Semilinear motion planning in REDLOG

Volker Weispfenning
Fakultät für Mathematik und Informatik
Universität Passau
D-94030 Passau, Germany

e-mail: `weispfen@uni-passau.de`

MIP-9906
May 12, 1999

Abstract

We study a new type of motion planning problem in dimension 2 and 3 via linear and quadratic quantifier elimination. The object to be moved and the free space are both semilinear sets with no convexity assumptions. The admissible motions are finite continuous sequences of translations along prescribed directions. When the number of translations is bounded in advance, then the corresponding path finding problem can be modelled and solved as a linear quantifier elimination problem. Moreover the problem to find a shortest or almost shortest admissible path can be modelled as a special quadratic quantifier elimination problem. We give upper complexity bounds on these problems, report experimental results using the elimination facilities of the REDLOG package of REDUCE, and indicate a possible application.

1 Introduction

Motion planning for a rigid object in a free space with obstacles in dimensions two or three is an area of intensive research in computational geometry and computer algebra (compare [Lat91] and the survey article [Sha97]). The general approaches to this problem include cylindrical algebraic cell decomposition with neighborhood information, roadmaps, and retraction-type approaches such as Voronoi diagrams of various kinds. Here we restrict our attention to translational motions of semilinear objects in a semilinear free space (compare [LW80, OY85, Yap87, LS87, AS97]). Semilinear sets in $\mathbb{R}^d$ are sets described by a boolean combination of linear inequalities with rational coefficients. So they can be construed as polyhydral sets (without any assumption on convexity or closedness) in a slightly generalized sense. We refer to the motion planning problem for semilinear objects in a semilinear free space as the semilinear motion planning problem.

The fact that the linear theory of the reals admits quantifier elimination [Wei88, LW93, Dri98] entails that the free placement space of a semilinear object in a semilinear free space is again semilinear. So the semilinear motion planning problem is reduced to the motion of a point in a semilinear free space of the same dimension. An analysis of the cylindrical algebraic decomposition approach [Col75, Col98b, Col98a, McC95] shows moreover that all cells arising will be convex polyhydrons. As a consequence, the semilinear motion planning problem can always be solved by a semilinear path, provided it can be solved by an arbitrary continuous path. Here a semilinear path is a continuous path consisting of finitely many translations along straight lines. Notice, however, that the approach via Voronoi diagrams in dimension two will not yield a semilinear path, since it includes in general parabolic arcs [OY85, Yap87]. This can be remedied in dimension two by using Voronoi diagrams for the L^∞-metric or the L^1-metric (compare [LW80]).

In this note we study a related, but different semilinear motion planning problem that appears to be new. We specify as part of the data finitely many vectors $\underline{v}_1, \ldots, \underline{v}_k$ in the ambient space, and a natural number n. We admit only semilinear paths consisting of n translations along admissible directions, i. e. along straight lines that are parallel to one of the vectors $\underline{v}_i$. We refer to paths of this kind as admissible (relative to the given data). We show that a solution of this admissible motion planning problem can be obtained via extended linear quantifier elimination in the reals (compare [Wei88, LW93, Wei97b]). In addition we study also the problem of finding shortest, or nearly shortest admissible paths. This problem can be modelled as a special type of quadratic real elimination problem considered in [Wei97b, Wei97a]. Both linear and special quadratic real quantifier elimination are implemented in the

REDLOG package of REDUCE, and have been applied successfully to a large variety of application problems [DS96, DS97a, SW98, DSW98a, DSW98b, DS99]. By specializing the complexity bounds for linear and special quadratic real quantifier elimination in[Wei88, Wei97b, Wei97a], we arrive at upper complexity bounds for the admissible motion planning problem. We show in particular that this problem is solvable in polynomial time for fixed dimension d of the ambient space and fixed number n of admissible translations. The same applies to the problem of finding admissible paths of (almost) minimal length.

We conclude with computational examples in REDLOG that show the practical usefulness of our quantifier elimination approach for the admissible motion planning problem for a small number n of edges in admissible paths. For larger n intermediate help positions are required. Possible industrial applications may include the automatic movement of a carrier for material in a factory.

An extension of the approach to a dynamically changing free space appears to be feasible.

2 Extended linear quantifier elimination

Linear real quantifier elimination by virtual substitution of test points dates back to the theoretical paper [Wei88]. During the last five years a lot of theoretical work has been done to improve the method, cf. [LW93, Wei94b, DSW98a, DS97b, Wei97a]. After promising experimental implementations by Burhenne in 1990, cf. [Bur90], and by T. Sturm in 1992, the method was efficiently reimplemented within the REDUCE package REDLOG by A. Dolzmann and T. Sturm. REDLOG is in fact a *computer logic system* providing not only quantifier elimination but a sophisticated working environment for first-order logic over various languages and theories, cf. [DS97a]. There are also interfaces to QEPCAD and QERRC available such that these packages can be called from REDLOG and the results are available to be further processed by REDLOG. The REDLOG source code and documentation are freely available on the WWW.[1]

We consider first-order formulas for the ordered field of real numbers. The applicability of the method is essentially restricted to formulas in which the quantified variables occur only with low degrees. Here we restrict our attention to the elimination of variables that occur linearly with at most one additional quadratic inequality allowed. In this case quantifier elimination by the methods of REDLOG is guaranteed to succeed in principle. Quantifiers

[1] `http://www.fmi.uni-passau.de/~redlog/`

are eliminated one by one. For eliminating the quantifiers from an input formula

$$\varphi(u_1, \dots, u_m) \equiv \mathsf{Q}_1 x_1 \dots \mathsf{Q}_n x_n \psi(u_1, \dots, u_m, x_1, \dots, x_n), \quad \mathsf{Q}_i \in \{\exists, \forall\}$$

the elimination starts with the innermost quantifier regarding the other quantified variables within ψ as extra parameters. Universal quantifiers are handled by means of the equivalence $\forall x \psi \longleftrightarrow \neg\exists x \neg\psi$. We may thus restrict our attention to a formula of the form

$$\varphi^*(u_1, \dots, u_k) \equiv \exists x \psi^*(u_1, \dots, u_k, x),$$

where the u_{m+1}, ..., u_k are actually x_i quantified from further outside.

