Extending Real Quantifier Elimination by Cylindrical Algebraic Decomposition to Get Answers

Andreas Seidl

University of Passau, Germany
seidl@fmi.uni-passau.de
http://www.fmi.uni-passau.de/~seidl/

Abstract. We show how the cylindrical algebraic decomposition (CAD) method for real quantifier elimination (QE) can be extended to provide answers, i.e. sample solutions for variables of a leading block of existential quantifiers, thus providing more and interesting information. In the general case, where there are free variables, these answers are in general parametric. For the dual case, i.e. for formulas with leading universal block of quantifiers, we get parametric counter-examples. The main benefit over a virtual substitution (VS) based approach is that there are no degree restrictions on the input.

1 Introduction

Real quantifier elimination (QE) is a versatile tool for solving a wide range of problems. Such problems are modeled as a first-order formula over the language of ordered rings. Real QE finds for a given formula a simpler quantifier-free one, which is equivalent over the reals, thus solving the problem. Applications include analysis of partial differential equations [9], control theory [1, 11], theoretical mechanics [10], motion planning [20, 19, 7], diagnosis of electrical networks [15], computer aided design and real implicitation [16, 14].

There exists a variety of methods for real QE. Methods that have been implemented *and* are available are *cylindrical algebraic decomposition* [3], *virtual substitution of test points* [17] and a method based on parametric real root counting [4], which is now called *Hermitian* QE [8]. All these methods are implemented and available within the computer logic system REDLOG [5], which is part of the computer algebra system REDUCE.

It has turned out that the virtual substitution (VS) method and the cylindrical algebraic decomposition (CAD) method complement each other. The VS method can cope with many parameters, but has degree restrictions. The CAD method, in contrast, has no degree restrictions, but parameters contribute to the complexity in the same way as quantified variables. For practical purposes it has turned out that having both methods available and combining them [12] results in most fast and successful [13] computations for very large problems.

For the virtual substitution method, an interesting extension has been developed [18]. This method was extended to give sample parametric answers for values of the variables of a leading block of existential quantifiers. This has turned out to be very useful in practice, e.g. for error diagnosis in electrical networks [15] and for collision problems [14].

This paper deals with extending the CAD method for real QE to give, next to a quantifier-free equivalent formula, answers. In this paper:

- We motivate that sometimes one would like to get more information out of QE than just an equivalent formula, e.g. sample values for certain variables.
- We demonstrate that prior progress was made to extend the virtual substitution method for real QE to give answers. Due to limitations on the VS method, however, QE with answers was so far not applicable to all problems over the reals.
- We introduce the CAD method for real quantifier elimination and show how the CAD method can be exploited to provide answers.
- By extending the CAD method we lift the limitations on the applicability of extended quantifier elimination.
- We explain how, by duality, one can get parametric counter-examples for formulas, which have an outermost block of universal quantifiers.

2 Motivating Examples and Prior Work

We look at some examples to give an impression on how real quantifier elimination can be used to model and solve a problem, to demonstrate prior progress made to extend the virtual substitution method to give answers, and to motivate the suggested improvement.

2.1 Solving a Tangram Style Puzzle

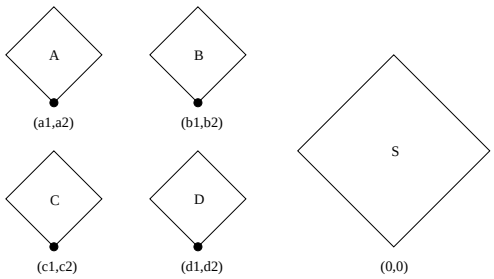

Fig. 1. Four small squares (left) and a big square (right)

Consider the following puzzle. There are four small squares A, B, C, D of diameter 2 and one square of diameter 4. The squares are arranged as can be seen in Figure 1. Every edge is parallel to the bisecting line of the first and third, or, of the second and fourth quadrant. More precisely, the position of, say, the square A in the real plane is given by the corner points (a_1, a_2), (a_1-1, a_2+1), (a_1, a_2+2) and (a_1+1, a_2+1). The position of the big square S is given by $(0,0)$, $(-2,2)$, $(0,4)$ and $(2,2)$. The challenge is to decide, whether the small squares fit by translation into the big one without overlapping each other.

