数式処理 *J.JSSAC (2003)*
Vol. 10, No. 1, pp. 13 - 22

特集 “Quantifier Elimination”

An Overview of QEPCAD B: a Tool for Real Quantifier Elimination and Formula Simplification

Christopher W. Brown*

United States Naval Academy

Abstract

This paper describes the basic functionality of Qepcad b, a system for computing with semi-algebraic sets via Cylindrical Algebraic Decomposition (CAD). Qepcad b is an interactive command-line based program, written in C, and built on top of the Saclib library. It extends and improves the Qepcad system. The article focuses on using Qepcad b to solve problems, describing the basic facilities offered by the system and providing examples of applications of these facilities. The program is freely available at `www.cs.usna.edu/~qepcad`.

1 Introduction

Qepcad b[1)] is a system for computing with semi-algebraic sets. A *semi-algebraic set* is a subset of $\mathbb{R}^n$ that can be defined as the set of points satisfying a boolean combination of polynomial equalities and inequalities in the variables $x_1, \ldots, x_n$. These defining formulas for semi-algebraic sets are often called *Tarski formulas.* Many familiar geometric objects are semi-algebraic sets — like discs (e.g. $x^2+y^2 < 1$), polygons (e.g. $a > 0 \wedge b > 0 \wedge a+b < 2$), ellipsoids (e.g. $2x^2 + 5y^2 + z^2 = 1$), semi-circles (e.g. $a^2 + b^2 = 1 \wedge b > 0$), etc. — as are many important mathematical objects that we might not tend to think of geometrically — like roots of polynomials, the set of parameter values for which solutions to a parameterized polynomial system exist, minimal values of rational functions in polynomially constrained regions, etc. The user describes semi-algebraic sets to Qepcad b by defining formulas, and the system is able to then do a variety of things. The two most basic are:

- *Formula Simplification* — Qepcad b produces a simplified formula defining the same set as defined by the input formula. For example, given input “$ab \leq 0 \wedge a + b =$

*wcbrown@usna.edu

1) Qepcad b extends and improves the Qepcad system [4], due primarily to Hoon Hong but with contributions by many others. The author thanks Hoon Hong for permission to branch Qepcad b off from Qepcad, and thanks all who contributed to Qepcad.

© 2003 Japan Society for Symbolic and Algebraic Computation

$0 \wedge b^2 + a^2 > 0 \vee a^2 = -b^2$" Qepcad b produces output "$b + a = 0$", which is equivalent over the reals.

- *Quantifier Elimination* — Qepcad b allows variables from the input formula to be quantified, in which case it returns a formula defining the same set as that defined by the quantified input formula, but which contains no quantifiers, and which only contains the variables that appeared unquantified in the input. For example, given input "$\forall x[x^2 + ax + b = 0 \Longrightarrow x > 1]$ assuming $ab < -2$" Qepcad b produces output "$a \leq 0 \wedge b + a + 1 > 0$". Although it might not be obvious, the output is equivalent over the reals to the input given the assumption $ab < -2$. The fact that for every Tarski formula with quantifiers there is an equivalent Tarski formula without quantifiers was proved by Tarski in the 1930's when he gave an algorithm for performing quantifier elimination.

The rest of this paper describes Qepcad b's basic functionality of simplification and quantifier elimination for Tarski formulas with two detailed examples, gives an overview of some extensions of this basic functionality, and provides a brief comparison with other current systems.

2 Simplification

Given an unquantified Tarski formula as input, Qepcad b returns a (hopefully!) simpler formula that is equivalent over the reals. In a very simple case, we might give Qepcad b the formula $x \leq 0 \wedge x^2 > 0$ and receive as output $x < 0$. The two formulas are equivalent over $\mathbb{R}$, and the output is clearly simpler. This section provides a specific example of how simplification helps solve problems, and ends with a discussion of why simplification is important in general.

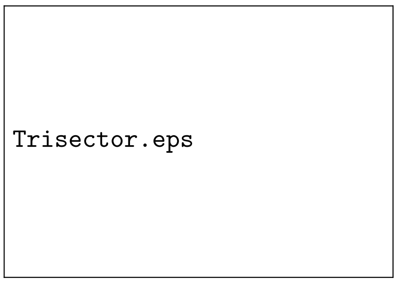

Figure 1: The external trisector of B with respect to A.