We fix real values a_1, ..., a_k for the parameters u_1, ..., u_k. Then all polynomials occurring in ψ^* become linear or quadratic univariate polynomials in x with real coefficients. So the set

$$M = \{\, b \in \mathbb{R} \mid \psi^*(a_1, \dots, a_k, b) \,\}$$

of all real values b of x satisfying ψ^* is a finite union of closed, open, and half-open intervals on the real line. The endpoints of these intervals are among $\pm\infty$ together with the rational zeroes of the linear polynomials occurring in ψ^*, if no quadratic inequality is present. In presence of an additional quadratic inequality $t \geq 0$ or $t > 0$, the endpoints may also be among the solutions of the corresponding quadratic equation. Candidate terms α_1, ..., α_r for the zeroes of the linear equations can be computed uniformly as a linear polynomial in u_1, ..., u_k. As additional candidate α_0 for a point in M we compute the zero of the partial derivative $\frac{\partial t}{\partial x}$ with respect to x of the quadratic polynomial t that may be present. Notice that α_0 can also be computed uniformly as a polynomial in u_1, ..., u_k whose degree in u_i is not greater than the corresponding degree of u_i in t. Notice that we do not include solutions of the quadratic equation $t = 0$ among our candidates.

If all inequalities in ψ^* are weak, then all the intervals constituting M will, into each direction, be either unbounded or closed. In the latter case, such an interval will contain its real endpoint. Thus M is non-empty if and only if the substitution of $\pm\infty$ or of one of the candidate solutions α_j for x satisfies ψ^*. The substitution of $\pm\infty$ into a linear equation or inequality is evaluated in the obvious sense. By disjunctively substituting all candidates into ψ^* we obtain a quantifier-free formula equivalent to $\exists x \psi^*$.

If ψ^* contains also strict inequalities, we need to add to those candidates for points in M that arise from strict linear inequalities expressions of the form $\alpha \pm \varepsilon$, where α is candidate solution for some left-hand side polynomial occurring in a strict inequality. The symbol ε stands for a positive infinitesimal number. Again the substitution of these expressions into a linear equation or inequality can be rewritten in such a form that there occur neither denominators involving any of the u_i, nor the symbol ε in the result, cf. [Wei97a]. Again this yields a quantifier-free formula equivalent to $\exists x \psi^*$. For practical applications this method, of course, has to be refined by a careful selection of a smaller number of candidate solutions and by a combination with powerful simplification techniques for quantifier-free formulas, cf. [DS97b] for details.

Suppose we have eliminated an existential quantifier. Then we have in general obtained a disjunction $\psi_1' \vee \cdots \vee \psi_r'$. If the next quantifier to be eliminated is also an existential one, then we make use of the equivalence

$$\exists x_{n-1}(\psi_1' \vee \cdots \vee \psi_r') \longleftrightarrow \exists x_{n-1}(\psi_1') \vee \cdots \vee \exists x_{n-1}(\psi_r')$$

eliminating all $\exists x_{n-1}(\psi_j')$ independently. As a consequence, no candidate solutions obtained from, say, ψ_1' are substituted into the other ψ_j. This decreases the complexity class of our procedure for single quantifier blocks from doubly exponential to singly exponential in the number of quantifiers, cf. [Wei88].

Dramatic improvements of the general procedure sketched up to now can be obtained by reducing the number of test candidates for M depending on the structure of the formula ψ^*, cf. [LW93, Wei97a]. Simple instances for such an improvement are extensions of *Gauss elimination* [DSW98a].

An extended quantifier elimination with answers can be obtained in a straightforward way from this method by not constructing a disjunction at the end. Instead all the quantifier-free substitution results are kept separately together with the candidate terms yielding them.

The notion of *virtual substitution* refers to both adding conditions and resolving non-standard subterms such as infinitesimals. The complexity of this method depends on the number of quantified variables and, even more, on the number of quantifier changes. In theory, parameters play a minor role for the complexity. They turn in fact out to be very cheap in practice, too. For existential (or universal) formulas with at most one quadratic inequality, no multiplicative parameters, m quantifiers and at least 2 atomic formulas the number of atomic formulas $\mathrm{at}(\varphi')$ of the quantifier-free output φ' can be bounded in terms of the number of atomic formulas $\mathrm{at}(\varphi)$ of the input formula φ by:

$$\mathrm{at}(\varphi') \leq \mathrm{at}(\varphi)^{m+1}.$$

If in addition all quantified variables are linear, then we have the better bound

$$\mathrm{at}(\varphi') \leq \lceil 2^{-m}\mathrm{at}(\varphi)^{m+1} \rceil.$$

(compare [Wei94a, Wei97a]).

3 Modeling semilinear motion planning

In this section we show how to model the semilinear motion planning problem addressed in the introduction as an extended linear quantifier elimination problem.

Let $\varphi(\underline{x})$ be a formula that is linear w. r. t. a specified list $\underline{x} = (x_1, \ldots, x_d)$ of free variables. Then we denote by $\varphi^{\mathbb{R}}$ the set defined by φ in $\mathbb{R}^d$, i. e. the set of all $\underline{c} \in \mathbb{R}^d$ such that $\varphi(\underline{c})$ holds in $\mathbb{R}$. We call $\varphi^{\mathbb{R}}$ the set defined by φ. A subset S of $\mathbb{R}^d$ is *semilinear* (compare [Dri98]), if it is defined by a linear formula. As a consequence of linear quantifier elimination, every semilinear set is definable by a quantifier-free linear formula. Every conjunction of atomic linear formulas defines a (not necessarily closed, possibly empty) convex polyhedron. So - via a formation of a disjunctive normal form for quantifier-free linear formulas - every semilinear set is a finite union of (not necessarily closed) convex polyhedrons, i. e. a generalized polyhedron.

In the following we restrict our attention to semilinear sets in $\mathbb{R}^d$ for $d = 2$ or $d = 3$. Underlined letters will denote vectors in R^d. We consider a semilinear object P in $\mathbb{R}^d$ given in standard position by a quantifier-free linear formula $\varphi(\underline{x})$. We attach to P in standard position the origin as reference point. After translation in $\mathbb{R}^d$ by a vector $\underline{y}$, the reference point has moved to $\underline{y}$, and P is described by the quantifier-free linear formula $\varphi(\underline{x} - \underline{y})$. We let $\psi(\underline{x})$ be a quantifier-free formula that describes the free space S in which the object P may move as semilinear subset of R^d. $\underline{v}_1, \ldots, \underline{v}_k$ are vectors in R^d that define the admissible directions of translations. Note that a translation along direction $\underline{v}_i$ is given by an arbitrary scalar multiple of $\underline{v}_i$; so the motion may be of arbitrary length moving forward or backward along $\underline{v}_i$.

