数式処理 *J.JSSAC (2003)*
Vol. 10, No. 1, pp. 34 - 40
特集 “Quantifier Elimination”

SyNRAC: A New Maple Package for Supporting Design and Analysis in Engineering

Hitoshi YANAMI [*] Hirokazu ANAI [†]

Fujitsu Laboratories Ltd. Fujitsu Laboratories Ltd.

Abstract

We present SyNRAC, a symbolic computation package for solving real algebraic constraints. Algorithms in SyNRAC are implemented on the Maple software and they are expected to provide many users with symbolic methods of dealing with practical problems in engineering. SyNRAC is to be aimed to support symbolic, numerical, and symbolic-numeric computation. Until now we have emphasized on symbolic methods and implemented two types of special QE algorithms. Our project's outline, the present state of SyNRAC, and our future plan are stated.

1 A Brief History of QE

Quantifier elimination (QE) is a procedure that takes a first-order formula as input and returns a quantifier-free formula equivalent to the input. The history of QE dates back to the 1930s when Tarski proved the existence of a decision procedure for theory of real closed fields. Tarski also presented the first quantifier elimination algorithm, which was later modified by Seidenberg and is now known as the Seidenberg-Tarski algorithm [3, 10]. The algorithm requires too much computation, far from handling practical problems.

In 1975 Collins [4, 5] achieved a historic breakthrough in QE. He provided a new decision method based on his concept of cylindrical algebraic decomposition (CAD). The complexity of the CAD algorithm is doubly exponential in the number of quantified variables, which greatly improved the Seidenberg-Tarski algorithm. Though Collins's CAD method still could solve only very simple problems, it stimulated researchers to study this area.

In the 1980s many revised algorithms and related research papers were presented. Some groups of people started implementing QE algorithms around 1990. The QEPCAD and

[*]yanami@flab.fujitsu.co.jp
[†]anai@jp.fujitsu.com

REDLOG packages have been known as pioneers and by far the famous ones; QE algorithms were also implemented on Mathematica and on Risa/Asir. For QEPCAD and REDLOG, see the respective articles in the present issue. Among the known general QE methods, the partial-CAD-based algorithm is known as the most organized and efficient one; the algorithm has been implemented in the QEPCAD package. In spite of the quality of the algorithm, its worst-case computation has still doubly exponential behavior.

This fact promoted another point of view in studying QE and new approaches—QE algorithms that target only a certain type of formulas—were attempted in the 1990s. The method, known as the special QE method, enables us to make use of typical features of the prescribed formulas to reduce its computational complexity. With the help of these theoretical improvement as well as the year-by-year growth in hardware power such as CPU performance and memory size, various types of engineering problems can now be dealt with in reasonable time and space.

2 The SyNRAC Project

We have been developing a toolbox including special QE methods on the Maple software. The toolbox has been named SyNRAC, which stands for a Symbolic-Numeric toolbox for Real Algebraic Constraints. The SyNRAC project started late last year after some successful results in solving practical engineering problems were obtained by using symbolic computation methods.

The reasons why we have chosen Maple as a platform are: 1) Though Maple has been very popular among researchers in engineering, there are no QE tools available; 2) Maple provides us with a stable and friendly development environment; 3) Maple packages are automatically incorporated into MATLAB, which is widely used in engineering, via its "Symbolic Math Toolbox."

Based on SyNRAC we are planning to develop some toolboxes on MATLAB tailored for specific application fields such as robust control design. Those are expected to be novel tools and would provide engineers with new systematic design procedures using symbolic, numerical, and symbolic-numeric methods.

3 The Present State of SyNRAC

It is just more than half a year since our SyNRAC project started. We are currently focusing on some known *special QE algorithms,* which are specialized to particular types of input formulas. Two sorts of special QE algorithms have been implemented so far and are now available in SyNRAC; one is a special QE algorithm for the sign definite condition (SDC) and the other is a special QE algorithm for linear formulas. These types of formulas cover a wide range of problems and the algorithms implemented have enough efficiency to solve various practical problems.

These algorithms are each built around respective techniques. The former is based on the Sturm-Habicht sequence; the latter on virtual substitution. We briefly explain them in the next two subsections. Details of these algorithms are described in Anai and Yanami [2].

3.1 Special QE Using the Sturm-Habicht Sequence

The Sturm-Habicht sequence associated to a polynomial in $\mathbb{R}[x]$ is, in a word, a revised version of the well-known Sturm sequence. See [8] for the definition of the Sturm-Habicht sequence. The Sturm-Habicht sequence with respect to $f(x) \in \mathbb{R}[x]$ is a list of polynomials and is used for counting the number of real zeros of f in a given interval. The number of zeros of an interval, say, between a and b is calculated as follows: (1) Substitute a for x in the Sturm-Habicht sequence and count the number of sign changes over the resulting sequence; denote it by N_a; (2) Do the same thing for the other endpoint b to obtain N_b; (3) The answer is the absolute value $|N_a - N_b|$ of the difference of these two numbers.

