



Multiple object semilinear motion planning

Andreas Dolzmann*, Volker Weispfenning

University of Passau, Passau, Germany

Received 5 April 2006; accepted 15 November 2006

Available online 7 January 2007

Abstract

We present a method based on extended linear real quantifier elimination for multiple object semilinear motion planning, i.e. finding collision-free trajectories for several robots in a time dependent environment. For practical applicability the method is limited to polygonal objects and linear trajectories. It can, however, deal with situations involving even non-convex objects.

© 2007 Elsevier Ltd. All rights reserved.

Keywords: Linear real quantifier elimination; Motion planning; Trajectory finding; REDLOG

1. Introduction

This paper is the third in a series of articles (Weispfenning, 2001a,b) dealing with semilinear motion planning via the method of extended real linear quantifier elimination. The first paper studied motion planning for a single object in a static environment, the second motion planning for a single object in an environment that changes with time. Here we are concerned with simultaneous motion planning for several controllable objects in an environment that changes with time. The common features in the three papers are the following: Both objects and environment are semilinear sets described by linear formulas over the reals. 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. Moreover the 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 objects, the final positions of the objects and a time interval for all movements. These additional data may be left parametric; then the output of our algorithms will provide finitely many solutions depending on certain semi-algebraic conditions on the parameters.

* Corresponding address: Universität Passau, Fakultät für Mathematik und Informatik, 94030 Passau, Germany.
E-mail addresses: dolzmann@uni-passau.de (A. Dolzmann), weispfen@uni-passau.de (V. Weispfenning).

As remarked in [Latombe \(1991\)](#), Chapter 8, Section 2, 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 high dimensional space. Here we concentrate on the second approach, and only sketch the first one. For other papers following the same general approach cf. e.g. [Schwartz and Sharir \(1983\)](#), [Sharir and Sifrony \(1988\)](#) and [Barraquand et al. \(1989\)](#).

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

For the practical examples we rely heavily on the rather sophisticated linear real quantifier elimination package in REDLOG ([Dolzmann and Sturm, 1997a](#)). It is used in two ways. First in a preprocessing phase quantifier elimination is used essentially to compute quantifier-free descriptions of semilinear free spaces. The actual solution producing a piecewise linear trajectory (or asserting the impossibility of such a trajectory for the given data) uses then the extended linear quantifier elimination facilities of REDLOG, in order to get sample solutions.

As the passage from single object motion planning to multiple object motion planning is far from routine we describe the actual modeling of the latter problem by linear formulas in considerable detail. We also analyze and compare the two main sources of complexity of the algorithm. First the *combinatorial complexity* that arises from the large selection of combination of speed vectors for the various controllable objects. Second the inherent complexity of our quantifier elimination method. In our examples we observe that the combinatorial complexity dominates by far the complexity of the total algorithm. This motivates a user-guided heuristic selection of sequence of speed vectors as an additional input. Our heuristic examples show that this additional restriction does indeed cause a significant reduction of the total complexity.

The plan of the paper is as follows. In Section 2 we summarize the necessary background from computer logic. In Section 3 we discuss in detail our model of multiple object semilinear motion planning. In Section 4 we present the (extended) linear quantifier elimination algorithm that we refer to. This concludes the more theoretical part of this note. In Section 5 we describe our demonstration system for motion planning and sketch some basics of REDLOG, which provides the software platform for our demonstration system. In Section 6 we present a sample computation. Finally, we conclude with our results in Section 7.

2. Formal background and formalism

2.1. First-order formulas

We consider first-order formulas over the language of ordered rings. Thus, in an intuitive way, atomic formulas are built up by combining polynomials of a certain multivariate polynomial ring with the usual relations $<$, \leq , $=$, \neq , \geq , and $>$. Atomic formulas are combined in quantifier-free formulas with the usual logical operators \neg , \wedge , \vee , and \longrightarrow . First-order formulas are generated from quantifier-free formulas by possibly using additionally the quantifiers $\exists x$ and $\forall x$. In our case, all variables range over the field of real numbers. Variables not occurring in any quantifier $\exists x$ or $\forall x$ are called parameters. For given real values for the parameters each first-order formula has a unique truth value, though it might be difficult to compute it. A quantifier-free and parameter-free formula can, in contrast, easily be evaluated to a truth value.

For a first-order formula φ containing at most the parameters x_1, \dots, x_n , we will write sometimes $\varphi(x_1, \dots, x_n)$ to emphasize the dependence of φ on the parameters. This notation allows us also to use a simplified notation for the substitution. Instead of $\varphi[x_1/t_1, \dots, x_n/t_n]$ we will use $\varphi(t_1, \dots, t_n)$. Note that for real values $\xi = (\xi_1, \dots, \xi_n)$ we denote as $\varphi(\xi) = \varphi(\xi_1, \dots, \xi_n)$, as usual, the evaluation of φ at the point ξ in the field of real numbers.