We specify in addition an initial position $\underline{y}$ and a final position $\underline{z}$ of the reference point of P and a natural number n. Then our **problem** is as follows:

Find a continuous, piecewise linear path α for the reference point of P, leading from the initial position $\underline{y}$ to the final position $\underline{z}$, such that α consists of at most n successive translations in admissible directions, and such that for every position of the reference point of P on this path the object P is contained in the free space S.

We refer to this problem as the semilinear motion planning problem for the data φ, $\psi, \underline{v}_1, \ldots, \underline{v}_k, \underline{y}, \underline{z}$, and n. A path that meets the specifications will be called an *admissible path* for the given data.

In a first preprocessing step we reduce this problem to the special case where the object P is a single point:

Let $\gamma(\underline{u})$ be the formula

$$\forall \underline{x}(\varphi(\underline{x}-\underline{u}) \implies \psi(\underline{x})).$$

Notice that γ is linear in $\underline{x}$ and $\underline{u}$; hence by linear elimination γ has a quantifier-free equivalent $\gamma'(\underline{u})$ that is linear in $\underline{u}$.

The set in $\gamma'^{\mathbb{R}}$ defined by γ' in $\mathbb{R}^d$ is known in the literature as the *free placement space* (or *configuration space*) FP of the object P with respect to the free space S. FP is the set of all positions $\underline{u}$ of the reference point, for which the object P is contained in the free space S.

After computation of γ' we are thus reduced to the following **specialized problem**:

Find a continuous, piecewise linear path α leading from the initial position $\underline{y}$ to the final position $\underline{z}$, such that α consists of at most n successive translations in admissible directions, and such that every point on this path is contained in the free placement space FP.

Next we describe the possibility of moving from position $\underline{q}$ to position $\underline{w}$ by a translation along a direction specified by the vector $\underline{v}$ within the free placement space FP by the formula:

$$\exists t(\underline{w}=\underline{q}+t\underline{v} \ \wedge \ \forall s((t\le s\le 0 \ \vee \ 0\le s\le t) \Longrightarrow \gamma'(\underline{q}+s\underline{v})))$$

This formula $\nu(\underline{q},\underline{w},\underline{v})$ is linear in $s,t,\underline{q},\underline{w}$; so again by by linear elimination it has a quantifier-free equivalent $\rho(\underline{q},\underline{w},\underline{v})$ that is linear in $\underline{q},\underline{w}$.

Then the possibility of moving from position $\underline{q}$ to position $\underline{w}$ by a translation along an admissible direction within FP is described by the formula

$$\rho_k(\underline{q},\underline{w}) := \bigvee_{j=1}^{k} \rho(\underline{q},\underline{w},\underline{v_j})$$

Again ρ_k is quantifier-free and linear in $\underline{q},\underline{w}$.

Finally for specified positive natural number n the following formula describes the possibility of moving from the initial position $\underline{y}$ to the final position $\underline{z}$ by a continuous chain of n translations along admissible directions within FP.

$$\exists \underline{u}_1 \ldots \underline{u}_{n-1}(\rho_k(\underline{y},\underline{u}_1) \ \wedge \ \bigwedge_{i=1}^{n-2} \rho_k(\underline{u}_i,\underline{u}_{i+1}) \ \wedge \ \rho_k(\underline{u}_{n-1},\underline{z}))$$

By definition this formula is linear in $\underline{y}, \underline{u}_1, \ldots, \underline{u}_{n-1}, \underline{z}$. So again by linear elimination it has a quantifier-free equivalent $\sigma_n(\underline{y}, \underline{z})$ that is linear in $\underline{y}, \underline{z}$. Using extended linear elimination we obtain moreover as answers values for the vectors $\underline{u}_1, \ldots, \underline{u}_{n-1}$ that together with the initial vector $\underline{y}$ and the final vector $\underline{z}$ specify the vertices of a continuous piecewise linear path from $\underline{y}$ to $\underline{z}$ that is totally contained in the free placement space FP.

4 Shortest admissible paths

So far we have considered the problem of finding an admissible path for given data or to determine that no such path exists. In this section we consider the more specialized problem of finding a shortest or almost shortest admissible path for given data or to determine that no admissible path exists.

We define the length $L(\pi)$ of an admissible path π in a slightly unusual way as the sum of the squares of the Euclidean lengths of the translation vectors that constitute the path. We want to minimize $L(\pi)$ within the set of all admissible path π for a given set of data. Notice that an admissible path of minimal length will also be a shortest admissible path in the sense of its Euclidean length.

We are going to show that this problem can be formalized as a existential linear elimination problem with one additional quadratic inequality. So it fits into the elimination framework of the REDLOG-package described in section 2.

Let us review the formalization in the last section. Let μ be a new variable.

We enter into the formula describing the existence of an admissible path from constant initial position $\underline{y}$ to constant final position $\underline{z}$ the additional condition that the length of this path is smaller or equal to μ :

$$\exists \underline{u}_1 \ldots \underline{u}_{n-1} (\rho_k(\underline{y}, \underline{u}_1) \ \wedge \ \bigwedge_{i=1}^{n-2} \rho_k(\underline{u}_i, \underline{u}_{i+1}) \ \wedge \ \rho_k(\underline{u}_{n-1}, \underline{z}) \ \wedge$$

$$||\underline{y} - \underline{u}_1||^2 + \sum_{i=1}^{n-2} ||\underline{u}_i - \underline{u}_{i+1}||^2 + ||\underline{u}_{n-1} - \underline{z}||^2 \leq \mu)$$

By definition this formula is linear in $\underline{u}_1, \ldots, \underline{u}_{n-1}$ and μ, except for a single weak inequality that is quadratic in $\underline{u}_1, \ldots, \underline{u}_{n-1}$, and linear in μ. So by quadratic elimination in REDLOG, it has a quantifier-free equivalent $\tau_n(\mu)$ that is linear in μ. From this formula it is easy to compute the infimum L_0 of all real values of μ that satisfy $\tau_n(\mu)$ in $\mathbb{R}^{\geq 0} \ \cup \{\infty\}$.

