Real Quantifier Elimination in the RegularChains Library

Changbo Chen[1] and Marc Moreno Maza[2]

[1] Chongqing Key Laboratory of Automated Reasoning and Cognition, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, China
changbo.chen@hotmail.com,
http://www.orcca.on.ca/∼cchen
[2] ORCCA, University of Western Ontario, Canada
moreno@csd.uwo.ca,
www.csd.uwo.ca/∼moreno

Abstract. Quantifier elimination (QE) over real closed fields has found numerous applications. Cylindrical algebraic decomposition (CAD) is one of the main tools for handling quantifier elimination of nonlinear input formulas. Despite of its worst case doubly exponential complexity, CAD-based quantifier elimination remains interesting for handling general quantified formulas and producing simple quantifier-free formulas. In this paper, we report on the implementation of a QE procedure, called QuantifierElimination, based on the CAD implementations in the RegularChains library. This command supports both standard quantifier-free formula and extended Tarski formula in the output. The use of the QE procedure is illustrated by solving examples from different applications.

Keywords: Quantifier elimination, cylindrical algebraic decomposition, triangular decomposition, RegularChains

1 Introduction

In the 1930's, A. Tarski [11] proved that quantifier elimination over the reals is possible and provided the first algorithm for real quantifier elimination, although the complexity of his algorithm is not even elementary recursive. In 1975, G. E. Collins [7] invented cylindrical algebraic decomposition, which opens the door for solving quantifier elimination practically. The worst case complexity for solving QE by means of CAD is doubly exponential in the number of variables. In the 1990's, QE algorithms, whose worst complexity are doubly exponential in the number of alternative quantifier blocks instead of variables, emerged [1]. Although QE based on CAD is not favorable in terms of complexity, it remains a practical tool for solving general QE problems and obtaining simple quantifier free formula.

Many authors have improved the practical efficiency of CAD based on the original projection-lifting scheme proposed by Collins. In [6], with B. Xia and L. Yang, we introduced an alternative way of computing CADs based on triangular decompositions. In this new method, one first computes a complex cylindrical decomposition (CCD), which partitions the complex space into cylindrically arranged cells, each of which is the complex zero set of a regular system. In a second stage, the real connected components of each regular system are computed, which all together form a CAD of the real space. A CAD computed in this way is called an RC-CAD. The efficiency of RC-CAD was substantially improved in [5], where an incremental algorithm was proposed to compute CCDs. Moreover, in the same paper, a systematic way for making use of equational constraints is presented.

In [4], an RC-CAD based quantifier elimination algorithm was proposed. A preliminary implementation of it in the `RegularChains` library is available through the function QuantifierElimination. The goal of this paper is to present the implementation details of such an algorithm. Several important optimizations are also discussed. The paper is organized as follows. In Section 2, we illustrate the user interface of QuantifierElimination by some simple examples. In Section 3, we present some non-trivial applications of it. In Section 4, we explain the underlying theory and algorithm as well as some optimizations realized in the implementation.

2 Functionality

In this section, we explain the usage of QuantifierElimination by some simple examples.

In Figure 1, the user interface of QuantifierElimination is illustrated by the famous Davenport-Heintz example.

Solve the Davenport–Heintz problem by QuantifierElimination.

(∃ c, ∀ b, a) ((a = d ∧ b = c) ∨ (a=c ∧ b = 1)) ⇒a^2 = b.

```
> f := &E([c]), &A([b, a]), ((a=d) &and (b=c)) &or
                 ((a=c) &and (b=1)) &implies (a^2=b);
```

$f := \&E([c]), \&A([b, a]), a = d \,\&\text{and}\, b = c \,\&\text{or}\, a = c \,\&\text{and}\, b = 1 \,\&\text{implies}\, a^2 = b$

```
> out := QuantifierElimination(f);
```

$out := d - 1 = 0 \,\&\text{or}\, d + 1 = 0$

Fig. 1. The user interface of QuantifierElimination.