Let us formulate this problem in first-order logic. First of all, we need a formula describing whether a point (x_1, x_2) lies within the small square A. Its relative position is given by (a_1, a_2).

$$\begin{aligned}\varphi_A \equiv\ & a1 + a2 - x1 - x2 + 2 > 0 \wedge a1 + a2 - x1 - x2 < 0 \wedge \\ & a1 - a2 - x1 + x2 - 2 < 0 \wedge a1 - a2 - x1 + x2 > 0\end{aligned}$$

Similarly, one finds formulas φ_B, φ_C, and φ_D. For the big square S

$$\varphi_S \equiv x1 + x2 - 4 < 0 \wedge x1 + x2 > 0 \wedge x1 - x2 + 4 > 0 \wedge x1 - x2 < 0$$

is appropriate. Note that we have to consider *open* objects without border.

The squares A, B, C, D lie within S, for fixed (a_1, a_2), (b_1, b_2), (c_1, c_2), (d_1, d_2), iff

$$(\varphi_A \vee \varphi_B \vee \varphi_C \vee \varphi_D) \longrightarrow \varphi_S$$

for all (x_1, x_2). Furthermore, the squares A, B, C, D do not overlap each other, iff

$$\neg(\varphi_A \wedge \varphi_B) \wedge \neg(\varphi_A \wedge \varphi_C) \wedge \neg(\varphi_A \wedge \varphi_D) \wedge \neg(\varphi_B \wedge \varphi_C) \wedge \neg(\varphi_B \wedge \varphi_D) \wedge \neg(\varphi_C \wedge \varphi_D)$$

Altogether the problem can be stated as follows:

$\varphi :\equiv \exists a_1 \exists a_2 \exists b_1 \exists b_2 \exists c_1 \exists c_2 \exists d_1 \exists d_2 \forall x_1 \forall x_2 (\neg(a_1 + a_2 - x_1 - x_2 + 2 > 0 \wedge a_1 + a_2 - x_1 - x_2 < 0 \wedge a_1 - a_2 - x_1 + x_2 - 2 < 0 \wedge a_1 - a_2 - x_1 + x_2 > 0 \wedge b_1 + b_2 - x_1 - x_2 + 2 > 0 \wedge b_1 + b_2 - x_1 - x_2 < 0 \wedge b_1 - b_2 - x_1 + x_2 - 2 < 0 \wedge b_1 - b_2 - x_1 + x_2 > 0) \wedge \neg(a_1 + a_2 - x_1 - x_2 + 2 > 0 \wedge a_1 + a_2 - x_1 - x_2 < 0 \wedge a_1 - a_2 - x_1 + x_2 - 2 < 0 \wedge a_1 - a_2 - x_1 + x_2 > 0 \wedge c_1 + c_2 - x_1 - x_2 + 2 > 0 \wedge c_1 + c_2 - x_1 - x_2 < 0 \wedge c_1 - c_2 - x_1 + x_2 - 2 < 0 \wedge c_1 - c_2 - x_1 + x_2 > 0) \wedge \neg(a_1 + a_2 - x_1 - x_2 + 2 > 0 \wedge a_1 + a_2 - x_1 - x_2 < 0 \wedge a_1 - a_2 - x_1 + x_2 - 2 < 0 \wedge a_1 - a_2 - x_1 + x_2 > 0 \wedge d_1 + d_2 - x_1 - x_2 + 2 > 0 \wedge d_1 + d_2 - x_1 - x_2 < 0 \wedge d_1 - d_2 - x_1 + x_2 - 2 < 0 \wedge d_1 - d_2 - x_1 + x_2 > 0) \wedge \neg(b_1 + b_2 - x_1 - x_2 + 2 > 0 \wedge b_1 + b_2 - x_1 - x_2 < 0 \wedge b_1 - b_2 - x_1 + x_2 - 2 < 0 \wedge b_1 - b_2 - x_1 + x_2 > 0 \wedge c_1 + c_2 - x_1 - x_2 + 2 > 0 \wedge c_1 + c_2 - x_1 - x_2 <$