Then if $L_0 \in \mathbb{R}$, it is the infimum of all $L(\pi)$ for all admissible path π. For any real $r \geq L_0$ one can now apply extended linear elimination to the formula

$$\exists \underline{u}_1 \ldots \underline{u}_{n-1} (\rho_k(\underline{y}, \underline{u}_1) \ \wedge \ \bigwedge_{i=1}^{n-2} \rho_k(\underline{u}_i, \underline{u}_{i+1}) \ \wedge \ \rho_k(\underline{u}_{n-1}, \underline{z}) \wedge$$

$$\|\underline{y} - \underline{u}_1\|^2 + \sum_{i=1}^{n-2} \|\underline{u}_i - \underline{u}_{i+1}\|^2 + \|\underline{u}_{n-1} - \underline{z}\|^2 \leq r)$$

in order to obtain as answer an admissible path π of length $L(\pi) = r$, or the answer "false" if $r = L_0$, and if there is no shortest admissible path.

In general there may not exist a shortest admissible path with given start and end point, as the following example shows:

Example 4.1 *Let* $d = 2, k = 2, n = 3$, $\underline{y} = (0, -1)$, $\underline{z} = (0, 1)$, $\underline{v}_1 = (0, 1)$, $\underline{v}_2 = (1, 0)$, $\varphi := (x_1 = 0 \ \wedge x_2 = 0)$, $\psi := (x_1 \neq 0 \ \vee \ x_2 < -1 \ \vee \ x_2 > 1)$.

Then the infimum of all length of admissible paths is 4 *and this infimum is not attained.*

For the case of a closed moving object and a closed free space we can, however, assert:

Theorem 4.2 *Suppose the object* P *defined by* φ *and the free space* S *defined by* ψ *are closed subsets of* $\mathbb{R}^d$*. Suppose moreover that for the given data there is an admissible path from the given initial position of* P *to the given final position of* P*. Then there exists such a path of smallest length.*

Sketch of the proof. It is obvious that the positions of P that are not contained in S form an open set; consequently the free placement space FP is a closed subset of $\mathbb{R}^d$. Consider the intermediate vertices $\underline{u}_i$ $1 \leq i \leq n-1$ of admissible paths from the given initial position $\underline{y}$ to the given final position $\underline{z}$. Then it is not difficult to see that the set of these $(n-1)$-tuples $(\underline{u}_1, \ldots, \underline{u}_{n-1})$ is a closed subset of $\mathbb{R}^{d(n-1)}$. Let $L_0 \geq 0$ be the infimum of all lengths of these admissible paths. Let π_i be a sequence of admissible paths, such that $L(\pi)$ converges to L_0. Then we have a corresponding sequence of $(n-1)$-tuples of vertices that is bounded. So by passing to a subsequence we may assume that the corresponding sequence of $(n-1)$-tuples of vertices converges in $\mathbb{R}^{d(n-1)}$. Let π be the path defined by the limit tuple. Then by the remark above π is also admissible. Since the length of an admissible path determined by these vertices is a continuous function of $(\underline{u}_1, \ldots, \underline{u}_{n-1})$, we may conclude that $L(\pi) = L_0$.

5 Complexity

From the general upper complexity bounds for linear elimination and quadratic elimination in [Wei94a, Wei97a] we can deduce upper complexity bounds for semilinear motion planning. Let as before $\mathrm{at}(\varphi)$ denote the number of atomic formulas in the formula φ.

Reviewing the quantifier elimination steps necessary to arrive at a solution of the semilinear motion planning problem, we find the following: The first step is the computation of a quantifier-free formula γ' describing the free placement space from the defining quantifier-free formulas φ and ψ defining the object P and the free space S, respectively. This requires the elimination of a block of d universal quantifiers w. r. t. linear variables in front of a formula with $\mathrm{at}(\varphi)+\mathrm{at}(\psi)$ many atomic formulas. So by the general bound, we get:

$$\mathrm{at}(\gamma') \leq 2^{-d}(\mathrm{at}(\varphi)+\mathrm{at}(\psi)+1)^{d+1}$$

Denote the bound on the right hand side by a. Then the successive elimination of the linear quantifiers $\forall s$ and $\exists t$ in the computation of ρ leads to the bound

$$\mathrm{at}(\rho) \leq (a+6)^4/8.$$

These two elimination steps may be considered as preprocessing steps, since they are independent of the admissible directions, the initial and final position and the number n of translations allowed.

The computation of σ_n and the actual solution path requires the extended elimination of a block of $d(n-1)$ existential linear quantifiers in front of a formula with at most $k(a+6)^4/8$ atomic formulas. This leads to an elimination tree with at most

$$(k(a+6)^4/8+1)^{d(n-1)}/2^{d(n-1)} \leq (k(a+7))^{4d(n-1)}/2^{4d(n-1)}$$

end nodes. By [Wei94a] the bit size of the coefficients grows only polynomially during iterated linear elimination. So for $d \leq 3$ the whole elimination procedure is polynomial in the data $\underline{v}_1, \dots, \underline{v}_k$, φ and ψ, and exponentially in n.

A similar argument shows shows that this conclusion holds as well for the computation of a shortest or almost shortest path.

The same complexity bounds also apply to the case where the final position is left unspecified. The result will then specify all final positions reachable from the given initial position with the given number of translations along admissible directions. Similarly for unspecified initial position.

6 Computational examples in REDLOG

For computational examples in the REDLOG package we first specify templates for the input of motion semilinear planning problems with fixed external parameters d, k, n.