The user interface of QuantifierElimination is implemented on top of the Logic package of Maple. This package supports usual logical operators, such as ∧,

$\vee$, $\neg$, $\implies$, $\iff$, and represent them respectively by &and, &or, &not, &implies, &iff. There is also a function called Normalize, which can convert a given logical formula into its disjunctive normal form or conjunctive normal form. However, the quantifiers are missing in the Logic package. We create the symbol $\&E$ and $\&A$ to represent respectively the existential quantifier $\exists$ and the universal quantifier $\forall$. To use them, the quantified variables have to been put in a list, as shown in Figure 1. Note that all operators in the Logic package have the same precedence. Parentheses should be used to correctly specify the precedence.

In Figure 1, the order of variables is not specified. In such case, QuantifierElimination calls the function SuggestVariableOrder of `RegularChains` library to pick a "good" order by some heuristic strategy. It is also possible for the user to choose her favorable order, as shown in Figure 2, where the variables supplied to the function PolynomialRing are in descending order.

```
> R := PolynomialRing([x, a, b, c]);
  f := &E([x]), a*x^2+b*x+c=0;
  out := QuantifierElimination(f, R);
```

$$R := polynomial\_ring$$

$$f := \&E([x]), x^2\, a + x\, b + c = 0$$

$$out := ((4\, a\, c - b^2 < 0 \;\&or\; 4\, a\, c - b^2 = 0 \;\&and\; a < 0) \;\&or\; 4\, a\, c - b^2 = 0 \;\&and\; 0 < a) \;\&or\; (4\, a\, c - b^2 = 0 \;\&and\; a = 0) \;\&and\; c = 0$$

Fig. 2. The default output of QuantifierElimination is quantifier free formula.

The default output of QuantifierElimination is a quantifier free formula formed by polynomial constraints and logical connectives, which is the same as the default output of QEPCAD. Such formulas are called Tarski formulas. An alternative output format, called extended Tarski formula, is also available, when the option 'output'='rootof' is specified. An extended Tarski formula extends Tarski formula by allowing indexed roots of polynomials to appear. This is illustrated by Figure 3 and Figure 4. Such an output format is the same as the default output of Mathematica.

The users who are familiar with MAPLE's RootOf may be surprised to see the real index there. Indeed, it is a new feature we added to Rootof, which is currently supported by the evalf function of MAPLE as shown in Figure 4.

3 Application

In this section, we present how QuantifierElimination is applied to solve several applications.

```
> f := &E([x]), a*x^2+b*x+c=0;
  out := QuantifierElimination(f,'output'='rootof');
```

$$f := \&E([x]),\ a\,x^2 + b\,x + c = 0$$

$$out := \left(\left(\left(\left(a < 0 \ \&\text{and}\ \frac{1}{4}\,\frac{b^2}{a} \le c \ \&\text{or}\ a = 0 \ \&\text{and}\ b < 0\right) \&\text{or}\ (a = 0 \ \&\text{and}\ b = 0) \ \&\text{and}\ c = 0\right) \&\text{or}\ a = 0 \ \&\text{and}\ 0 < b\right) \&\text{or}\ 0 < a \ \&\text{and}\ c \le \frac{1}{4}\,\frac{b^2}{a}\right)$$

Fig. 3. The output of QuantifierElimination in extended Tarski formula.

The first application is on the verification and synthesis of switched and hybrid dynamical systems [10]. A common problem studied in this field is to determine if a system remains in the safe state if it starts in an initial safe state. A typical approach to solve this problem is to find a certificate, or an invariant set, such that the following are satisfied simultaneously:

- the initial states satisfy the invariant set
- any states that satisfy the invariant set are safe
- the system dynamics cannot force the system to leave the invariant set

Finding such a certificate can be casted into a real quantifier elimination problem.