$0 \wedge c_1 - c_2 - x_1 + x_2 - 2 < 0 \wedge c_1 - c_2 - x_1 + x_2 > 0) \wedge \neg(b_1 + b_2 - x_1 - x_2 + 2 > 0 \wedge b_1 + b_2 - x_1 - x_2 <$
$0 \wedge b_1 - b_2 - x_1 + x_2 - 2 < 0 \wedge b_1 - b_2 - x_1 + x_2 > 0 \wedge d_1 + d_2 - x_1 - x_2 + 2 > 0 \wedge d_1 + d_2 - x_1 - x_2 <$
$0 \wedge d_1 - d_2 - x_1 + x_2 - 2 < 0 \wedge d_1 - d_2 - x_1 + x_2 > 0) \wedge \neg(c_1 + c_2 - x_1 - x_2 + 2 > 0 \wedge c_1 + c_2 - x_1 - x_2 <$
$0 \wedge c_1 - c_2 - x_1 + x_2 - 2 < 0 \wedge c_1 - c_2 - x_1 + x_2 > 0 \wedge d_1 + d_2 - x_1 - x_2 + 2 > 0 \wedge d_1 + d_2 - x_1 - x_2 <$
$0 \wedge d_1 - d_2 - x_1 + x_2 - 2 < 0 \wedge d_1 - d_2 - x_1 + x_2 > 0) \wedge (((a_1 + a_2 - x_1 - x_2 + 2 > 0 \wedge a_1 + a_2 - x_1 - x_2 <$
$0 \wedge a_1 - a_2 - x_1 + x_2 - 2 < 0 \wedge a_1 - a_2 - x_1 + x_2 > 0) \vee (b_1 + b_2 - x_1 - x_2 + 2 > 0 \wedge b_1 + b_2 - x_1 - x_2 <$
$0 \wedge b_1 - b_2 - x_1 + x_2 - 2 < 0 \wedge b_1 - b_2 - x_1 + x_2 > 0) \vee (c_1 + c_2 - x_1 - x_2 + 2 > 0 \wedge c_1 + c_2 - x_1 - x_2 <$
$0 \wedge c_1 - c_2 - x_1 + x_2 - 2 < 0 \wedge c_1 - c_2 - x_1 + x_2 > 0) \vee (d_1 + d_2 - x_1 - x_2 + 2 > 0 \wedge d_1 + d_2 - x_1 - x_2 <$
$0 \wedge d_1 - d_2 - x_1 + x_2 - 2 < 0 \wedge d_1 - d_2 - x_1 + x_2 > 0)) \longrightarrow (x_1 + x_2 - 4 < 0 \wedge x_1 + x_2 > 0 \wedge x_1 - x_2 + 4 >$
$0 \wedge x_1 - x_2 < 0)))$.

Generating such formulas by hand can be a tedious and error-prone task. Therefore the author has implemented a package TANGRAM. This package was written to aid the formulation of tangram style problems. It allows to easily create formulas describing convex objects, which have a polygonal outline. In addition, based on already defined shapes, a formula saying that certain small shapes fit into a bigger shape without overlapping each other, can be produced. The above formulas were generated by this software.

Let us now apply QE to this problem formulation φ. We derive *true* with the virtual substitution method. Although this is the correct answer, we wish to get more information. Where do we have to place the small shapes? This is where *extended* QE, or QE *with answers* comes into the game.

For the virtual substitution method, prior research [18] has shown that for the outermost block of quantifiers one can get sample values in the existential case or counter-examples in the universal case for the according variables. For our example, the virtual substitution method yields *true* and the sample values

$$a_1 = 0\ ,\ a_2 = 2\ ,\ b_1 = 0\ ,\ b_2 = 0\ ,\ c_1 = 1\ ,\ c_2 = 1\ ,\ d_1 = -1\ ,\ d_2 = 1.$$

2.2 A Parametric Example

Variables, which are not within the scope of a quantifier, are called *free variables* or *parameters*. The preceding example was a *closed* formula, i.e. a formula without free variables. For such an example we can get concrete values as answers. If there are parameters such answers will have to be parametric as well in general.

Past research for the virtual substitution method showed that there is a natural way for this method to deliver the parametric answers [18], thus this method is not restricted to decision problems.

Consider the following example. The formula $\varphi :\equiv \exists x(ax^2 + bx + 1 = 0)$ asks for conditions on the parameters a, b such that the quadratic polynomial $ax^2 + bx + 1$ has a real root. QE by VS with answers returns three guarded points:

$$\left(4a - b^2 \leq 0 \wedge a \neq 0, \left(x = \frac{-\sqrt{-4a + b^2} - b}{2a}\right)\right)$$
$$\left(4a - b^2 \leq 0 \wedge a \neq 0, \left(x = \frac{+\sqrt{-4a + b^2} - b}{2a}\right)\right)$$
$$\left(a = 0 \wedge b \neq 0, \left(x = \frac{-1}{b}\right)\right).$$

This tells us not only that $4a - b^2 \leq 0 \wedge a \neq 0 \vee a = 0 \wedge b \neq 0$ is equivalent to $ax^2 + bx + 1$ having a real root. It tells us in addition what such a root x looks like in the two cases $4a - b^2 \leq 0 \wedge a \neq 0$ and $a = 0 \wedge b \neq 0$.