For dimension $d = 2,\ k = 3$, and $n \leq 5$ the *input template* looks as follows:

```
y1:=   ; y2:=    ; z1:=    ; z2:=    ; % Initial and final position
phi:=    ; % Defining formula for the object in standard position
v11:=   ; v12:=   ; v21:=   ; v22:=   ; v31:=   ; v32:=   ;
% admissible directions
psi:=    ; % Defining formula for the free space
phip:= sub({x1=x1-u1,x2=x2-u2},phi);
% Defining formula for the object after translation
gamma := all({x1,x2}, phip impl psi);
% Defining formula for the free placement space
gammap:= rlqe(gamma);
% Quantifier-free defining formula for the free placement space
rlatnum(gammap);
% Check for the number of atomic formulas in gammap
gammap1:= rlsimpl(sub({u1=q1+s*v11,u2=q2+s*v12},gammap))$
% Movement from position q in one step in direction v1
gammap2:= rlsimpl(sub({u1=q1+s*v21,u2=q2+s*v22},gammap))$
% Movement from position q in one step in direction v2
gammap3:= rlsimpl(sub({u1=q1+s*v31,u2=q2+s*v32},gammap))$
% Movement from position q in one step in direction v3
rhop1:= rlqe(ex(t, w1=q1+t*v11 and w2=q2+t*v12 and
      all(s, (t <= s <= 0 or 0 <= s <= t) impl gammap1)))$
% Movement from position q in one translation along direction v1
rhop2:= rlqe(ex(t, w1=q1+t*v21 and w2=q2+t*v22 and
      all(s, (t <= s <= 0 or 0 <= s <= t) impl gammap2)))$
% Movement from position q in one translation along direction v2
rhop3:= rlqe(ex(t, w1=q1+t*v31 and w2=q2+t*v32 and
      all(s, (t <= s <= 0 or 0 <= s <= t) impl gammap3)))$
% Movement from position q in one translation along direction v3
rho:= rhop1 or rhop2 or rho3$
%  Movement from position q in one translation in
%admissible directions v1 or v2 or v3
rlatnum(rho); %Check for the number of atomic formulas in rho
%Movement in n <=5 translations in three admissible
%directions v1, v2, v3
sigma1:= rlsimpl(sub({q1=y1,q2=y2,w1=z1,w2=z2},rho));
```

```
% Movement in 1 translation in an admissible direction
sigma2:= rlqea(ex({u11,u12}, rlsimpl(
         sub({q1=y1,q2=y2,w1=u11,w2=u12},rho) and
         sub({q1=u11,q2=u12,w1=z1,w2=z2},rho))));
% Movement in 2 translations in admissible directions
sigma3:= rlqea(ex({u11,u12,u21,u22}, rlsimpl(
         sub({q1=y1,q2=y2,w1=u11,w2=u12},rho) and
         sub({q1=u11,q2=u12,w1=u21,w2=u22},rho) and
         sub({q1=u21,q2=u22,w1=z1,w2=z2},rho))));
% Movement in 3 translations in admissible directions
sigma4:= rlqea(ex({u11,u12,u21,u22,u31,u32},rlsimpl(
          sub({q1=y1,q2=y2,w1=u11,w2=u12},rho) and
          sub({q1=u11,q2=u12,w1=u21,w2=u22},rho) and
          sub({q1=u21,q2=u22,w1=u31,w2=u32},rho) and
          sub({q1=u31,q2=u32,w1=z1,w2=z2},rho))));
% Movement in 4 translations in admissible directions
sigma5:= rlqea(ex({u11,u12,u21,u22,u31,u32,u41,u42}, rlsimpl(
          sub({q1=y1,q2=y2,w1=u11,w2=u12},rho) and
          sub({q1=u11,q2=u12,w1=u21,w2=u22},rho) and
          sub({q1=u21,q2=u22,w1=u31,w2=u32},rho) and
          sub({q1=u31,q2=u32,w1=u41,w2=u42},rho) and
          sub({q1=u41,q2=u42,w1=z1,w2=z2},rho))));
% Movement in 5 translations in admissible directions
```

Here sigma$n$ has to be selected according to the desired value of n.

For dimension $d = 3,\ k = 3$, and $n \leq 5$ the corresponding *input template* looks as follows:

```
$y1:=  ; y2:=  ;  y3:=  ; z1:=  ; z2:=  ; z3=  ;
% Initial and final position
phi:=  ; % Defining formula for the object in standard position
v11:=   ; v12:=   ;  v13:=   ;
v21:=   ; v22:=   ;  v23:=   ;
v31:=   ; v32:=   ;  v33:=   ;
% Admissible directions
psi:=    ; % Defining formula for the free space
phip:= sub({x1=x1-u1, x2=x2-u2, x3=x3-u3}, phi);
% Defining formula for the object after translation
gamma := all({x1,x2,x3}, phip impl psi);
% Defining formula for the free placement space
gammap:= rlqe(gamma);
% Quantifier-free defining formula for the free placement space
```

```
rlatnum(gammap);
% Check for the number of atomic formulas in gammap
gammap1:= rlsimpl(sub({u1=q1+s*v11,u2=q2+s*v12,u3=q3+s*v13},gammap))$
% Movement from position q in one  one step in direction v1
gammap2:= rlsimpl(sub({u1=q1+s*v21,u2=q2+s*v22,u3=q3+s*v23},gammap))$
% Movement from position q in one  one step in direction v2
gammap3:= rlsimpl(sub({u1=q1+s*v31,u2=q2+s*v32,u3=q3+s*v33},gammap))$
% Movement from position q in one  one step in direction v3
rhop1:= rlqe(ex(t, w1=q1+t*v11 and w2=q2+t*v12  and w3=q3+t*v13 and
      all(s, (t <= s <= 0 or 0 <= s <= t) impl gammap1)))$
% Movement from position q in one translation along direction v1
rhop2:= rlqe(ex(t, w1=q1+t*v21 and w2=q2+t*v22  and w3=q3+t*v23 and
      all(s, (t <= s <= 0 or 0 <= s <= t) impl gammap2)))$
% Movement from position q in one translation along direction v2
rhop3:= rlqe(ex(t, w1=q1+t*v31 and w2=q2+t*v32  and w3=q3+t*v33 and
      all(s, (t <= s <= 0 or 0 <= s <= t) impl gammap3)))$
% Movement from position q in one translation along direction v3
rho:= rhop1 or rhop2 or rho3$
%  Movement from position q in one translation in
%admissible directions v1 or v2 or v3
rlatnum(rho);  %Check for the number of atomic formulas in rho
%Movement in n <=5 translations in three admissible
%directions v1, v2, v3
sigma1:= rlsimpl(sub({q1=y1,q2=y2,q3=y3,w1=z1,w2=z2,w3=z3},rho));
% Movement in 1 translation in an admissible direction
sigma2:= rlqea(ex({u11,u12,u13}, rlsimpl(
         sub({q1=y1,q2=y2,q3=y3,w1=u11,w2=u12,w3=u13},rho) and
         sub({q1=u11,q2=u12,q3=u13,w1=z1,w2=z2,w3=z3},rho))));
% Movement in 2 translations in admissible directions
sigma3:= rlqea(ex({u11,u12,u13,u21,u22,u23}, rlsimpl(
         sub({q1=y1,q2=y2,q3=y3,w1=u11,w2=u12,w3=u13},rho) and
         sub({q1=u11,q2=u12,q3=u13,w1=u21,w2=u22,w3=u23},rho) and
         sub({q1=u21,q2=u22,q3=u23,w1=z1,w2=z2,w3=z3},rho))));
% Movement in 3 translations in admissible directions
sigma4:= rlqea(ex({u11,u12,u13,u21,u22,u23,u31,u32,u33}, rlsimpl(
          sub({q1=y1,q2=y2,q3=y3,w1=u11,w2=u12,w3=u13},rho)    and
          sub({q1=u11,q2=u12,q3=u13,w1=u21,w2=u22,w3=u23},rho) and
          sub({q1=u21,q2=u22,q3=u23,w1=u31,w2=u32,w3=u33},rho) and
          sub({q1=u31,q2=u32,q3=u33,w1=z1,w2=z2,w3=z3},rho)));
% Movement in 4 translations in admissible directions
sigma5:= rlqea(ex({u11,u12,u13,u21,u22,u23,u31,u32,u33,u41,u42,u43},
         rlsimpl(
          sub({q1=y1,q2=y2,q3=y3,w1=u11,w2=u12,w3=u13},rho)    and
```

```
          sub({q1=u11,q2=u12,q3=u13,w1=u21,w2=u22,w3=u23},rho) and
          sub({q1=u21,q2=u22,q3=u23,w1=u31,w2=u32,w3=u33},rho) and
          sub({q1=u31,q2=u32,q3=u33,w1=u41,w2=u42,w3=u43},rho) and
          sub({q1=u41,q2=u42,q3=u43,w1=z1,w2=z2,w3=z3},rho))));
% Movement in 5 translations in admissible directions
```