Consider the triangle ABC from Figure 1. We define *the external trisector of B with respect to A* as the segment connecting vertex B with the intersection of the external trisector of B and the ray from A through C. Of course the external trisector of B with respect to A does not always exist, and the problem we consider is to characterize in terms of the side lengths a, b and c the triangles for which it does exist.[2)] We can do this quite nicely with formula simplification.

It is clear from the picture that the external trisector of B with respect to A exists if and only if $(\pi - \phi)/3 < \theta$. However, we'd like a characterization in terms of side lengths not angles. We derive an equivalent statement to $(\pi - \phi)/3 < \theta$ in terms of a, b and c using little more than the law of cosines:

Case 1: Assuming $\theta \leq \pi/3$ we derive the following:

$$\begin{array}{rcl} \pi - \phi & < & 3\theta, \text{ note that both sides are in } [0, \pi] \\ -\cos(\pi - \phi) & < & -\cos(3\theta) \\ \cos\phi & < & -\cos(3\theta) \\ \cos\phi & < & -4\cos^3\theta + 3\cos\theta \\ \frac{a^2+c^2-b^2}{2ac} & < & -4\left(\frac{a^2+b^2-c^2}{2ab}\right)^3 + 3\left(\frac{a^2+b^2-c^2}{2ab}\right) \\ a^2b^3\left(a^2+c^2-b^2\right) & < & -c\left(a^2+b^2-c^2\right)^3 + 3a^2b^2c\left(a^2+b^2-c^2\right) \end{array}$$

Case 2: Assuming $\theta > \pi/3$ we see immediately that $\pi - \phi < 3\theta$ holds.

We are in Case 1 exactly when

$$\begin{array}{rcl} \theta & \leq & \pi/3, \text{ note that both sides are in } [0, \pi] \\ \cos\theta & \geq & \cos\frac{\pi}{3} \\ \frac{a^2+b^2-c^2}{2ab} & \geq & \frac{1}{2} \\ a^2+b^2-c^2 & \geq & ab, \end{array}$$

and so are in Case 2 when $a^2 + b^2 - c^2 < ab$. Putting it all together, the external trisector of B with respect to A exists if and only if the side lengths a, b, c satisfy

$$\underbrace{a^2+b^2-c^2 \geq ab}_{\text{Case 1}} \wedge \underbrace{a^2b^3\left(a^2+c^2-b^2\right) < -c\left(a^2+b^2-c^2\right)^3 + 3a^2b^2c\left(a^2+b^2-c^2\right)}_{\text{Derived condition for Case 1}} \vee \underbrace{a^2+b^2-c^2 < ab}_{\text{Case 2}}$$

This formula is a characterization of the existence of the external trisector in terms of the side lengths, but not a very nice characterization. Figure 2 shows a Qepcad b session simplifying it. Notice that we must explicitly declare the assumption that a, b and c are actually side lengths of a non-degenerate triangle, i.e. that all three are positive and satisfy the triangle inequalities. These assumptions are implicit in the law of cosines. The resulting characterization, $c^2 + bc - a^2 > 0$, is quite a bit simpler, gives more insight, and provides better input for further computation.

The preceding example demonstrates simplification and gives some insight as to how simplification problems arise. However, it doesn't demonstrate one of the most important

[2)] This question arose from work with George Nakos.

Figure 2: Qepcad b simplification session.

sources of simplification problems: other algorithms! Other methods for computing with semi-algebraic sets systematically produce large redundant formulas. The Redlog system [5], for example, employs the method of quantifier elimination by virtual substitution [10], which is essentially a rewrite rule. Each time this rule is applied to a formula, the size of the formula grows. Thus, a very large formula is the expected result, even if the set defined by that formula is expected to be quite simple. The restricted quantifier elimination algorithm described in [6], has the same property. Instead of using a rewrite rule, it enumerates for various sets of polynomials all sign-sequences having certain properties. Many of these sign-sequences may not be realizable by the given set of polynomials, or it may be that many sign-sequences can be merged to form an equivalent sign-sequence condition on a smaller set of polynomials. In any event, large output is produced systematically. Other algorithms have the same behaviour. Simplification is an important postprocessing step when such methods are employed.[3]

[3] The program Slfq (`www.cs.usna.edu/~qepcad/SLFQ`) employs Qepcad b as a black box in a divide-and-conquer approach to simplifying *very* large formulas ([1] describes the process and gives an example). For formulas in two variables, the system has proved quite successful. It is able to reduce formulas containing many thousands of inequalities to equivalents that can be written down in a line or two. With more variables, time and space requirements are often prohibitive.