In Figure 5, we show how to use QuantifierElimination to solve the quantifier elimination problem casted from an 1-D robot model [10], where the details of the casting are explained. This problem was originally solved in [10] by a combination of Reduce and QEPCAD.

The second application is on computing control Lyapunov function. Suppose we are given a dynamical system $\dot{x} = f(x, u)$, where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}$ are respectively the state variables and the control input implicitly depending on t.

Definition 1 *A function $V(x) : \mathbb{R}^n \to \mathbb{R}$ is called a* control Lyapunov function *of the dynamical system $\dot{x} = f(x, u)$ if the following are satisfied:*

- *$V(x)$ is positive definite, that is $V(0) = 0$ and $\forall x \neq 0$, $V(x) > 0$.*
- *$\dot{V}(0) = 0$ and $\forall x \neq 0$, $\exists u$, such that $\dot{V} < 0$, where $\dot{V} = \nabla V(x) \cdot f(x, u)$.*
- *V is radically unbounded, that is $\|x\| \to \infty$ implies that $V \to \infty$.*

Suppose one wants to know if there exists a control Lyapunov function of a given template $V(a, x)$, where a are parameters. The equivalent QE problem is:

$$(\forall x, \exists u)(x \neq 0) \implies (V > 0 \wedge \nabla V(x) \cdot f(x, u) < 0).$$

If one also wants to find out if there exists u of a given template $g(b, x)$, then the equivalent QE problem is:

$$(\forall x, \exists u)\left((u = g(b, x)) \wedge (x \neq 0 \implies (V > 0 \wedge \nabla V(x) \cdot f(x, u) < 0))\right).$$

```
> f := &E([y]), y^2+x^2=2;
  out := QuantifierElimination(f, output=rootof);
                    f := &E([y]), x^2 + y^2 = 2
              out := -sqrt(2) <= x &and x <= sqrt(2)

> f := &E([y]), y^4+x^4=2;
  out := QuantifierElimination(f, output=rootof);
                    f := &E([y]), x^4 + y^4 = 2
out := RootOf(_Z^4 - 2, index = real_1) <= x &and x <= RootOf(_Z^4 - 2, index
    = real_2)

> evalf(op(1, out)); evalf(op(2, out));
                    -1.189207115 <= x
                    x <= 1.189207115
```

Fig. 4. Solve QuantifierElimination in extended Tarski formula.

Let's illustrate this application by a bivariate dynamical system.

$$\begin{cases} \frac{dx_1}{dt} = -x_1 + u \\ \frac{dx_2}{dt} = -x_1 - x_2^3 \end{cases}$$

We aim to find control Lyapunov function of the form $V := a_1x_1^2 + a_2x_2^2$ and control input of the form $u := b_1x_1 + b_2x_2$. Figure 6 shows how to call QuantifierElimination to find parameters a_1, a_2, b_1, b_2 such that control Lyapunov function exists. The computation takes several seconds. To verify the result, let $a_1 = a_2 = b_2 = 1$ and $b_1 = 0$, we obtain $u = x_2$, $V = x_1^2 + x_2^2$ and $\dot{V} = -2x_1^2 - 4x_2^2$. Clearly V is a control Lyapunov function.

4 Underlying theory

Let $PF := (Q_{k+1}x_{k+1}, \ldots, Q_nx_n)FF(x_1, \ldots, x_n)$, where FF is a logical formula formed by polynomial constraints with real number coefficients and logical connectives and each Q_i, $k+1 \le i \le n$, is an existential or universal quantifier. The problem of quantifier elimination looks for an equivalent quantifier free formula SF involving only the free variables $x_1, \cdots, x_k$. Let F be the set of polynomials appearing in FF.