We extend also the notion of the quantifiers $\forall x$ and $\exists x$ to vectors of variables, namely $\forall(x_1, \dots, x_n)$ and $\exists(x_1, \dots, x_n)$, respectively. We interpret this notion in the natural sense.

In a similar way we extend the notion of substitution: Let $x = (x_1, \dots, x_n)$ be a vector of variables and $t = (t_1, \dots, t_n)$ be a vector of terms; then we denote as $\varphi[x/t]$ the substitution $\varphi[x_1/t_1, \dots, x_n/t_n]$ for a first-order formula φ .

A formula φ is linear in a set V of variables if there is no power of a variable of V contained in φ and there is no product of two variables of V . We call a formula linear if it is linear in each set of variables.

2.2. Quantifier elimination

A quantifier elimination procedure computes for a given first-order formula φ in parameters (u_1, \dots, u_m) a quantifier-free formula $\varphi'(u_1, \dots, u_m)$ which is equivalent. In other words $\varphi(\xi_1, \dots, \xi_m)$ and $\varphi'(\xi_1, \dots, \xi_m)$ have the same truth value for all real values (ξ_1, \dots, ξ_m) for the parameters (u_1, \dots, u_m) .

Let φ be a first-order formula linear in some set V of variables. Then there is an algorithm that computes for φ a quantifier-free equivalent φ' which is in turn linear in V . Such an algorithm is called linear quantifier elimination. In other words, the set of linear formulas is closed under linear quantifier elimination.

2.3. Extended quantifier elimination

Extended quantifier elimination computes for an input formula of the form

$$\varphi \equiv \exists x_1 \cdots \exists x_n (\psi),$$

where $\psi(x_1, \dots, x_n, u_1, \dots, u_m)$ is a quantifier-free formula a schema of the following form:

$$\left[\begin{array}{l|l} \gamma_1 & x_1 = t_{11} \cdots x_n = t_{1n} \\ \vdots & \vdots \\ \gamma_l & x_1 = t_{l1} \cdots x_n = t_{ln} \end{array} \right],$$

such that the following properties hold:

- (i) The t_{ij} are expressions in the parameters (u_1, \dots, u_m) . These expressions may contain fractions.
- (ii) All guards $\gamma_i(u_1, \dots, u_m)$ are quantifier-free formulas.
- (iii) The disjunction $\gamma_1 \vee \cdots \vee \gamma_l$ of all guards is equivalent to φ , i.e. a quantifier-free equivalent to the input formula.
- (iv) If $\gamma_i(\xi_1, \dots, \xi_m)$ holds for some $\xi_1, \dots, \xi_m \in \mathbb{R}$ then

$$\psi(t_{i1}(\xi_1, \dots, \xi_m), \dots, t_{in}(\xi_1, \dots, \xi_m), \xi_1, \dots, \xi_m)$$

holds, too.

In other words, extended quantifier elimination provides for existentially quantified variables a sample point for which the formula holds. This sample point may depend on the chosen values for the parameters.

3. Modeling and algorithms

In this section we discuss in detail how to model a given motion planning problem as a first-order formula and how to solve it by applying extended quantifier elimination. We divide the description into three parts. Firstly, we state the problem formally. We focus on stating the problem in such a way that the first-order formula describing the problem reflects the formal problem specification as far as possible. Secondly, we explain how to construct a first-order formula describing the problem. During this construction we already use linear real quantifier elimination to compute certain subformulas. Finally, we describe how to solve the problem by applying extended linear real quantifier elimination to the previously constructed first-order formula. In a final section we discuss some options and alternatives of the first-order formula construction.

3.1. Problem formulation

We consider a time dependent environment, i.e. the set of all admissible points for the robots. This may typically consist of a static environment together with some obstacles moving in this environment in a known deterministic manner. This environment is a semi-algebraic set described by a formula $\sigma(t, x)$, where $x = (x_1, x_2)$ for two-dimensional motion planning or $x = (x_1, x_2, x_3)$ for three-dimensional motion planning. The variable t ranges over the time. The environment at time $\tau \in \mathbb{R}$ is given as

$$\{x \in \mathbb{R}^s \mid \mathbb{R} \models \sigma(\tau, x)\},$$

where s is either 2 or 3. Our approach does not depend on the dimension s . In the sequel we have always $s = 2$ or $s = 3$ in mind. Higher dimensions $s \in \mathbb{N}$ are, of course, also possible, but have less practical relevance.

All aspects of the environment can change with the time. Consider for example that the environment is simply a rectangular region. Then this region may change its dimension by blowing up. A corresponding first-order formula σ is, e.g.,

$$\sigma(t, x_1, x_2) \equiv x_1 \geq 0 \wedge x_2 \geq 0 \wedge x_1 \leq 10 \cdot (t + 1) \wedge x_2 \leq 5 \cdot (t + 1).$$

This formula describes a rectangular region of size 10×5 at time 0, then one of size 20×10 at time 1, and in general a rectangular region of size $10 \cdot (\tau + 1) \times 5 \cdot (\tau + 1)$ at time τ .