2.3 Finding Extraneous Points

Consider the parametric curve (f_1, g_1) with

$$f_1(t) = -6t^4 - 63, \quad g_1(t) = 92t^3 + 70t^2.$$

By computing the resultant ρ of $x - f_1$ and $y - g_1$ wrt. t one can get an implicit description of this parametric curve. Such an description is often not exact. More precisely, the graph

$$\{(x, y) \in \mathbb{R}^2 | \text{exists } t \in \mathbb{R} \text{ such that } x - f_1(t) = 0 \text{ and } y - g_1(t) = 0\}$$

of the parametric curve is often a proper subset of the real variety of $\{\rho\}$, i.e. the set

$$\{(x, y) \in \mathbb{R}^2 | \rho(x, y) = 0\}.$$

To get an *exact* real implicit representation, we apply real quantifier elimination to

$$\exists t(x = -6t^4 - 63 \text{ and } y = 92t^3 + 70t^2)$$

and derive

$$8954912x^3 - 1777440x^2y + 1710485868x^2 + 44100xy^2 - 223957440xy + 108895082184x + 27y^4 + 2778300y^2 - 7054659360y + 2310620648364 = 0 \text{ and } x + 63 \leq 0$$

To do this we need the CAD method, as the VS methods fails due to degree restrictions. The result is equivalent to

$$\rho = 0 \text{ and } x \leq -63$$

At this point, quantifier elimination with answers could help us to find possible extraneous points, which lie in the real variety of ρ, but not in the graph of the given parametric curve.

These examples make it obvious that for decision problems as well as for parametric problems getting answers out of quantifier elimination in addition to quantifer-free equivalents is highly desirable. Furthermore, getting answers from the CAD method is desirable in particular, due to limitations of the VS method, as the last example showed.

3 QE with Parametric Answers

We want to specify now what the result of QE with answers is. From now on we assume the input formula

$$\varphi(x_1, \ldots, x_k) \equiv \exists x_{k+1} \cdots \exists x_l \mathsf{Q}_{l+1} x_{l+1} \cdots \mathsf{Q}_r x_r \psi$$

to be in prenex normal form. The input formula has $x_1, \ldots, x_k$ as free and $x_{k+1}, \ldots, x_r$ as bound variables. For each $k + 1 \leq j \leq r$ the symbol Q_j denotes x_j's quantifier. We furthermore assume that Q_{l+1} is a universal quantifier. So $(x_{k+1}, \ldots, x_l)$ is the leading block of existentially quantified variables of maximal length.

3.1 Specification

The output of QE with answers is specified to be a finite set of guarded points [6]

$$\{(\psi'_i, (x_{k+1} = b_{i,k+1}, \ldots, x_l = b_{i,l}))|i \in I\}.$$

Here the ψ_i's are quantifier-free formulas, and $b_{i,j}$ is, for some i and $k + 1 \leq j \leq l$, a term which contains at most the variables $x_1, \ldots, x_{j-1}$. For this term a slight extension of the language of ordered rings will be needed.

A guarded point $(\psi, (x_{k+1} = b_{k+1}, \ldots, x_l = b_l))$ can be viewed to define a partial map $\gamma : \mathbb{R}^k \to \mathbb{R}^{l-k}$: Given values $(a_1, \ldots, a_k) \in \mathbb{R}^k$, such that $\mathbb{R} \models \psi(a_1, \ldots, a_k)$, we can successively compute $a_{k+1}, \ldots, a_l$ in the following way: Let us assume that $a_{k+1}, \ldots, a_j$ are already computed for a $k \leq j < l$, and now we want to compute a_{j+1}. We substitute in the term b_{j+1} the values $a_1, \ldots, a_j$ for the variables $x_1, \ldots, x_j$. This yields a term without free variables, which can be evaluated in $\mathbb{R}$ to a_{j+1}.

After detailing the syntactical definition, we specify how the semantics of the output of QE with answers should be. There are two conditions.

(C1) *Quantifier-free formula.* We want to easily construct from the set of guarded points a quantifier-free formula, which is equivalent to the input formula:

$$\mathbb{R} \models \varphi \leftrightarrow \bigvee_{i \in I} \psi_i'.$$

(C2) *Example solution.* For every guarded point $(\psi, (x_{k+1} = b_{k+1}, \ldots, x_l = b_l))$ in the output set and every $(a_1, \ldots, a_k) \in \mathbb{R}^k$ with $\mathbb{R} \models \psi(a_1, \ldots, a_k)$, we want to have

$$\mathbb{R} \models \mathsf{Q}_{l+1} x_{l+1} \cdots \mathsf{Q}_r x_r \psi(a_1, \ldots, a_l),$$

where $\gamma(a_1, \ldots, a_k) = (a_{k+1}, \ldots, a_l)$.