The QE algorithm based on RC-CAD consists of the following steps:

1. Compute an F-sign invariant CCD of $\mathbb{C}^n$, that is a CCD such that above any given cell of it, each polynomial in F either vanishes at all points of the cell or no points of the cell.

```
> phi1 := ( ( 74 <= x ) &and ( x <= 76 ) &and ( v = 0 )
  &implies ( -v^2 - a * (x-75)^2 + b >= 0 ) ):

> phi2 := ( ( -v^2 - a * (x-75)^2 + b >= 0 )
  &implies (( 80 >= x ) &and ( x >= 70 )) ):

> phi3 := ( ( -v^2 - a * (x-75)^2 + b = 0 )
  &implies (( -2*v - a * 2 * (x-75)* v >= 0 ) &or ( 2*v - a
  * 2 * (x-75)* v >= 0 )) ):

> phi := phi1 &and phi2 &and phi3:
> t0 := time():
  psi := QuantifierElimination(&A([x,v]),phi,output=rootof);
  t1 := time() - t0;
```

$$\psi := ((0 < a \;\&\text{and}\; a \le 1) \;\&\text{and}\; a \le b) \;\&\text{and}\; b \le \min\left(\frac{1}{a}, 25\, a\right)$$

$$t1 := 15.094$$

Fig. 5. Solve a QE problem related to 1-D robot model

2. Produce an F-invariant CAD of $\mathbb{R}^n$ from the CCD by real root isolation.
3. For each cell c of the CAD, evaluate FF at a sample point of c and attach the resulting truth value to c.
4. Propagate the truth value according to the quantifiers until each cell in the the induced CAD of $\mathbb{R}^k$ is attached with a truth value, see Figure 7.
5. Each true cell has a defining extended Tarski formula representation. If only extended Tarski formula output is required, then the disjunction of the representation of all true cells, with possible simplification, gives the solution formula SF. If Tarski formula is required, one tests if the signs of polynomials in the CCD are enough to distinguish true and false cells of the CAD. If yes, a representation of the true cells by the signs of these polynomials gives SF. If no, the CCD is refined and the algorithm resumes from Step 2.

It was proved in [4] that the above process terminates in finitely many steps.

We explain now briefly a few optimizations that have been implemented in QuantifierElimination. Let $PF := (Q_{k+1}x_{k+1}, \ldots, Q_n x_n) FF(x_1, \ldots, x_n)$. If FF is a conjunctive formula having equational constraints, then truth-invariant CCDs and CADs are computed instead of sign-invariant ones using techniques proposed in [5]. If FF is in disjunctive normal form and has equational constraints, then truth table invariant CCDs and CADs are computed using algorithm presented in [2]. If there exists m, $k+1 \le m \le n$, such that $Q_m = \cdots = Q_n = \forall$, then PF is converted to its equivalent form

$$(Q_{k+1}x_{k+1}, \ldots, Q_{m-1}x_{m-1}, \neg\exists x_m, \exists x_{m+1}, \ldots, \exists x_n) \neg FF(x_1, \ldots, x_n).$$

```
> f1 := -x_1+u: f2 := -x_1-x_2^3:
  V := a_1*x_1^2+a_2*x_2^2;
  Vt := diff(V, x_1)*f1 + diff(V, x_2)*f2;
```

$$V := x\_2^2\ a\_2 + x\_1^2\ a\_1$$

$$Vt := 2\ a\_1\ x\_1\ (u - x\_1) + 2\ a\_2\ x\_2\left(-x\_2^3 - x\_1\right)$$

```
> QuantifierElimination( &A([x_1,x_2]), &E([u]), (x_1<>0)
  &or (x_2<>0) &implies ((V>0) &and (Vt<0)) );
```

$$0 < a\_1 \ \&\text{and}\ 0 < a\_2$$

```
> QuantifierElimination( &A([x_1, x_2]), &E([u]), (u=b_1*
  x_1+b_2*x_2) &and (a_1>0) &and (a_2>0) &and ((x_1<>0) &or
  (x_2<>0) &implies ((Vt<0))) );
```

$$((b\_2\ a\_1 - a\_2 = 0 \ \&\text{and}\ 0 < a\_1) \ \&\text{and}\ 0 < a\_2) \ \&\text{and}\ b\_1 < 1$$

Fig. 6. Compute control Layapunov function.