The scope of first order allows us not only to consider smooth changes but even sudden changes. Consider for example the formula

$$\begin{aligned} \sigma(t, x_1, x_2) \equiv \\ t \leq 10 &\longrightarrow (x_1 \geq 0 \wedge x_2 \geq 0 \wedge x_1 \leq 10 \wedge x_2 \leq 10) \\ t > 10 &\longrightarrow (x_1 \geq 0 \wedge x_2 \geq 0 \wedge x_1 \leq 5 \wedge x_2 \leq 5). \end{aligned}$$

First it describes a 10×10 rectangle and then, after $t = 10$, a 5×5 rectangle.

Recall, that typically we use such a time dependent environment to consider besides fixed obstacles additional moving obstacles as indicated above and discussed further in Section 5.2.

Inside the environment there are several robots. These robots are the essential objects in our model. We denote as $R = \{1, \dots, m\}$ the finite set of all robots. Each robot $r \in R$ is just like the environment given as an semi-algebraic set, i.e. there is for each robot $r \in R$ a first-order formula $\varrho_r(t, x)$. In this description we assume that one special point of the robot is located at

the origin of our Cartesian coordinate system. Again, the shape of each robot may vary in a time dependent manner. For placing the robot at the point x_0 we compute $\varrho_r[x/x - x_0]$.

We assume that there is a time interval $T = [T_0, T_\infty]$ with $0 \leq T_0 < T_\infty \in \mathbb{R}$ specified. This time interval fixes the period of our modeling.

For each robot r there is a starting position $\xi_0^{(r)}$ and a final position $\xi_\infty^{(r)}$ specified. A trajectory is a continuous map $p : T \rightarrow \mathbb{R}^s$. Such a trajectory determines both a path in the space and the velocity of the object moving along this path.

The aim of multiple object motion planning is to find trajectories $p^{(r)}$ for each robot $r \in R$ such that the following conditions hold:

- (i) For all points $\xi \in \mathbb{R}^s$, for all robots r , and for all times $\tau \in [T_0, T_\infty]$ we have

$$\mathbb{R} \models \varrho_r(\tau, \xi - p_r(\tau)) \implies \sigma(\tau, \xi).$$

In other words, all points belonging to the robot r at position $p_r(\tau)$ at the time τ belong to the environment σ at time τ , i.e. the robot is actually located in the environment.

- (ii) For all points $\xi \in \mathbb{R}^s$, for all robots r , and for all times $\tau \in [T_0, T_\infty]$ we have

$$\mathbb{R} \models \varrho_r(\tau, \xi - p_r(\tau)) \implies \neg \left(\bigvee_{R \ni s \neq r} \varrho_s(\tau, \xi - p_r(\tau)) \right).$$

In other words, all points belonging to the robot r at position $p_r(\tau)$ at the time τ do not belong to one of the robots ϱ_s at time τ , i.e. there is no collision of two robots.

- (iii) For each robot r we have $p(T_0) = \xi_0^{(r)}$ and $p(T_\infty) = \xi_\infty^{(r)}$. This means that the robots move actually from their start position to their end position during the time interval considered, $[T_0, T_\infty]$.

As discussed above, we want to do motion planning by linear quantifier elimination. The fact that we use quantifier elimination and in particular linear real quantifier elimination enforces some restrictions.

1. We consider only piecewise linear trajectories.
2. The number of pieces of all trajectories is fixed.
3. The directions and speeds of all robots are taken from a fixed finite set.
4. The environment is described by a linear formula.
5. All robots are given by linear formulas.

The last three restrictions are only imposed by the fact that we use a quantifier elimination restricted to linear formulas. Likewise this imposes the restriction that all pieces of the trajectories are linear. For a general quantifier elimination method, we could instead allow trajectories composed piecewise from semi-algebraic curves.

3.2. Formula construction

In the sequel we discuss how to transform the formal specification described above into a linear first-order formula.

Let σ be, as above, the description of the environment and let for $r \in R$ ϱ_r be the description of the robots. Let Δ_r be a finite set of direction vectors for the robot r . Each $\delta \in \Delta$ is of the form $\delta = (\delta_x, \delta_y) \in \mathbb{R}^2$ for $s = 2$ or is of the form $\delta = (\delta_x, \delta_y, \delta_z)$ for $s = 3$, describing a direction and a speed of the robot. In one time unit the specific robot possibly travels along the direction vector δ and covers a distance of $\sqrt{\delta_x^2 + \delta_y^2}$ or $\sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2}$, respectively. This means,

to be more precise, that each δ is both a direction and a velocity vector. In the following we will nevertheless use the term direction vector.

