACM Press, New York, NY, July 1999.

[Bro01] Christopher W. Brown. Improved projection for cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 32(5):447–465, November 2001.

[Bro03] Christopher W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin*, 37(4):97–108, 2003.

[BW94] Eberhard Becker and Thorsten Wörmann. On the trace formula for quadratic forms. In William B. Jacob, Tsit-Yuen Lam, and Robert O. Robson, editors, *Recent Advances in Real Algebraic Geometry and Quadratic Forms*, volume 155 of *Contemporary Mathematics*, pages 271–291. American Mathematical Society, American Mathematical Society, Providence, Rhode Island, 1994. Proceedings of the RAGSQUAD Year, Berkeley, 1990–1991.

[BWK93] Thomas Becker, Volker Weispfenning, and Heinz Kredel. *Gröbner Bases, a Computational Approach to Commutative Algebra*, volume 141 of *Graduate Texts in Mathematics*. Springer, New York, 1993.

[CH91] George E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, September 1991.

[CJ92] Robert M. Corless and David J. Jeffrey. Well ... it isn't quite that simple. *ACM SIGSAM Bulletin*, 26(3):2–6, August 1992.

[Col75] George E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In H. Brakhage, editor, *Automata Theory and Formal Languages. 2nd GI Conference*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183. Springer-Verlag, Berlin, Heidelberg, New York, 1975.

[Dol94] Andreas Dolzmann. Reelle Quantorenelimination durch parametrisches Zählen von Nullstellen. Diploma thesis, Universität Passau, D-94030 Passau, Germany, November 1994.

[Dol00] Andreas Dolzmann. *Algorithmic Strategies for Applicable Real Quantifier Elimination*. Doctoral dissertation, Department of Mathematics and Computer Science. University of Passau, Germany, D-94030 Passau, Germany, July 2000.

[DS97c] Andreas Dolzmann and Thomas Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.

[DS97d] Andreas Dolzmann and Thomas Sturm. Simplification of quantifier-free formulae over ordered fields. *Journal of Symbolic Computation*, 24(2):209–231, August 1997.

[DS99] Andreas Dolzmann and Thomas Sturm. P-adic constraint solving. In Sam Dooley, editor, *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (ISSAC 99), Vancouver, BC*, pages 151–158. ACM Press, New York, NY, July 1999.

[DSW98] Andreas Dolzmann, Thomas Sturm, and Volker Weispfenning. A new approach for automatic theorem proving in real geometry. *Journal of Automated Reasoning*, 21(3):357–380, 1998.

[GBD+94] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM 3 User's Guide and Reference Manual*. Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, September 1994.

[Gil03] Lorenz A. Gilch. Effiziente Hermitesche Quantorenelimination. Diploma thesis, Universität Passau, D-94030 Passau, Germany, September 2003.

[HLRS00] Hoon Hong, Richard Liska, Nicolas Robidoux, and Stanly Steinberg. Elimination of variables in parallel. *SIAM News*, 33(8), October 2000.

[Hon90] Hoon Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In Shunro Watanabe and Morio Nagata, editors, *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 90)*, pages 261–264. ACM Press, New York, August 1990.

[Hon93] Hoon Hong. Parallelization of quantifier elimination on a workstation network. In Gérard Cohen, Teo Mora, and Oscar Moreno, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. Proceedings of the 10th International Symposium, AAECC-10*, volume 673 of *Lecture Notes in Computer Science*, pages 170–179, San Juan de Puerto Rico, Puerto Rico, May 1993. Springer-Verlag, Berlin, Heidelberg, New York.

[How86] John A. Howell. Spans in the module $(\mathbb{Z}_m)^s$. *Linear and Multilinear Algebra*, 19(1):67–77, 1986.

[Küc91] Wolfgang W. Küchlin. PARSAC-2: A parallel SAC-2 based on threads. In Shojiro Sakata, editor, *Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes: 8th International Conference, AAECC-8*, volume 508 of *Lecture Notes in Computer Science*, pages 341–353, Tokyo, Japan, August 1990, 1991. Springer-Verlag, Berlin, Heidelberg, New York.

[Küc95] Wolfgang W. Küchlin. PARSAC-2: Parallel computer algebra on the desktop. In J. Fleischer, J. Grabmeier, F. Hehl, and W. Küchlin, editors, *Computer Algebra in Science and Engineering*, pages 24–43, Bielefeld, Germany, August 1994, 1995. World Scientific, Singapore.

[Lat91] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[Laz94] Daniel Lazard. An improved projection for cylindrical algebraic decomposition. In C. L. Bajaj, editor, *Algebraic Geometry and its Applications: Collections of Papers from Shreeram S. Abhyankar's 60th Birthday Conference*. Springer, Berlin, 1994.