Here again sigma$n$ has to be selected according to the desired value of n. For other values of k, n these templates are modified in the obvious way.

For computing a shortest path the templates are modified as follows: First the appropriate command

`sigma n := rlqea( expression )`

is replaced by

`tau n := rlqe( expression  and`
$\sum_{i=1}^{n-2} \|(\underline{u}_i - \underline{u}_{i+1}\|^2 + \|(\underline{u}_{n-1} - \underline{z}\|^2 \leq \mu)$.

From the resulting quantifier-free formula zeta$n$ one obtains a minimal real value r of μ satisfying zeta$n$. Finally the call

`rlqea( expression  and`
$\sum_{i=1}^{n-2} \|(\underline{u}_i - \underline{u}_{i+1}\|^2 + \|(\underline{u}_{n-1} - \underline{z}\|^2 \leq r)$

yields the desired shortest admissible path.

The specific examples discussed below use these templates with the specified values of d, k, n. In each case we specify in addition the initial and final position, the defining formula for the moving object, the vectors defining the admissible directions, and the defining formula for the free space. All timings refer to a SUN SPARCstation Ultra I with 140 MHz. In the figures the obstacles are hatched upward and the object in initial and final position is hatched downward. Paths are indicated by dotted lines.

Example 6.1 *In this 2D-example a closed unit square is moved in a free space consisting of a closed square with an interior wall.*

Data: $d = 2, n = 3, k = 2$

```
y1:=0; y2:=0; z1:=5; z2:=0;
phi:= 0<=x1<=1 and 0<=x2<=1;
v11:=1; v12:=0; v21:=0; v22:=1;
psi:= 0<=x1<=10 and 0<=x2<=10 and (x2>=8 or  x1<=3 or x1>=4);
```

Answer:

```
sigma3 := {{true,{u11 = 0,u12 = 9,u21 = 5,u22 = 9}}}
```

Time: 0.4s

In order to compute an admissible path of shortest length, we first compute

```
tau3 := mu - 153 >= 0
```

Time: 7.5s

Next we obtain the shortest admissible path by computing

```
zeta3 := {{true,{u11 = 0,u12 = 8,u21 = 5,u22 = 8}}} }
```

Time: 1.5s

Notice that this shortest path is actually different from the one computed previously.

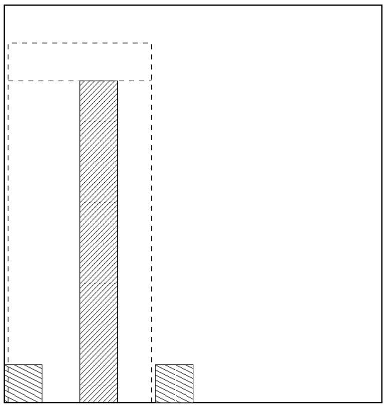

Figure 1

Example 6.2 *In this 2D-example a closed L-shaped (and hence non-convex) object is moved in a free space consisting of a closed rectangle with obstacles.*

Data: $d = 2, n = 3, k = 2$

```
y1:=0; y2:=8; z1:=1; z2:=0;
phi:= (0<=x1<=2 and 0<=x2<=1) or (0<=x1<=1 and 1<=x2<=2);
v11:=1; v12:=0; v21:=0; v22:=1;
psi:= 0<=x1<=4 and 0<=x2<=10 and (x1>=1 or x2>=1) and
      (x1<=2 or x2<=2);
```

Answer:

```
sigma3 := {{true,{u11 = 0,u12 = 1,u21 = 1,u22 = 1}}}
```

Time: 0.3s

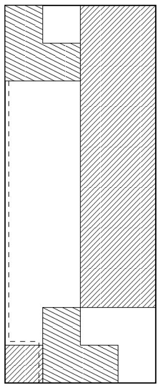

Figure 2

Example 6.3 *In this 2D-example a closed unit square is moved in a free space consisting of a closed square with additional obstacles. The path requires a change of direction at a non-collision point.*

Data: $d = 2, n = 3, k = 2$

```
y1:=0; y2:=0; z1:=7; z2:=7;
phi:= (0<=x1<=1 and 0<=x2<=1);
v11:=1; v12:=0; v21:=1; v22:=1;
psi:= 0<=x1<=10 and 0<=x2<=10 and
      (x2<=3 or x2>=7 or (x2<=x1-1  and  x2>=x1-3));
```

Answer:

```
sigma3 := {{true,{u11 = 2,u12 = 0,u21 = 9,u22 = 7}}}
```

Time: 2s

Variant for computing the shortest path:

```
tau3 := mu - 106 >= 0
```

Time: 3.3s (Minimal value mu = 106)

```
zeta3 := {{true,{u11 = 2,u12 = 0,u21 = 9,u22 = 7}}}\\
```

Time: 0.8s

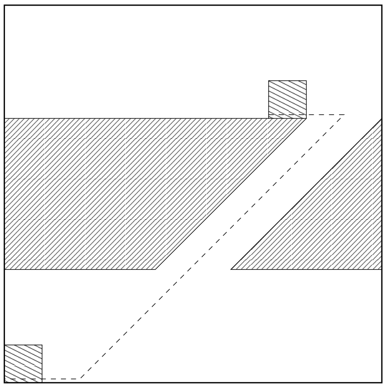

Figure 3

Example 6.4 *In this 2D-example a closed unit square is moved in a free space consisting of a closed square with additional walls.*
Data: $d = 2, n = 4, k = 2$

```
y1:=0; y2:=0; z1:=9; z2:=0;
phi:= 0<=x1<=1 and 0<=x2<=1;
v11:=0; v12:=1; v21:=1; v22:=1;
psi:= 0<=x1<=10 and 0<=x2<=10 and
   (x1=3 impl x2>=2) and (x1=8 impl x2>=1) and (x1=6 impl 2*x2<=5);
```