These two conditions motivate, why QE with answers is also called extended QE: The output of a quantifier-free formula, as provided by classical QE, is extended to provide in addition sample solutions.

4 Situation for the CAD Method

4.1 QE by Full CAD

The QE by CAD algorithm can be described as consisting of three phases. The following sketch is only intended to introduce notation. For a more detailed description see e.g. [3, 2]:

1. *Projection.* One starts out with the set $F_r = F$ of polynomials in r variables, which is extracted from the input formula. In $r-1$ projection steps there are $r-1$ further finite sets $F_{r-1}, \ldots, F_1$ of polynomials with one fewer variable in each step generated.
2. *Extension.* Based on the trivial decomposition D_0 of 0-space one successively constructs decompositions $D_1, \ldots, D_r$ of higher-dimensional spaces. For obtaining D_{i+1} there are sample points of D_i plugged into the polynomials of F_{i+1}. This yields univariate polynomials with real algebraic numbers as coefficients. Essentially, the roots of these polynomials and rational points between these roots extend the sample point of the underlying cell to give new sample points for cells in D_{i+1}
3. *Solution formula construction.* Truth values are computed for the leaf cells of the tree, i.e. the cells in D_r. Depending on the quantifiers, these values are propagated down to cells in $D_r, \ldots, D_k$. Based on signs of projection polynomials, a quantifier-free solution formula is constructed to describe the subset of $\mathbb{R}^k$ which is comprised of true cells.

4.2 QE by Full CAD with Answers

We now want to investigate if we can achieve similar results for the CAD method. Still the same assumptions on the form of the input formula φ are made as at the beginning of Section 3.

Let us furthermore assume that a full CAD tree for this problem has been constructed, and cells of $D_r, \ldots, D_k$ bear a truth value. For each cell $C \in D_k$ let δ_C denote a quantifier-free description of this cell. Such a formula exists, as each cell represents a semi-algebraic set. Let G denote the finite set, which is generated by collecting for each true cell $C^{(l)}$ in D_l a guarded point $(\delta_C, x_{k+1} = p_{k+1}, \ldots, x_l = p_l)$, Here $C \in D_k$ is the unique predecessor of $C^{(l)}$, and $p_{k+1}, \ldots, p_l$ are derived in the following way: Let $C, C^{(k+1)}, \ldots, C^{(l)}$ be the path from C to $C^{(l)}$ in the CAD tree and $k+1 \leq j \leq l$. To define p_j, we look how the last component s_j of $C^{(j)}$'s sample point $(s_1, \ldots, s_j)$ was generated. There are four cases.

1. There is a projection polynomial $f \in F_j$ such that s_j is the n-th root of $f(s_1, \ldots, s_{j-1}) \in \mathbb{A}[x_j]$. Then $p_j := \mathit{Root}(f, n)$.
2. There are projection polynomials $f, f' \in F_j$ such that s_j lies between the n-th root of the polynomial $f(s_1, \ldots, s_{j-1})$ and the m-th root of $f(s_1, \ldots, s_{j-1})$ and there exists no other root in between. Then $p_j := (\mathit{Root}(f, n) + \mathit{Root}(f, m))/2$.
3. If s_j is smaller than the smallest or greater than the biggest root, then $p_j := \mathit{Root}(f, 1) - 1$ or $p_j := \mathit{Root}(f, m) + 1$ for an appropriate $f \in F_j$ and integer m.

4. Otherwise, if there was no root at all, p_k can be simply defined as 0 or as the special symbol *arbitrary*.

After this set of guarded points is defined, we want to allow two ways to manipulate this set to make it more concise.

1. *Combine.* Two guarded points, which only differ in the first part, can be combined:

$$(\psi, b), (\psi', b) \mapsto (\psi \vee \psi', b).$$

2. *Simplify.* The first part of a guarded point can be replaced by an equivalent one.

In contrast to the virtual substitution method, the degree of the polynomials can be arbitrary. Thus we cannot rely on radicals to specify p_j. Instead we have to introduce the symbol *Root*. The delinability property of the projection set ensures *Root* to be well-defined.

Correctness of the Algorithm We need to check whether the conditions (C1) and (C2) are satisfied. As for (C1) a valid solution formula can be constructed as disjunction over formulas describing the true cells of D_k.