[LW93] Rüdiger Loos and Volker Weispfenning. Applying linear quantifier elimination. *THE Computer Journal*, 36(5):450–462, 1993. Special issue on computational quantifier elimination.

[McC84] Scott McCallum. *An Improved Projection Operation for Cylindrical Algebraic Decomposition*. Ph.D. dissertation, Computer Sciences Department, University of Wisconsin, Madison, 1984.

[McC88] Scott McCallum. An improved projection operation for cylindrical algebraic decomposition of three-dimensional space. *Journal of Symbolic Computation*, 5(1–2):141–161, February–April 1988.

[MN95] Herbert Melenk and Winfried Neun. RR: Parallel symbolic algorithm support for PSL based REDUCE. Preliminary Draft, 1995.

[PRS93] Paul Pedersen, Marie-Françoise Roy, and Aviva Szpirglas. Counting real zeroes in the multivariate case. In F. Eysette and A. Galigo, editors, *Computational Algebraic Geometry*, volume 109 of *Progress in Mathematics*, pages 203–224. Birkhäuser, Boston, Basel; Berlin, 1993. Proceedings of the MEGA 92.

[Sei04] Andreas Seidl. Extending real quantifier elimination by cylindrical algebraic decomposition to get answers. In V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, pages 423–431, St. Petersburg, Russia, July 2004.

[SLA89] B. David Saunders, Hong R. Lee, and S. Kamal Abdali. A parallel implementation of the cylindrical algebraic decomposition algorithm. In Gaston H. Gonnet, editor, *Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation (ISSAC 89)*, pages 298–307, Portland, Oregon, July 1989. ACM Press, New York.

[SM98] A. Storjohann and T. Mulders. Fast algorithms for linear algebra modulo N^*. In Gianfranco Bilardi, Giuseppe F. Italiano, Andreas Pietracaprina, and Geppino Pucci, editors, *Algorithms — ESA '98*, volume 1461 of *Lecture Notes in Computer Science*, pages 139–150, Berlin, 1998. Springer.

[SS83] J. T. Schwartz and M. Sharir. On the piano movers' problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4:298–351, 1983.

[SS03] Andreas Seidl and Thomas Sturm. A generic projection operator for partial cylindrical algebraic decomposition. In Rafael Sendra, editor, *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation (ISSAC 03), Philadelphia, Pennsylvania*, pages 240–247. ACM Press, New York, NY, 2003.

[Stu00] Thomas Sturm. Linear problems in valued fields. *Journal of Symbolic Computation*, 30(2):207–219, August 2000.

[Tar51] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley and Los Angeles, CA, 2nd revised edition, 1951. Reprinted in B. F. Caviness and J. R. Johnson, eds., Quantifier Elimination and Cylindrical Algebraic Decomposition, Texts and Monographs in Symbolic Computation, pp. 24–84. Springer, 1998.

[Wei88] Volker Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1&2):3–27, February–April 1988.

[Wei92] Volker Weispfenning. Comprehensive Gröbner bases. *Journal of Symbolic Computation*, 14:1–29, July 1992.

[Wei93] Volker Weispfenning. A new approach to quantifier elimination for real algebra. Technical Report MIP-9305, FMI, Universität Passau, D-94030 Passau, Germany, July 1993.

[Wei94] Volker Weispfenning. Quantifier elimination for real algebra—the cubic case. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 94)*, pages 258–263, Oxford, England, July 1994. ACM Press, New York, 1994.

[Wei97a] Volker Weispfenning. Quantifier elimination for real algebra—the quadratic case and beyond. *Applicable Algebra in Engineering Communication and Computing*, 8(2):85–101, February 1997.

[Wei97b] Volker Weispfenning. Simulation and optimization by quantifier elimination. *Journal of Symbolic Computation*, 24(2):189–208, August 1997. Special issue on applications of quantifier elimination.

[Wei98] Volker Weispfenning. A new approach to quantifier elimination for real algebra. In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Symbolic Computation, pages 376–392. Springer, Wien, New York, 1998.

[Wei01a] Volker Weispfenning. Semilinear motion planning among moving objects in REDLOG. In V. G. Ganzha and E. W. Mayr, editors, *Computer Algebra in Scientific Computing. Proceedings of the CASC 2001*, pages 541–553. Springer, Berlin, 2001.

[Wei01b] Volker Weispfenning. Semilinear motion planning in REDLOG. *Applicable Algebra in Engineering, Communication and Computing*, 12:455–475, June 2001.