Recall from the above that we consider only piecewise linear trajectories. We call the points on which two pieces adjoin, glue points. Note that for each trajectory the number of glue points is not fixed, because each piece can always be considered to consist of two pieces. However, we consider, of course, only trajectories with a finite set of pieces and thus with a finite set of glue points. By a suitable refinement of all trajectories we can assume wlog that all trajectories consist of exactly n pieces glued together at time points t_1, \dots, t_{n-1} .

The key idea for semilinear motion planning (Weispfenning, 2001a) and for semilinear motion planning among moving objects (Weispfenning, 2001b) was the computation of the free space for the robot. Recall that in these approaches we consider only one robot in a possible time dependent environment.

The intention of introducing the free space is to reduce the shape of a robot to one singular point while simultaneously coding its original shape into the form of the environment. More precisely, the free space of one robot is the set of all points at which the robot can be positioned. If we have only one robot described by ϱ then the free space is given as a semi-algebraic set defined by the following first-order formula:

$$\forall x(\varrho[X/(x - x_0)] \longrightarrow \sigma).$$

Since ϱ and σ are linear it is clear that this formula is linear, too. So we can apply our linear quantifier elimination and compute a quantifier-free equivalent in the variable x_0 which is also linear. Note that the free space depends on the time t because both σ and ϱ can depend on the time t .

This concept cannot be easily generalized to the case of several robots because in this case the free space of each robot depends on the position of each other robot and this is not known in advance. As a solution we can position all robots on parametric positions, say robot r on $x^{(r)}$, and compute a formula $\hat{\sigma}(t, x^{(1)}, \dots, x^{(m)})$ which is true if, at a given time t , the positions of all robots are valid. This means that all robots are completely inside the environment and no two robots collide. Using quantifiers $\hat{\sigma}(t, x^{(1)}, \dots, x^{(m)})$, this can be expressed by $\hat{\sigma}_1 \wedge \dots \wedge \hat{\sigma}_m$, where each $\hat{\sigma}_r$ is defined as

$$\forall x \left(\varrho_r[x/(x - x^{(r)})] \longrightarrow \left(\sigma \wedge \bigwedge_{s \in R \setminus \{r\}} \neg \varrho_s[x/x - x^{(s)}] \right) \right).$$

As above, this formula is linear and we can compute a quantifier-free equivalent which is in turn also linear. For simplicity, we denote this quantifier-free equivalent again by $\hat{\sigma}(t, x^{(1)}, \dots, x^{(m)})$.

In the next step we formulate a valid movement of all robots with respect to given direction vectors from one point to another. The formula constructed is true if each robot $r \in R$ can move from $X^{(r)}$ along the direction vector $d^{(r)}$ between t and t' . This can be expressed by the following formula:

$$\forall s \left(0 \leq s \leq (t' - t) \longrightarrow \hat{\sigma}(t, x^{(1)} + s \cdot d^{(1)}, \dots, x^{(m)} + s \cdot d^{(m)}) \right).$$

This formula is not linear in the set V of all variables because it contains products $s \cdot d^{(r)}$ but it is linear in the set $V \setminus \{d^{(r)} \mid r \in R\}$. Thus we can compute a quantifier-free equivalent

$$\gamma(t, t', x^{(1)}, d^{(1)}, x^{(2)}, d^{(2)}, \dots, x^{(m)}, d^{(m)}),$$

which is linear in $V \setminus \{d^{(r)} \mid r \in R\}$. This feature, that γ may be not linear in all variables, is why we have to fix a set of possible direction vectors for each robot.

From γ we can easily compute the formula

$$\Gamma(t, t', x^{(1)}, x^{(1)'}, d^{(1)}, \dots, x^{(m)}, x^{(m)'}, d^{(m)}) \equiv \gamma \wedge \bigwedge_{r \in R} (x^{(r)'} = x^{(r)} + (t - t') \cdot d^{(r)}).$$

This states that the robot 1 can move from $x^{(1)}$ to $x^{(1)'}$ along the direction vector $d^{(1)}$ between t and t' while the other robots can move from their source points to their target points along their direction vectors in the same time range from t to t' .

In the next step we eliminate the occurrences of the $d^{(r)}$'s by substituting the given values for the variables. The possible choice of the direction is translated into a disjunction. We have firstly to compute the Cartesian product $\Delta = \Delta_1 \times \dots \times \Delta_m$, because we can independently choose the directions for all robots. Recall that each Δ_r is the set of possible directions vectors for the robot r . The result formula π is then

$$\pi(t, t', x^{(1)}, x^{(1)'}, \dots, x^{(m)}, x^{(m)'}) \equiv \bigvee_{(\delta_1, \dots, \delta_m) \in \Delta} \Gamma[d^{(1)}/\delta_1, \dots, d^{(m)}/\delta_m].$$

This formula is again linear in all its variables because the possibly non-linear occurrences of the $d^{(r)}$ are eliminated by the substitution. The formula π states the possibility of a simultaneous movement of all robots along one piece of their trajectories.