To see that (C2) holds, note that a full CAD tree D for

$$\mathsf{Q}_{l+1} x_{l+1} \cdots \mathsf{Q}_r x_r \psi$$

is essentially the same as for φ. The only difference is that cells on the levels $D_k, \ldots, D_{l-1}$ bear no truth value. If one now computes for a guarded point and for $(a_1, \ldots, a_k) \in \mathbb{R}^k$ the values $a_{k+1}, \ldots, a_l$, then the point $(a_1, \ldots, a_l)$ lies in a true cell of D_l. Hence (C2) holds.

Furthermore, if for a set of guarded points both (C1) and (C2) hold, then combining and simplifying points as described above will preserve these properties.

4.3 Examples

Second Example Revisited We revisit the example $\varphi :\equiv \exists x(ax^2 + bx + 1 = 0)$ from Subsection 2.2 to demonstrate the algorithm. Figure 4.3 shows the full CAD tree for this problem. Here $k = 2$, $l = 3$, and $r = 3$ with our notation. Let us denote the three cells of D_1 by C_1, C_2, C_3. We name

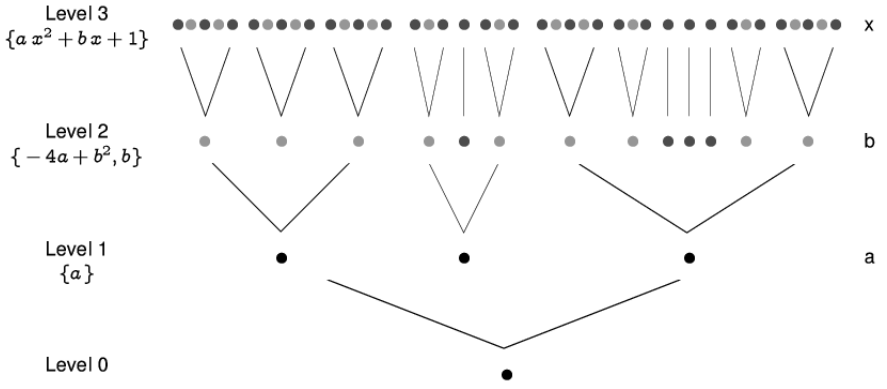

Fig. 2. Full CAD tree for $\varphi :\equiv \exists x(ax^2 + bx + 1 = 0)$. True cells are light gray, false cells are dark gray, and solid black cells bear no truth value.

cells on a higher level similarly, but extend the index of the corresponding base cell. So, e.g., the three children of C_1 in D_2 will be called C_{11}, C_{12}, C_{13} and the five children of C_{11} will be

called $C_{111}, \ldots, C_{115}$. There are 14 true cells in D_3. The first true cell is C_{112}. The last component of its sample point was generated as a first root. If δ_{11} is a describing formula for C_{11} then $(\delta_{11}, (x = \mathit{Root}(ax^2 + bx + 1, 1)))$ is the first guarded point. The last component of the sample point of C_{114} was generated as a second root. Hence $(\delta_{11}, (x = \mathit{Root}(ax^2 + bx + 1, 2)))$ is the next guarded point to collect. Proceeding with the algorithm we get the following twelve additional guarded points:

$$
\begin{gathered}
(\delta_{12}, (x = \mathit{Root}(ax^2 + bx + 1, 1))) \\
(\delta_{12}, (x = \mathit{Root}(ax^2 + bx + 1, 2))) \\
(\delta_{13}, (x = \mathit{Root}(ax^2 + bx + 1, 1))) \\
(\delta_{13}, (x = \mathit{Root}(ax^2 + bx + 1, 2))) \\
(\delta_{21}, (x = \mathit{Root}(ax^2 + bx + 1, 1))) \\
(\delta_{23}, (x = \mathit{Root}(ax^2 + bx + 1, 1))) \\
(\delta_{31}, (x = \mathit{Root}(ax^2 + bx + 1, 1))) \\
(\delta_{31}, (x = \mathit{Root}(ax^2 + bx + 1, 2))) \\
(\delta_{32}, (x = \mathit{Root}(ax^2 + bx + 1, 1))) \\
(\delta_{36}, (x = \mathit{Root}(ax^2 + bx + 1, 1))) \\
(\delta_{37}, (x = \mathit{Root}(ax^2 + bx + 1, 1))) \\
(\delta_{37}, (x = \mathit{Root}(ax^2 + bx + 1, 2)))
\end{gathered}
$$

We can reduce the number of points by combining them as follows: As $\delta_{21} \vee \delta_{23}$ is equivalent to $a = 0 \wedge b \neq 0$, we combine point 7 and 8 to

$$(a = 0 \wedge b \neq 0, (x = \mathit{Root}(ax^2 + bx + 1, 1))).$$

The latter could be further simplified by making use of the knowledge $a = 0$. This reduces the quadratic polynomial to a linear one.