3 Quantifier Elimination

A Tarski formula with quantified variables is, in a sense, a question. For example, $\exists x[x^2+bx+c=0]$ is the question "when does x^2+bx+c have a real root?" The well-known answer "when $b^2-4c \geq 0$" is an equivalent formula from which the quantified variable has been eliminated. Thus, quantifier elimination algorithms can be seen as providing "answers" to "questions" about semi-algebraic sets.

Many problems from a wide variety of fields in mathematics, science and engineering can be phrased as quantifier elimination problems. However, current quantifier elimination algorithms are not able to deal with large problems in general. In fact, the problem is fundamentally doubly exponential in the number of variables in the worst case, so this limitation is not surprising. None the less, many problems of "moderate" size can be solved within reasonable space and time constraints. The example problem considered here is sort of a "toy" problem, but it is a nice demonstration of solving problems with quantifier elimination. The following is from the last Putnam Exam:

> Let C_1 and C_2 be circles whose centers are 10 units apart and whose radii are 1 and 3. Find, with proof, the locus of all points M for which there exist points X on C_1 and Y on C_2 such that M is the midpoint of the line segment $\overline{XY}$.

Letting $X=(x_1,y_1)$ and $Y=(x_2,y_2)$, the locus to be computed can be defined as the points $M=(x,y)$ satisfying the following quantified formula:

$$\exists x_1 \exists y_1 \exists x_2 \exists y_2 [\ \overbrace{x_1^2+y_1^2=1}^{X \text{ on } C_1} \ \wedge\ \underbrace{(x_2-10)^2+y_2^2=9}_{Y \text{ on } C_2} \ \wedge\ \overbrace{2x=x_1+x_2 \wedge 2y=y_1+y_2}^{M \text{ is the midpoint of } \overline{XY}}\ ]$$

Thus, this problem can be solved by quantifier elimination. QEPCAD B is particularly sensitive to the number of variables in a problem, and this 6-variable phrasing of the problem cannot be solved by it in a reasonable amount of time and space. However, since $x_2=2x-x_1$ and $y_2=2y-y_1$, we can eliminate x_2 and y_2 by substitution, arriving at the 4-variable quantifier elimination problem

$$\exists x_1 \exists y_1 [\ x_1^2+y_1^2=1 \ \wedge\ (2x-x_1-10)^2+(2y-y_1)^2=9]$$

which QEPCAD B solves quite easily, as shown in Figure 3. The solution is seen to be the region outside of one circle and inside another. Putting the inequalities in the general form for circles yields $y^2+(x-5)^2 \geq 1^2 \wedge y^2+(x-5)^2 \leq 2^2$, from which we see that the solution is in fact an annulus of inner radius 1 and outer radius 2, centered at $(5,0)$.

Many problems more serious than Putnam Exam questions can be phrased as quantifier elimination problems, of course[4]. However, this example demonstrates two key points.

[4] The August 1997 issue of the Journal of Symbolic Computation (vol. 24, no. 2), a "Special Issue on Applications of Quantifier Elimination", has several examples.

pqesession.eps

Figure 3: QEPCAD B quantifier elimination session.

First of all, that quantifier elimination problems naturally arise when one asks questions about objects defined by polynomials. Secondly, QEPCAD B is not intended to be a true black box tool. One has to understand a little bit about how it works to use it effectively. It does not try to preprocess an input formula to remove variables by, for example, the obvious substitution we made for the Putnam problem. Given a formula like $\exists y[F(x,y) \vee G(x,y)]$, QEPCAD B does not return the "or" of the two smaller quantifier elimination problems $\exists y[F(x,y)]$ and $\exists y[G(x,y)]$, even though this is almost always the most efficient way to proceed. Instead, QEPCAD B always constructs a Cylindrical Algebraic Decomposition directly from its input formula, and uses this to perform quantifier elimination. It is thus up to the user to phrase his problem as an input formula (or a combination of input formulas) that is best suited for QEPCAD B. A preprocessor that would do such a thing automatically would be a good tool.