In the next step we combine conjunctively the trajectory segments given by appropriate substitutions of π into formulas describing the total trajectory:

$$\Pi \equiv \bigwedge_{i=0}^{n-1} \pi(t^i, t^{i+1}, x_i^{(1)}, x_{i+1}^{(1)}, \dots, x_i^{(m)}, x_{i+1}^{(m)}).$$

We have to add constraints stating that the variables representing the time are strictly monotone increasing. Next we substitute $x_0^{(r)}$ by $\xi_0^{(r)}$, $x_n^{(r)}$ by $\xi_\infty^{(r)}$ for $r \in R$, t_0 by τ_0 and t_n by τ_∞ in the resulting formula. This results in the following formula M :

$$\left(\bigwedge_{i=0}^{n-1} t_i < t_{i+1} \wedge \Pi \right) [t_0/\tau_0, t_n/\tau_\infty, x_0^{(1)}/\xi_0^{(1)}, \dots, x_0^{(m)}/\xi_0^{(m)}, x_n^{(1)}/\xi_\infty^{(1)}, \dots, x_\infty^{(m)}/\xi_\infty^{(m)}].$$

Finally we have to quantify the variables $x_j^{(r)}$ and t_j with $r \in R$ and $j \in \{1, \dots, n - 1\}$ existentially. Note that these variables are after the substitution the only remaining variables, i.e. the resulting formula M is parameter free. To be precise, M states only that there is a trajectory for each robot fulfilling all our restrictions. In the next section we discuss how to compute the trajectories of a solution.

3.3. Solution computation

In the previous section we have explained in detail how to construct a formula describing the motion planning problem. The constructed formula consists of two parts: a block of existential quantifiers and a quantifier-free matrix. We can thus apply extended quantifier elimination to the formula M . Because M is parameter free all guards have to be also parameter free. Because parameter-free formulas can easily be evaluated to truth values we have to distinguish only two cases. In the first case, the quantifier-free equivalent computed as the disjunction of all guards

is false. This means that our motion planning problem cannot be solved. In the second case we have at least one guard which is true. In this case the result of the extended quantifier elimination contains also equations $t_i = \tau_i$ and $x_i^{(r)} = \xi_j^{(r)}$ with $r \in R$ and $j \in \{1, \dots, n - 1\}$. Recall that from the definition of extended quantifier elimination it follows that M holds at the point given by the right-hand sides of these equations. In other words, the $x_j^{(r)}$ define the glue points of the trajectories and the τ_j define the times of the glue points. From this information one can easily construct the trajectories p_r : the robot $r \in R$ moves from $\xi_i^{(r)}$ to $\xi_{i+1}^{(r)}$ starting at τ_i ending at τ_{i+1} along the direction vector $(\xi_{i+1} - \xi_i)/(\tau_{i+1} - \tau_i)$.

3.4. Options and alternatives

We would like to remark that from a theoretical point of view there is no necessity to apply quantifier elimination during the formula construction. This allows us, in principal, to construct systematically one input formula. The resulting input formula can thus be constructed more quickly and it is smaller as compared to the approach we discussed above. On the other hand, the computation time increases, because we systematically copy quantified subformulas.

One of the main factors for the complexity is the substitution of the direction vectors. Recall that we have to compute the Cartesian product of all $\Delta = \Delta_1 \times \dots \times \Delta_m$. This is actually one of the main sources for the extended running times of our demonstration system; cf. 5.2 and Section 6. Each reduction of only one of the Δ_r reduces the computation time.

To support the reduction we provide the following option for the generation of the formula, in particular, for the generation of π and Π . We allow the user to specify for each of the n pieces of the trajectories of a robot r its own set $\Delta_{n,r}$.

For this we define $\Delta_n = \Delta_{n,1} \times \dots \times \Delta_{n,m}$ and

$$\pi_n(t, t', x^{(1)}, x^{(1)'}, \dots, x^{(m)}, x^{(m)'}) \equiv \bigvee_{(\delta_1, \dots, \delta_m) \in \Delta_n} \Gamma[d^{(1)}/\delta_1, \dots, d^{(m)}/\delta_m].$$

The formula Π is then defined as $\bigwedge_{i=0}^{n-1} \pi_n(t^i, t^{i+1}, x_i^{(1)}, x_{i+1}^{(1)}, \dots, x_i^{(m)}, x_{i+1}^{(m)})$. There is no need to change any other part of our model or of the solution computation.

There is finally another generalization of the concept of the free space. In the formulation above we have replaced the free space by a formula that is true if the positions of all robots are valid. More related to the original approach of the free space, a free space for each robot can be computed separately. Suppose we want to compute the free space of the robot $r \in R$; then we place all robots $s \in R \setminus \{r\}$ on parametric positions inside in the environment. Unlike in our original approach we do not require here that these parametric positions are valid. This approach leads to the computation of $\hat{\sigma}_r$'s instead of $\hat{\sigma}$. The generation of γ has then to be adapted to the new situation. After the computation of γ we can proceed as usual.