Answer:

```
sigma4 := {{true,

            {u11 =  - epsilon2 + 5,

             u12 =  - epsilon2 + 5,

             u21 =  - epsilon2 + 5,

                     - 2*epsilon1 + 1
             u22 = -------------------,
                             2
```

```
u31 = 9,

        - 2*epsilon1 + 2*epsilon2 + 9
u32 = --------------------------------}}}
                     2
```

Time: 5s

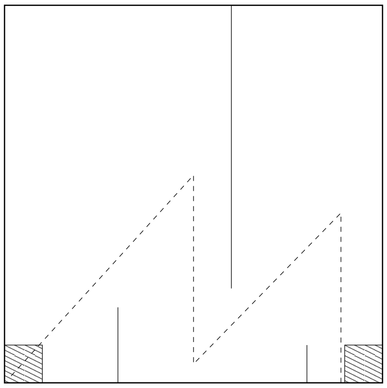

Figure 4

Example 6.5 *In this 3D-example a point is moved in an oblique way over a barrier.*
Data: $d = 3, n = 3, k = 3$

```
y1:=0; y2:=0; y3:=0; z1:=5; z2:=4; z3:=0;
phi:= 0=x1 and 0=x2;
v11:=1; v12:=1; v13:=1; v21:=1; v22:=0; v23:=0;
v31:=1; v32:=1; v33:=-1;
psi:= 0<=x1 and 0<=x2 and 0<=x3 and (x2<=2 or x2>=3 or x3>=1);
```

Answer:

```
sigma3 := {{true,
            {u11 = 2,u12 = 2,u13 = 2,u21 = 3,u22 = 2,u23 = 2}}}
```

Time: 0.5s

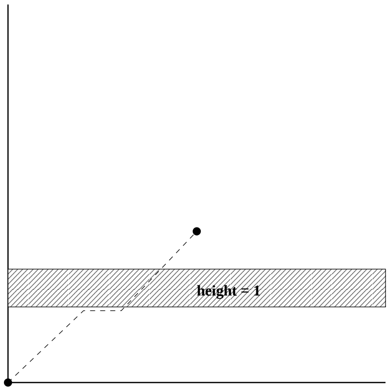

Figure 5

Example 6.6 *In this 3D-example a unit cube is moved inside a cube with two additional walls from one corner to the diagonally opposed corner.*
Data: $d = 3, n = 3, k = 3$:

```
y1:=0; y2:=0; y3:=0; z1:=9; z2:=9; z3:=9;
phi:= 0<=x1<=1 and 0<=x2<=1 and 0<=x3<=1;
v11:=1; v12:=0; v13:=0; v21:=0; v22:=1; v23:=0;
v31:=0; v32:=0; v33:=1;
psi:= 0<=x1<=10 and 0<=x2<=10 and 0<=x3<=10 and
      (x2= 10 - x1 impl x3>=8) and
      ((x1>=3 and x1<=7) impl (x2<=3 or x2>=7));
```

Answer:

```
sigma3 := {{true,
            {u11 = 0,u12 = 0,u13 = 9,u21 = 0,u22 = 9,u23 = 9}}}
```

Time: 5s

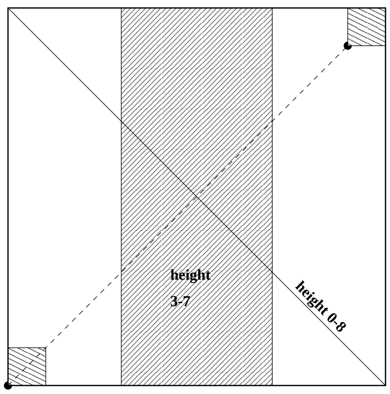

Figure 6

We conclude with two more complicated examples, where $n = 4$ or $n = 5$. They illustrate the rapid growth of the complexity of the algorithm with the number n of edges in the paths.

Example 6.7 *In this 2D-example an L-shaped object is moved in a free space consisting of a closed square with additional walls. Data:* $d = 2, k = 3, n = 4$ *or* $n = 5$

```
y1:=0; y2:=8; z1:=8; z2:=0;
phi:= (0<=x1<=2 and 0<=x2<=1) or (0<=x1<=1 and 1<=x2<=2);
v11:=1; v12:=0; v21:=0; v22:=1; v31:=1; v32:=1;
psi:= 0<=x1<=10 and 0<=x2<=10 and (x1<=2 or x1>=3 or x2>=7) and
      (x1<=4 or x1>=5 or x2<=8 or x2>=9) and
      (x1<=7 or x1>=8 or x2>=2);
```

Answer for $n = 5$*:*

```
sigma5 := {{true,
 {u11 = 0,u12 = 7,u21 = 3,u22 = 7,u31 = 3,u32 = 3,u41 = 8,u42 = 8}}}
```

Time: 975s

Answer for $n = 5$*:*

```
sigma4 := {}
```

Time: 68s

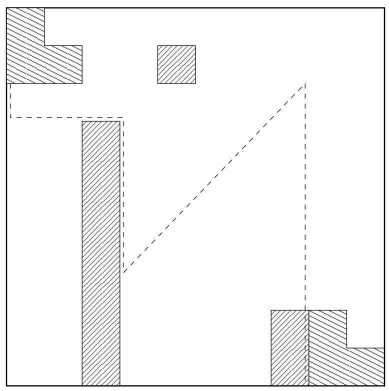

Figure 7

Example 6.8 *In this 2D-example an oblique object is moved in a free space consisting of a closed square with additional walls.*
Data: $d = 2, k = 3, n = 4$

```
y1:=0; y2:=0; z1:=7; z2:=0;
v11:=1; v12:=0; v21:=0; v22:=1; v31:=1; v32:=1;
phi:= 0<=x1<=2 and x1<=x2<=x1+1;
psi:= 0<=x1<=10 and 0<=x2<=10 and (x1>=2 or x2<=8) and
(x1<=9 or x2<=4 or x2>=5) and (x1<=3 or x1>=4 or x2>=6)
and (x1<=3 or x1>06 or x2<=5 or x2>=6);
```