4 Cylindrical Algebraic Decomposition

To use QEPCAD B effectively, one must understand a bit about the theory of Cylindrical Algebraic Decomposition[5)] on which it is based. A *Cylindrical Algebraic Decomposition* (CAD) is essentially a data-structure for explicitly representing semi-algebraic sets. A CAD contains a great deal of information about the set it represents. One can, for example, immediately read off the dimension of the set, whether it is empty or the entire space, the number of points in the set (if it is finite), and many other important things. Moreover projection, which is the essence of quantifier elimination, is trivial in the CAD

[5)] CAD was invented by George Collins [3] in the early 1970's as the basis for his quantifier elimination algorithm. The outline of quantifier elimination by CAD given here is different than his original algorithm, following the algorithm given in [4] instead.

representation, and there is a straightforward and fast algorithm for computing a minimal representation of a set as a CAD, in decided contrast to the Tarski formula representation!

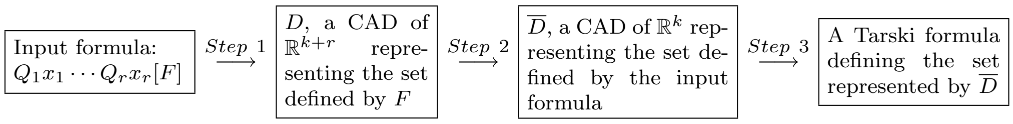
Input formula: $Q_1x_1 \cdots Q_rx_r[F]$
Step 1
D, a CAD of $\mathbb{R}^{k+r}$ representing the set defined by F
Step 2
$\overline{D}$, a CAD of $\mathbb{R}^k$ representing the set defined by the input formula
Step 3
A Tarski formula defining the set represented by $\overline{D}$

Figure 4: The process of quantifier elimination by CAD on input $Q_1x_1 \cdots Q_rx_r[F]$, where the Q_i's are quantifiers, and F is a Tarski formula in the variables $s_1, \ldots, s_k, x_1, \ldots, x_r$.

Conceptually, quantifier elimination by CAD is the three step process shown in Figure 4. Step 1 is the most time-consuming, being doubly exponential in $k + r$, the number of variables. Therefore, phrasing problems with as few variables as possible is crucial for using Qepcad b successfully. Step 1 is polynomial in other parameters measuring input size — coefficient size, degrees of polynomials, number of polynomials — but in practice it is important to keep these down as well. Step 2 is computationally trivial, which is one of the key properties of CAD. Step 3 is, in a sense, the most important, because the ability to produce *simple* Tarski formulas is unique to CAD-based methods. [7] reduces Step 3 to the well-known combinatorial problem of boolean formula minimization when the CAD $\overline{D}$ has the special property of *projection definability.* When $\overline{D}$ is not projection definable, Qepcad b adds an extra step before Step 3 to make $\overline{D}$ projection definable. The extra time and space required for this varies widely in practice, and it is impossible to predict in advance whether the extra step will even be needed.

5 Extensions

Qepcad b includes several extensions to the basic functionality of formula simplification and quantifier elimination that allow the user to use the CAD data structure more efficiently than is possible by using it simply as a black box for these two basic operations. Two extensions are described in this section. The first addresses the sensitivity of Step 1 to the number of variables, and the second addresses the issue of projection definability in Step 3.

In addition to "$\exists$" and "$\forall$", Qepcad b can eliminate several other "quantifiers" just as efficiently (or even more efficiently). One example is the "there exists exactly k" quantifier, which we will denote $\exists_k$. When this quantifier is applicable, problems can be phrased in fewer variables, which can dramatically affect the cost of Step 1. For example, to determine whether a formula $F(x, y)$ defines y as a function of x in the interval $-1 < x < 1$ using standard quantifier elimination, we would write $\exists y \forall y'[F(x, y) \wedge [F(x, y') \implies y = y']]$, assuming $-1 < x < 1$, which requires three variables. Using the "$\exists_k$" quantifier, this can be written in only two variables as $\exists_1 y[F(x,y)]$, assuming $-1 < x < 1$. As written, neither of these formulas return a true/false answer to the original question — rather they return a formula in x. However, if the quantified formula is satisfied for every x satisfying the stated assumption, QEPCAD B is guaranteed to return TRUE as its output formula. So the answer to the original question is true if QEPCAD B returns TRUE, and FALSE otherwise. QEPCAD B handles several other "quantifiers", including "there exist infinitely many" and "for all but finitely many". Both of these also allow a problem to be expressed in fewer variables than is possible with only $\exists$ and $\forall$, but they are also interesting because QEPCAD B can often eliminate theses quantifiers more efficiently than it can $\exists$ and $\forall$. Thus, when problems can be phrased with these quantifiers it is usually worthwhile to do so.