4. Quantifier elimination by virtual substitution

The virtual substitution method dates back to a theoretical paper by the second author (Weispfenning, 1988). During the last thirteen years a lot of theoretical work has been done to improve the method; cf. Loos and Weispfenning (1993), Weispfenning (1994), Dolzmann and Sturm (1997b), Weispfenning (1997) and Dolzmann et al. (1998).

The applicability of the method in the form described here is restricted to formulas in which the quantified variables occur at most quadratically. For our purposes, however, linear quantifier

elimination is sufficient. Quantifiers are eliminated one by one, and the elimination of one quantifier can increase the degree of other quantified variables. Nevertheless, if the input is linear in some set V of variables the output is again linear in V . Thus, this method provides actually a linear quantifier elimination as indicated above. We will thus restrict our presentation to the case on linear formulas.

For eliminating the quantifiers from an input formula

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

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}, \dots, u_k are actually x_i quantified from further outside.

We fix real values ξ_1, \dots, ξ_k for the parameters u_1, \dots, u_k . Then all polynomials occurring in ψ^* become linear univariate polynomials in x with real coefficients. So the set

$$M = \{\zeta \in \mathbb{R} \mid \psi^*(\xi_1, \dots, \xi_k, \zeta)\}$$

of all real values ζ 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 real zeros of the linear polynomials occurring in ψ^* . Candidate terms $\alpha_1, \dots, \alpha_r$ for the zeros can be computed uniformly in u_1, \dots, u_k using the solution formulas for linear equations.

If all inequalities in ψ^* are weak, then all the intervals constituting M will, in 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 polynomial equation or inequality is evaluated in the obvious sense. The substitution of expressions in u_1, \dots, u_k of the form a/d among the α_j can be rewritten in such a way that all denominators involving the u_i are removed from the result; cf. Weispfenning (1997). 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 our candidates for points in M expressions of the form $\alpha \pm \varepsilon$, where α is a 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 polynomial 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. Weispfenning (1997). 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. Dolzmann and Sturm (1997b) for details.

Recall that the well-known solution formula for linear equations $ax + b = 0$ requires $a \neq 0$. In our situation a is a term in u_1, \dots, u_k , so $a \neq 0$ can in general not be decided uniformly but depends on the interpretation of the u_i . Thus a linear polynomial $ax + b$ delivers an expression $\alpha = -b/a$ as a candidate solution, which requires $a \neq 0$. Let t be the candidate point for M obtained from α by possibly adding or subtracting ε . With the substitution of t into ψ^* , it is necessary to add the conditions on the non-vanishing of a . Formally, we obtain

$$a \neq 0 \wedge \psi^*[x/t].$$

Suppose we have eliminated an existential quantifier. Then we have in general obtained a disjunction $\psi'_1 \vee \dots \vee \psi'_r$. If the next quantifier which is to be eliminated is also an existential one, then we make use of the equivalence

$$\exists x_{n-1}(\psi'_1 \vee \dots \vee \psi'_r) \longleftrightarrow \exists x_{n-1}(\psi'_1) \vee \dots \vee \exists x_{n-1}(\psi'_r)$$

eliminating all $\exists x_{n-1}(\psi'_j)$ independently. As a consequence, no candidate solutions which are 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, as has been shown by Weispfenning (1988).

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. Loos and Weispfenning (1993) and Weispfenning (1997). One simple instance for such an improvement is the following natural extension of *Gauss elimination*: Suppose ψ^* is of the form

$$ax + b = 0 \wedge \psi^{**},$$

where at least one of the coefficient terms a, b is a rational non-zero constant. Then we know that under any interpretation of the u_i the equation is *non-trivial*, i.e. different from $0 = 0$. Hence the only test candidate required is $-b/a$ substituted, of course, with the condition $a \neq 0$. No additional test candidates arising from equations or inequalities in the remainder ψ^{**} of ψ^* need be considered. This idea can easily be extended to a quadratic equation instead of a linear one, taking into account again the discriminant. The first author has systematically analyzed improvements of the method gaining much practical efficiency (Dolzmann, 2000).

An extended quantifier elimination can be straightforwardly derived from this method by not constructing a disjunction at the end. Instead all the quantifier-free substitution results are kept separately as guards together with the candidate terms yielding them. The computational overhead for the extended quantifier elimination in contrast to the regular one is negligible. Note that candidate terms can contain both fractions and non-standard symbols such as infinitesimals. While fractions can be easily interpreted in the field of real numbers, there is no simple way to interpret the infinitesimals. There is, of course, always a solution without infinitesimals if there is a solution containing infinitesimals. For a parameter-free formula such a solution can be obtained by computing small numbers for the ε 's and large numbers for the ∞ 's. For parametric candidate terms there is no simple method for obtaining such numbers. Moreover, the concrete values may depend on the parameters. We would like to remark that the original linear quantifier elimination presented by the second author does not introduce non-standard symbols but has less practical efficiency.