González-Vega [7] used the sequence to eliminate the quantifier in a first-order formula of the form $\forall x,\ f(x) > 0$, where $f(x) \in \mathbb{R}[x]$. Anai and Hara [1] examined a similar type of formulas

$$\forall x > 0,\ f(x) > 0\ ,$$

and called this condition the *sign definite condition* (SDC). They also showed that a vast range of problems arising in engineering can be reduced to SDC and succeeded in solving them symbolically. It is easy to see that SDC is equivalent to the condition that f has no real zeros in the interval $(0, +\infty)$, plus f takes a positive value for some $x > 0$. By considering all the possible sign changes over the Sturm-Habicht sequence, we obtain an equivalent formula without the quantified variable x.

3.2 Special QE by Virtual Substitution

The other QE algorithm implemented in SyNRAC deals with linear formulas by virtual substitution. A formula is called *linear* when every atomic subformula in it is linear with respect to its quantified variables, i.e., it is of the form

$$a_0 + a_1 x_1 + \cdots + a_n x_n\ \rho\ 0\ ,$$

where $x_1, \ldots, x_n$ are the quantified variables, each term a_i, $0 \leq i \leq n$, contains no quantified variables, and $\rho \in \{=, \neq, \leq, <\}$.

In 1988 Weispfenning [11] proposed a QE algorithm by virtual substitution for linear formulas. The main idea for eliminating x in a formula $\exists x \varphi(x)$ is to find a suitable *finite* set of terms $S = \{a_1, \ldots, a_n\}$ each term of which contains no x, so that the formula and $\varphi(x//a_1) \vee \cdots \vee \varphi(x//a_n)$ are equivalent. Here $\varphi(x//a_i)$ is the formula obtained by a modified substitution.[1)]

A theorem in [11] says that when the input is a linear formula, one can find S, called an *elimination set*, satisfying the above condition, and moreover, the resulting formula $\vee_{s \in S}\varphi(x//s)$ is again linear. This fact, as well as the equivalence

$$\forall x \varphi(x) \Longleftrightarrow \neg(\exists x \neg\varphi(x))$$

with φ quantifier-free,[2)] enables us to eliminate all the quantifiers in a given linear formula; remove them one by one from inside.

Loos and Weispfenning [9] have found elimination sets smaller than in [11], which promote to increase algorithm's efficiency. We have implemented the algorithms in both papers.

3.3 Simplification

During a QE algorithm formulas tend to grow rapidly due to automatic formula rewriting. They are usually getting deeply nested and highly redundant. Thus simplification, a procedure that makes the quantifier-free part of a formula simpler, is important. Combined with simplification methods, a QE algorithm not only returns a readable output formula, but achieves more computational efficiency. There are many research papers on simplification; see [6, 7] for possible simplification methods.

The difficulty in simplifying formulas lies in deciding what formulas are regarded as simple and concise. Such standards vary. Each simplification method is based on a certain standard, and distinct simplification methods might be mutually incompatible.

In SyNRAC only primitive simplification methods have been implemented. Implementation of more complicated ones is ongoing, which is our primary concern. After that it is required to coordinate various simplification techniques to enhance efficiency.

4 Examples of Computation

In this section we show some computational examples[3)] to illustrate how algorithms in SyNRAC work. Figure 1 shows a Maple worksheet in which we have executed some command in SyNRAC. Let us explain them in order.

[1)]There is a procedure assigning the expression $\varphi(x/a_i)$ obtained from φ by substituting t for x a formula equivalent to it. We denote the resulting formula by $\varphi(x//a_i)$.

[2)]The negation '$\neg$' that precedes a quantifier-free formula can be easily eliminated if required; use De Morgan's law and rewrite the atomic subformulas.

[3)]All computations were executed on a Pentium III 1.0 GHz processor.