QEPCAD B is able to both read input and write output in an extended language that adds to the usual language of Tarski formulas the ability to refer to a root of a polynomial by its *index*. This allows some sets to be defined with fewer polynomials, which is important for the efficiency of Step 1. More importantly, all CADs are "projection definable" in this extended language, so that the extra step before Step 3 is never needed. This is especially important for combining results of different quantifier elimination problems. The extended language adds a new type of atomic formula, which has the form $x_i \ \sigma \ root_k f(x_1,\ldots,x_i)$ where $\sigma \in \{=,\neq,<,>,\leq,\geq\}$ and k is a nonzero integer. This formula is satisfied at $(\alpha_1,\ldots,\alpha_i)$ if $f(\alpha_i,\ldots,\alpha_{i-1},x_i)$ has at least $|k|$ roots and if α_i has relation σ with the $|k|$th of them, ordered least to greatest if k is positive and greatest to least if k is negative. If, for example, QEPCAD B is given input $\exists y[x^2+y^2=1 \wedge x+y<0]$ it produces output $x+1 \geq 0 \wedge x < root_{-1} 2x^2-1$ in the extended language, while in the language of Tarski formulas it gives $x+1 \geq 0 \wedge [2x^2-1 < 0 \vee x \leq 0]$. This second computation requires the extra step to make $\overline{D}$ projection definable, and an extra polynomial is needed in the formula.

6 Other Software

The Redlog system, which is distributed with Reduce, also provides facilities for quantifier elimination and formula simplification. Its quantifier elimination is based on virtual term substitution and its simplification is based on a variety of strategies, but CAD is not a part of either. To the best of the author's knowledge there is no other system that offers both facilities for any significant subset of the Tarski formulas. Redlog has restrictions on the degree of the polynomials from the input formula in the quantified variables. For formulas that meet these degree restrictions it is often significantly faster than QEPCAD B, particularly when there are many free variables. Redlog's simplification is much faster than QEPCAD B's, but much less successful at actually simplifying. It is primarily aimed at quickly simplifying formulas arising as intermediate results during Redlog's quantifier elimination computations. Redlog and QEPCAD B actually complement one another quite well. For example, results of quantifier elimination by Redlog can be simplified by QEPCAD B to provide results that neither program could produce on its own.

There are several implementations of CAD and of quantifier elimination by CAD. Mathematica has an implementation [9], an implementation is being developed as part of Redlog [8], and there is, of course, the QEPCAD system [4] on which QEPCAD B is based. None of these is able to do formula simplification. Mathematica is the only one of the three to provide something similar to QEPCAD B's extended language, however, it does not provide Tarski formula equivalents to formulas in this extended language. QEPCAD B's extended quantifiers are, to the best of the author's knowledge, unique to QEPCAD B.

References

[1] C. W. Brown. Simple CAD construction and its applications. *Journal of Symbolic Computation*, 31(5):521–547, May 2001.

[2] B.F. Caviness and J. R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition.* Texts and Monographs in Symbolic Computation. Springer-Verlag, 1998.

[3] G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *Lecture Notes In Computer Science*, volume Vol. 33, pages 134–183. Springer-Verlag, Berlin, 1975. Reprinted in [2].

[4] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, Sep 1991.

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

[6] L. Gonzalez-Vega. A combinatorial algorithm solving some quantifier elimination problems. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Symbolic Computation. Springer-Verlag, Vienna, 1998.

[7] H. Hong. Simple solution formula construction in cylindrical algebraic decomposition based quantifier elimination. In *Proc. International Symposium on Symbolic and Algebraic Computation*, pages 177–188, 1992.

[8] A. Seidl and T. Sturm. A generic projection operator for partial cylindrical algebraic decomposition. Technical Report MIP-0301, Universität Passau, January 2003.

[9] A. Strzebonski. Solving systems of strict polynomial inequalities. *Journal of Symbolic Computation*, 29:471–480, 2000.

[10] V. Weispfenning. Quantifier elimination for real algebra — the quadratic case and beyond. *AAECC*, 8:85–101, 1997.