The notion of *virtual substitution* refers to both adding conditions and resolving fractions or non-standard subterms such as infinitesimals. The complexity of this method depends on the number of quantified variables and, even more so, on the number of quantifier changes. In theory, parameters play a minor role in the complexity. They turn out in fact to be very cheap in practice, too.

5. Implementation

5.1. REDLOG

REDLOG is a package extending the computer algebra system REDUCE to a computer logic system. It provides a working environment for first-order logic over various languages and theories; cf. Dolzmann and Sturm (1997a). Besides many other algorithms it contains a very

```

{
  area(125,125),
  obstacle {rectangle(0,0,45,45),pt(40,40)},
  moving_obstacle {rectangle(0,0,25,10),pos(95,85,0),
    move(0,95,-1,0),id(1)},
  robot {rectangle(0,0,15,15),from(0,0,0),to(105,105,250),
    dv(1,0),dv(0,1),dv(0,0),id(7)},
  robot {rectangle(0,0,10,10),from(105,105,0),to(0,0,250),
    dv(-1,0),dv(0,-1),dv(0,0),id(9)}
};

```

Fig. 1. Sample input.

sophisticated implementation of the (extended) linear quantifier elimination discussed in the above section. REDLOG is contained in the current release of REDUCE.

5.2. REDLOG based motion planning

Using REDLOG we have implemented a demonstration system for multiple object semilinear motion planning. This implementation shows that a user does not need to have any idea of formulas or quantifier elimination. Moreover, we have used the implementation for tuning the formulation of the problem as a first-order formula. The complete logical background is hidden behind an algebraic user specification system and a graphical result presentation. For further processing, the result can additionally be output as a list. There is no difficulty in designing a complete graphical user interface or a system expecting XML input and generating XML output.

Our demonstration system is currently restricted to the two-dimensional case. In our description of our algorithm we have assumed that the shape of the environment and the shape of the robots are given as first-order formulas. This is on one hand a very general approach but it is on the other hand not comfortable for the user nor does it show the possible scope of applications. In our system both environment and robots are described by unions of base objects. Base objects are either rectangular regions or arbitrary convex polygons, given as the convex hulls of sets of points. For rectangular regions we can currently generate more efficient formulas than for arbitrary convex regions.

The environment is divided into two parts: the fixed environment and the dynamic environment. The fixed environment is also divided into two subparts. Firstly, it is described by a rectangular boundary located at the origin with a user defined fixed size. Secondly, additional objects are placed in this environment as obstacles for the robots. The name fixed environment refers to the fact that its description does not depend on the time.

In contrast to the fixed environment the dynamic environment depends on the time. In our case, the dynamic environment consists of a set of so called moving obstacles. These are a second set of robots. These robots may move around in the environment, their trajectories are known in advance and they cannot be controlled by our motion planning system. As any other object, the moving obstacles are also described by a union of base objects. The trajectories for the moving obstacles are given as a set of time intervals together with a direction and velocity vector. Thus they are piecewise linear.

We use a simple list notation for inputting the description. Fig. 1 shows the input of the example discussed in the next section. Our demonstration system computes from such a list description the input formula for the extended quantifier elimination as described in Section 3.

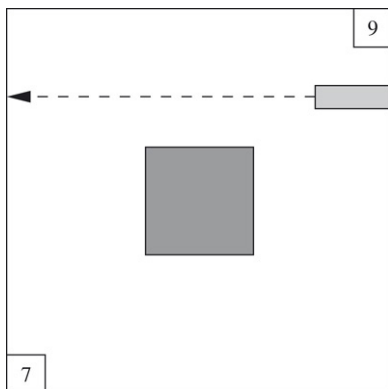


Fig. 2. The situation.

We apply the regular quantifier elimination for constructing the subformulas as early as possible. After the construction of the formula M we apply to M the extended quantifier elimination provided by REDLOG, which is based on the quantifier elimination by virtual substitution. As indicated in Section 4 the result may contain infinitesimals, namely some ε 's. Assuming that for practical purposes there has to be always a safety distance between robots and other objects one can simply neglect these infinitesimals, i.e. replace them with 0. In our test computations we have observed that there are no infinitesimals in the output if the input is slightly modified. In our models all formulas σ and q_r contain only weak ordering relations \leq and \geq . Instead of later assuming the infinitesimal safety distance this distance might be introduced in the input in the following way. For the computation of $\hat{\sigma}$ we compute $\neg q_r$. Instead of computing q_r from our description and then negating the formula we can also compute $\neg q_r$ directly from the description. We modify this computation in such a way that the resulting formula contains also only weak ordering relations. This means that the border of the robot belongs both to the robot and to the environment which can be the case if we have an infinitesimal safety distance as discussed above. Then our quantifier elimination does not need to introduce infinitesimals which remain in the output. This leads, in general, to faster computations times.

Our demonstration system provides two commands for the solution computation both operating on the list input, i.e. the formula generation is completely hidden from the user. The first one computes a list representation of all trajectories. The second one generates a MAPLE animation of both the environment and the robots moving along their trajectories.