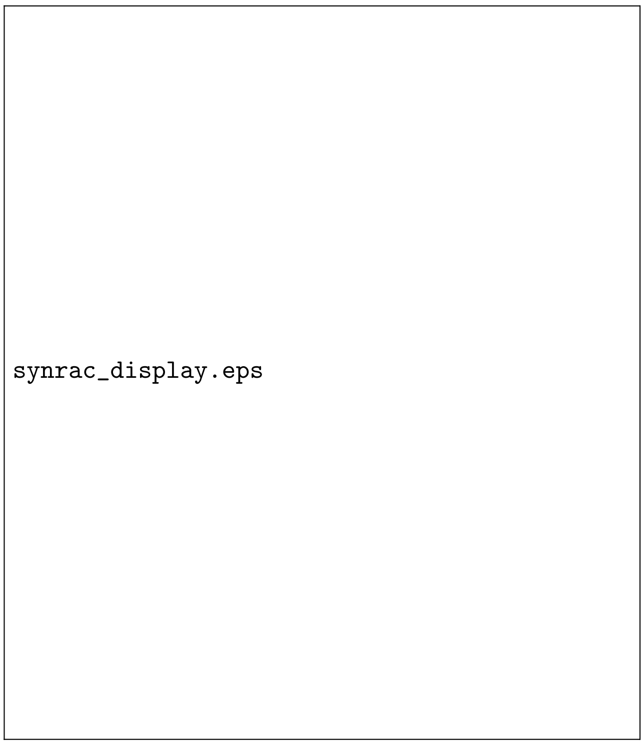

Figure 1: Examples of QE commands in SyNRAC

We load the next packages:

```
> read "synrac";   with(combinat);
```

First we solve the QE problem $\forall x > 0,\ x^2 + a_1 x + a_0 > 0$:

```
> qe_sdc(x^2+a1*x+a0,x);

                   -a0 < 0 and  a1 < 0 and -4*a0+a1^2 < 0 or
                   -a0 < 0 and -a1 < 0 and -4*a0+a1^2 < 0 or
                   -a0 < 0 and -a1 < 0 and (-4*a0+a1^2) &= 0 or
                   -a0 < 0 and -a1 < 0 and 4*a0-a1^2 < 0 or
                   a0 &= 0 and -a1 < 0 or
                   a0 &= 0 and a1 &= 0 or
                   -a0 < 0 and a1 &= 0

  time = 0.02, bytes = 123614
```

Note that the symbol `&=` is defined as equality.[4] Next we solve the existential linear QE problem $\exists x \exists y (y > 2x + 3 \wedge x > 0 \wedge y < s)$:

```
> qe_lin([x,y], y>2*x+3 and x>0 and y<s);
```

[4] As for the usual `=`, it tests object equality of the Maple representations of the expressions, which is not the same as we expect.

```
                                      -1/2*s < -3/2

  time = 0.03, bytes = 144686
```

Finally we show the examples of decision problems for both commands:

```
> qe_sdc(x^5-x^2+3*x-9,x);
                                         false
  time = 1.11, bytes = 8774262

> qe_lin([x,y], y<2*x+2 and y<=-3*x+12 and y>(1/3)*x+5);
                                          true
                    A sample point:  [x, y], [52/25, 144/25]

  time = 0.03, bytes = 155078
```

5 Future Work

We have presented our SyNRAC project and the algorithms now available in SyNRAC. Our project is currently under development and there is still a considerable way to go until the latest techniques in real quantifier elimination are implemented.

We are continually improving the efficiency of implemented algorithms and are going to implement other algorithms—including numerical and symbolic-numeric algorithms—for solving real algebraic constraints into SyNRAC. We also plan to develop some toolboxes tailored for specific applications such as parametric robust control toolbox based on SyNRAC.

Being based on the Maple software, we believe, gives our toolbox a great advantage. In order to make our system applicable to the Maple users who are interested in but not familiar with symbolic computation methods, we are going to incorporate SyNRAC into MATLAB and implement interfaces to modeling formulas and sophisticated visualization facility of feasible parameter regions in a parameter space.

References

[1] H. Anai and S. Hara. Fixed-structure robust controller synthesis based on sign definite condition by a special quantifier elimination. In *Proceedings of American Control Conference 2000*, pages 1312–1316, 2000.

[2] H. Anai and H. Yanami. SyNRAC: A maple-package for solving real algebraic constraints. In *Proceedings of the International Workshop on Computer Algebra Systems and Their Applications: CASA'2003, P.M.A. Sloot et al.(ICCS 2003) editors*, volume 2657 of *LNCS*. Springer-Verlag, 2003.

[3] A. Seidenberg. A new decision method for elementary algebra. *Annals of Math*, 60:365–374, 1954.

[4] G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In H. Brakhage, editor, *Automata Theory and Formal Languages. 2nd GI Conference*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183. Gesellschaft für Informatik, Springer-Verlag, Berlin, Heidelberg, New York, 1975.

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

[6] A. Dolzmann and T. Sturm. Simplification of quantifier-free formulae over ordered fields. *Journal of Symbolic Computation*, 24(2):209–231, Aug. 1997.

[7] L. González-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, pages 365–375. Springer-Verlag, 1998.

[8] L. González-Vega, H. Lombardi, T. Recio, and M.-F. Roy:. Sturm-habicht sequence. In *Proceedings of ISSAC'89*, pages 136–146, Portland, 1989. ACM Press.

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

[10] A. Tarski. A decision method for elementary algebra and geometry. *University of California Press*, Berkeley, 1951. (2nd edition, revised.)

[11] V. Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1–2):3–27, Feb.–Apr. 1988.

logo_A_mono.eps