Answer:

```
sigma4 := {{true,
 {u11=0, u12=-epsilon1 + 4, u21 = epsilon1 + 2, u22=6,u31=7,u32=6}}}
```

Time: 1050s

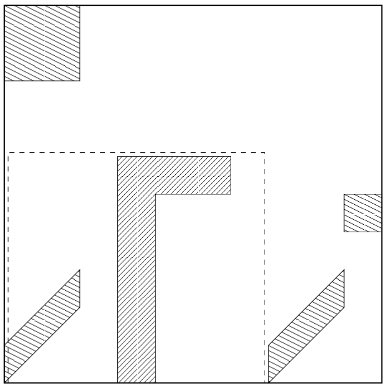

Figure 8

Example 6.9 *Variant of the previous example with different final position that requires a movement with* $n = 5$. *Data:* $d = 2, k = 3, n = 5$

```
y1:=0; y2:=0; z1:=8; z2:=0;
v11:=1; v12:=0; v21:=0; v22:=1; v31:=1; v32:=1;
phi:= 0<=x1<=2 and x1<=x2<=x1+1;
psi:= 0<=x1<=10 and 0<=x2<=10 and (x1>=2 or x2<=8) and
(x1<=9 or x2<=4 or x2>=5) and (x1<=3 or x1>=4 or x2>=6)
and (x1<=3 or x1>06 or x2<=5 or x2>=6);
```

Here no result was obtained within 45 minutes.
If one simplifies the problem by specifying an intermediate "help position" of the object to be reached by a movement with $n = 3$, *followed by a movement with* $n = 2$ *to the final position, then the solutions are obtained rather quickly:*
First partial problem:

```
y1:=0; y2:=0; z1:=7; z2:=6;
```

Second partial problem:

```
y1:=7; y2:=6; z1:=8; z2:=0;
```

Answer for the first part:

```
sigma3 := {{true,
       {u11 = 0,u12 =  - epsilon1 + 4,u21 = epsilon1 + 2,u22 = 6}}}
```

Time: 40s

Answer for the second part:

```
sigma2 := {{true,{u11 = 7,u12 = 0}}}
```

Time: 0.25s

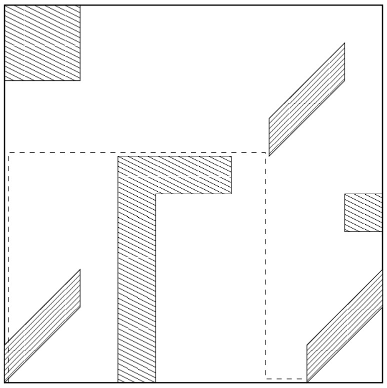

Figure 9

7 Conclusions

We have shown that admissible semilinear motion planning can be performed via linear real quantifier elimination with answers. The extended problem of finding shortest or almost shortest admissible paths can be solved via quadratic real quantifier elimination with answers. The approach is implemented in the REDLOG package of REDUCE. It is of practical use for a small number n of edges in admissible paths. For larger n intermediate help positions are required. Possible industrial applications may include the automatic movement of a carrier for material in a factory.
An extension of the approach to a dynamically changing free space appears to be feasible.

References

[AS97] Boris Aronov and Micha Sharir. On translational motion planning of a convex polyhedron in 3-space. *Siam Journal on Computing*, 26(6):1785–1803, December 1997.

[Bur90] Klaus-Dieter Burhenne. Implementierung eines Algorithmus zur Quantorenelimination für lineare reelle Probleme. Diploma thesis, Universität Passau, D-94030 Passau, Germany, December 1990.

[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. Gesellschaft für Informatik, Springer-Verlag, Berlin, Heidelberg, New York, 1975.

[Col98a] Georg E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Symbolic Computation, pages 85–121. Springer, Wien, New York, 1998.

[Col98b] George E. Collins. Quantifier elimination by cylindrical algebraic decomposition - twenty years of progress. In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Symbolic Computation, pages 8–23. Springer, Wien, New York, 1998.

[Dri98] Lou van den Dries. *Tame Topology and o-minimal structures.* Cambridge University Press, 1998.

[DS96] Andreas Dolzmann and Thomas Sturm. *Redlog User Manual.* FMI, Universität Passau, D-94030 Passau, Germany, October 1996. Edition 1.0 for Version 1.0.

[DS97a] Andreas Dolzmann and Thomas Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.

[DS97b] 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. Technical Report MIP-9901, FMI, Universität Passau, D-94030 Passau, Germany, January 1999. To appear in the proceedings of the ISSAC 99.

[DSW98a] 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.

[DSW98b] Andreas Dolzmann, Thomas Sturm, and Volker Weispfenning. Real quantifier elimination in practice. In B. H. Matzat, G.-M. Greuel, and G. Hiss, editors, *Algorithmic Algebra and Number Theory*, pages 221–247. Springer, Berlin, 1998.

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

[LS87] Daniel Leven and Micha Sharir. Planning a purely translational motion for a convex object in two-dimensional space using generalized voronoi diagrams. *Discrete and Computational Geometry*, 2:9–31, 1987.

[LW80] D.T. Lee and C.K. Wong. Voronoi diagrams in $l_1(l_\infty)$ metrics with 2-dimensional storage applications. *Siam Journal on Computing*, 9(1):200–211, February 1980.

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

[McC95] Scott McCallum. Partial solution to path finding problems using the CAD method. Electronic Proceedings of the IMACS ACA 1995 on `http://math.unm.edu/ACA/1995.html`, 1995.

[OY85] Colm Ó'Dúnlaing and Chee Yap. A "restriction" method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1985.

[Sha97] Micha Sharir. Algorithmic motion planning. In Jacob Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 733–754. CRC Press, 1997.

[SW98] Thomas Sturm and Volker Weispfenning. Computational geometry problems in REDLOG. In Dongming Wang, editor, *Automated Deduction in Geometry*, volume 1360 of *Lecture Notes in Artificial Intelligence (Subseries of LNCS)*, pages 58–86. Springer-Verlag, Berlin Heidelberg, 1998.

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

[Wei94a] Volker Weispfenning. Parametric linear and quadratic optimization by elimination. Technical Report MIP-9404, FMI, Universität Passau, D-94030 Passau, Germany, April 1994.

[Wei94b] 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.

[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.

[Yap87] Chee Yap. An $o(nlogn)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Discrete and Computational Geometry*, 2:365–393, 1987.