6. Example

In this section we present an example computation of our test implementation. The input of Fig. 1 describes the situation as shown in Fig. 2. The dark grey rectangle in the middle is a fixed obstacle. The light grey rectangle on the right-hand side is a moving obstacle. It moves between 0 and 80 from the right border to the left border with speed 1. The rectangles labeled with 7 and 9 are our two robots. They have to swap their positions. The time interval given is $T = [0, 255]$. The robot no. 7 can move rightwards and upwards with speed 1 and the robot no. 9 can move leftwards and downwards with speed 1. Both robots can always pause. We fix the number n , the maximal number of glue points, to 4. Our system solves this given situation in about 113 min. For all computations we have used an Intel Pentium III, 1 GHz, and 128 MB heap size for REDUCE. The solution is shown in Fig. 3.

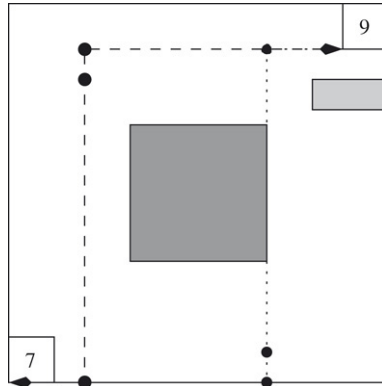


Fig. 3. One solution.

If we provide the correct directions for each step in our alternative formulation discussed in Section 3.4 we can compute the solution, or to be more precise the times of the glue points, in only 2 min. If we give the robots on each segment two alternatives, namely either to move in the correct direction or to pause, we can compute the solution in 3 min.

In the alternative formulation with several free spaces also discussed in Section 3.4 we can compute the solution in 92 min.

7. Conclusions

We have shown that multiple object motion planning among other movable objects can be solved by extended linear quantifier elimination with only some slight restrictions. Designing a demonstration system that completely hides the formula construction and quantifier elimination from the user we have shown that the user does not have to have mathematical background for applying our approach to motion planning problems. In the last few years it has turned out that there are many ways to improve quantifier elimination algorithms dramatically. In this respect it is fruitful to discover a wide range of applications as in our case for motion planning problems.

References

- Barraquand, Jérôme, Langlois, Bruno, Latombe, Jean-Claude, 1989. Robot motion planning with many degrees of freedom and dynamic constraints. In: *Preprints of the Fifth International Symposium of Robotic Research*.
- Dolzmann, Andreas, 2000. Algorithmic strategies for applicable real quantifier elimination. Doctoral Dissertation. Department of Mathematics and Computer Science, University of Passau, Germany, D-94030 Passau, Germany. March.
- Dolzmann, Andreas, Sturm, Thomas, 1997a. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin* 31 (2), 2–9.
- Dolzmann, Andreas, Sturm, Thomas, 1997b. Simplification of quantifier-free formulae over ordered fields. *Journal of Symbolic Computation* 24 (2), 209–231.
- Dolzmann, Andreas, Sturm, Thomas, Weispfenning, Volker, 1998. A new approach for automatic theorem proving in real geometry. *Journal of Automated Reasoning* 21 (3), 357–380.
- Latombe, Jean-Claude, 1991. *Robot Motion Planning*. Kluwer Academic Publishers, Boston.
- Loos, Rüdiger, Weispfenning, Volker, 1993. Applying linear quantifier elimination. In: *Computational Quantifier Elimination*. *The Computer Journal* 36 (5), 450–462 (special issue).
- Schwartz, Jacob T., Sharir, Micha, 1983. On the piano movers' problem: Iii. coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *International Journal of Robotics Research* 2 (3), 46–75.

- Sharir, Micha, Sifrony, Shmuel, 1988. Coordinated motion planning for two independent robots. In: Proceedings of the Fourth ACM Symposium on Computational Geometry.
- Weispfenning, Volker, 2001a. Semilinear motion planning in REDLOG. *Applicable Algebra in Engineering, Communication and Computing* 12, 455–475.
- Weispfenning, Volker, 2001b. Semilinear motion planning among moving objects in REDLOG. In: Ganzha, V.G., Mayr, E.W. (Eds.), *Computer Algebra in Scientific Computing. Proceedings of the CASC 2001*. Springer, Berlin, pp. 541–553.
- Weispfenning, Volker, 1988. The complexity of linear problems in fields. *Journal of Symbolic Computation* 5 (1–2), 3–27.
- Weispfenning, Volker, 1994. Quantifier elimination for real algebra—the cubic case. In: Proceedings of the International Symposium on Symbolic and Algebraic Computation. ISSAC 94, Oxford, England. ACM Press, New York, pp. 258–263.
- Weispfenning, Volker, 1997. Quantifier elimination for real algebra—the quadratic case and beyond. *Applicable Algebra in Engineering Communication and Computing* 8 (2), 85–101.