This trick is particular useful if FF is of the form $A \implies B$, where A has equational constraints, as $\neg FF$ is equivalent to $A \wedge \neg B$, which can benefit from the techniques for making use of equational constraints in [5, 2].

We have also implemented some simple partial lifting techniques when FF is a conjunctive formula. Exploiting systematically the partial lifting techniques as in [8] is working in progress. In [4], some simplification strategies for the Tarski formula output of QuantifierElimination was proposed. The simplification remains to be enhanced by integrating techniques as in [9, 3]. For the extended Tarski formula, a better technique for merging true cells is working in progress.

Acknowledgments

This work was supported by the NSFC (11301524) and the CSTC (cstc2013jjys0002).

References

1. S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computations in Mathematics*. Springer-Verlag, 2006.
2. R. Bradford, C. Chen, J.H. Davenport, M. England, M. Moreno Maza, and D. Wilson. Truth table invariant cylindrical algebraic decomposition by regular chains. Submitted. Preprint: `http://opus.bath.ac.uk/38344/`, 2014.
3. C. W. Brown. *Solution Formula Construction for Truth Invariant CAD's*. PhD thesis, University of Delaware, 1999.
4. C. Chen and M. Moreno Maza. Quantifier elimination by cylindrical algebraic decomposition based on regular chains. (Accepted for ISSAC 2014), 2014.

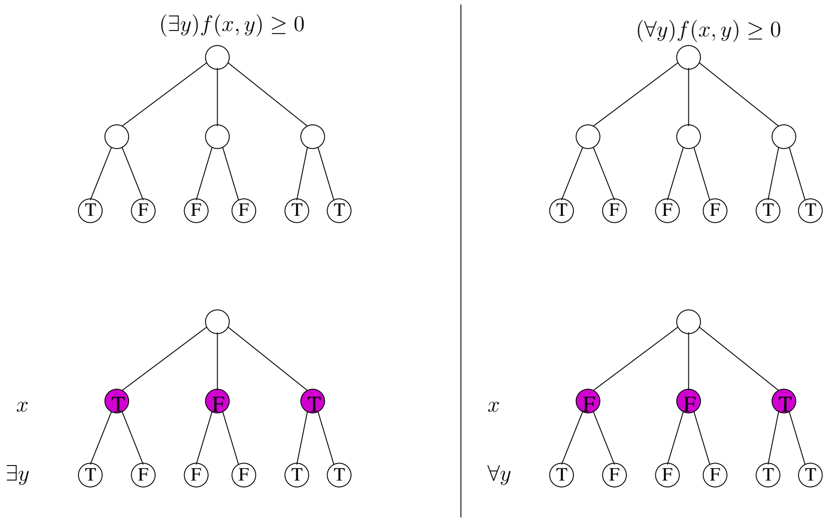

Fig. 7. Propagate truth values.

5. C. Chen and M. Moreno Maza. An incremental algorithm for computing cylindrical algebraic decompositions. *Proc. ASCM '12*, Oct. 2012.
6. C. Chen, M. Moreno Maza, B. Xia, and L. Yang. Computing cylindrical algebraic decomposition via triangular decomposition. In *ISSAC'09*, pages 95–102, 2009.
7. G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. *Springer Lecture Notes in Computer Science*, 33:515–532, 1975.
8. G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 12(3):299–328, 1991.
9. H. Hong. Simple solution formula construction in cylindrical algebraic decomposition based quantifier elimination. In *ISSAC*, pages 177–188, 1992.
10. T. Sturm and A. Tiwari. Verification and synthesis using real quantifier elimination. In *ISSAC*, pages 329–336, 2011.
11. A. Tarski. *A decision method for elementary algebra and geometry*. University of California Press, 1951.