Furthermore, as $\delta_{11} \vee \delta_{12} \vee \delta_{13} \vee \delta_{31} \vee \delta_{32} \vee \delta_{36} \vee \delta_{37}$ is equivalent to $4a - b^2 \leq 0 \wedge a \neq 0$, we can combine the points $1, 3, 5, 9, 11, 12, 13$ to

$$(4a - b^2 \leq 0 \wedge a \neq 0, (x = \mathit{Root}(ax^2 + bx + 1, 1))).$$

Finally, as $\delta_{11} \vee \delta_{12} \vee \delta_{13} \vee \delta_{31} \vee \delta_{37}$ is equivalent to $4a - b^2 < 0 \wedge a \neq 0$, we can combine the points $2, 4, 6, 10, 14$ to

$$(4a - b^2 < 0 \wedge a \neq 0, (x = \mathit{Root}(ax^2 + bx + 1, 2))).$$

We finish this example by concluding that we could reduce the 14 guarded points to three, which quite match the three cases found by the VS method. (Actually, the reason why we did not get exactly the same result are the cells C_{322} and C_{362}, where the last component of the sample point is a single root with multiplicity two.)

Third Example Revisited In Subsection 2.3 we found an exact implicit description for a given parametric curve, and ended up with the formula

$$\rho = 0 \text{ and } x \leq -63.$$

This hints that there might be extraneous points in the halfspace $x > -63$. So we apply extended QE by CAD to

$$\exists x \exists y (\rho = 0 \wedge x > 63)$$

and derive *true* and the answer

$$x = \frac{-30758091}{559682}, \quad y = \frac{42875}{529}.$$

We can now convince ourselves that this is the only extraneous point by applying QE to

$$\exists x \exists y (x > -63 \wedge \rho = 0 \wedge \neg(559682x = -30758091 \wedge 529y = 42875))$$

and deriving *false*.

5 Parametric Counter-Examples

In this section we assume the input formula

$$\varphi(x_1, \ldots, x_k) \equiv \forall x_{k+1} \cdots \forall x_l \mathsf{Q}_{l+1} x_{l+1} \cdots \mathsf{Q}_r x_r \psi$$

to have a leading block of universal quantifiers. This is dual to the assumption in Section 3. By negating this input formula, we get

$$\overline{\varphi}(x_1, \ldots, x_k) \equiv \exists x_{k+1} \cdots \exists x_l \overline{\mathsf{Q}_{l+1}} x_{l+1} \cdots \overline{\mathsf{Q}_r} x_r \overline{\psi}.$$

For formulas, overlining is just an alternative notation for negation. For quantifiers, an overlined quantifier denotes the dual one. Let

$$\{(\psi'_i, (x_{k+1} = b_{i,k+1}, \ldots, x_l = b_{i,l})) | i \in I\}$$

be the output of the extended algorithm on input of $\overline{\varphi}$. Then, dually to (C1) and (C2) the following two conditions hold.

($\overline{\text{C1}}$) *Quantifier-free formula.* We can easily construct from the set of guarded points a quantifier-free formula, which is equivalent to the input formula:

$$\mathbb{R} \models \varphi \leftrightarrow \bigwedge_{i \in I} \neg \psi'_i.$$

($\overline{\text{C2}}$) *Counter-example.* For every guarded point $(\psi, (x_{k+1} = b_{k+1}, \ldots, x_l = b_l))$ in the output set and every $(a_1, \ldots, a_k) \in \mathbb{R}^k$ with $\mathbb{R} \models \psi(a_1, \ldots, a_k)$, we have

$$\mathbb{R} \not\models \mathsf{Q}_{l+1} x_{l+1} \cdots \mathsf{Q}_r x_r \psi(a_1, \ldots, a_l),$$

where $\gamma(a_1, \ldots, a_k) = (a_{k+1}, \ldots, a_l)$.

In other words: Instead as a union of true cells we get the set described by a solution formula as intersection of complements of false cells. If given parameter values $(a_1, \ldots, a_k)$ lie in a false cell, then the appropriate guarded point delivers counter-examples $(a_{k+1}, \ldots, a_l)$ such that $\mathsf{Q}_{l+1} x_{l+1} \cdots \mathsf{Q}_r x_r \psi(a_1, \ldots, a_l)$ does not hold.

6 Implementation

The algorithms described in this paper are currently being implemented in the computer logic system REDLOG as part of the computer algebra system REDUCE. The new version 3.8 was recently announced. For details see the webpage `http://reduce-algebra.com/`.

7 Acknowledgment

The author has been supported by the DFG Project WE 566/5.

Conclusions

We have motivated that it is highly desirable to produce answers in addition to a solution formula as output of quantifier elimination for a formula with leading existential block of quantifiers. We have devised an algorithm to extend the cylindrical algebraic decomposition method to produce such answers. For non-decision problems, these sample solutions are in general parametric. For the dual case, i.e. for formulas with leading universal block of quantifiers, we get parametric counter-examples. The main benefit over a virtual substitution based approach is that there are no degree restrictions on the input.

References

1. Chaouki T. Abdallah, Peter Dorato, Wei Yang, Richard Liska, and Stanly Steinberg. Applications of quantifier elimination theory to control system design. In *Proceedings of the 4th IEEE Mediterranean Symposium on Control and Automation*, pages 340–345. IEEE, 1996.
2. Dennis S. Arnon, George E. Collins, and Scott McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal on Computing*, 13(4):865–877, November 1984.
3. George E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In H. Brakhage, editor, *Automata Theory and Formal Languages. 2nd GI Conference*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183. Springer-Verlag, Berlin, Heidelberg, New York, 1975.
4. Andreas Dolzmann. Reelle Quantorenelimination durch parametrisches Zählen von Nullstellen. Diploma thesis, Universität Passau, D-94030 Passau, Germany, November 1994.
5. Andreas Dolzmann, Andreas Seidl, and Thomas Sturm. *REDLOG User Manual*, April 2004. Edition 3.0.
6. Andreas Dolzmann and Thomas Sturm. Guarded expressions in practice. In Wolfgang W. Küchlin, editor, *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (ISSAC 97)*, pages 376–383, Maui, HI, July 1997. ACM, ACM Press, New York, 1997.
7. Andreas Dolzmann and Volker Weispfenning. Multiple object semilinear motion planning. To appear.
8. Lorenz A. Gilch. Effiziente Hermitesche Quantorenelimination. Diploma thesis, Universität Passau, D-94030 Passau, Germany, September 2003.
9. Hoon Hong, Richard Liska, and Stanly Steinberg. Testing stability by quantifier elimination. *Journal of Symbolic Computation*, 24(2):161–187, August 1997. Special issue on applications of quantifier elimination.
10. Nikolaos I. Ioakimidis. REDLOG-aided derivation of feasibility conditions in applied mechanics and engineering problems under simple inequality constraints. *Journal of Mechanical Engineering (Strojnícky Časopis)*, 50(1):58–69, 1999.
11. Mats Jirstrand. Nonlinear control system design by quantifier elimination. *Journal of Symbolic Computation*, 24(2):137–152, August 1997. Special issue on applications of quantifier elimination.
12. Andreas Seidl. Cylindrical algebraic decomposition for real quantifier elimination within redlog. Talk at the 8th International Conference on Applications of Computer Algebra, June 2002.
13. Werner M. Seiler and Andreas Weber. Deciding ellipticity by quantifier elimination. In V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing. Proceedings of the CASC 2003*, pages 345–356. Institut für Informatik, Technische Universität München, Passau, 2003.
14. Thomas Sturm. *Real Quantifier Elimination in Geometry*. Doctoral dissertation, Department of Mathematics and Computer Science. University of Passau, Germany, D-94030 Passau, Germany, December 1999.
15. Thomas Sturm. Reasoning over networks by symbolic methods. *Applicable Algebra in Engineering, Communication and Computing*, 10(1):79–96, September 1999.
16. Thomas Sturm. An algebraic approach to offsetting and blending of solids. In V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing. Proceedings of the CASC 2000*, pages 367–382. Springer, Berlin, 2000.
17. Volker Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1&2):3–27, February–April 1988.
18. Volker Weispfenning. Parametric linear and quadratic optimization by elimination. Technical Report MIP-9404, FMI, Universität Passau, D-94030 Passau, Germany, April 1994.
19. Volker Weispfenning. Semilinear motion planning among moving objects in REDLOG. In V. G. Ganzha and E. W. Mayr, editors, *Computer Algebra in Scientific Computing. Proceedings of the CASC 2001*, pages 541–553. Springer, Berlin, 2001.
20. Volker Weispfenning. Semilinear motion planning in REDLOG. *Applicable Algebra in Engineering, Communication and Computing*, 12:455–475, June 2001.