Development of SyNRAC

—Real Quantifier Elimination Based on Cylindrical Algebraic Decomposition and Visialization—

Hitoshi Yanami[1,2] and Hirokazu Anai[1,2]

[1] Information Technology Core Laboratories, Fujitsu Laboratories Ltd.,
Kamikodanaka 4-1-1, Nakahara-ku, Kawasaki 211-8588, Japan
yanami@flab.fujitsu.co.jp, anai@jp.fujitsu.com
[2] CREST, Japan Science and Technology Agency,
Kawaguchi Center Building, 4-1-8, Honcho, Kawaguchi 332-0012, Japan

Abstract. We present newly implemented functions in SyNRAC, which is a Maple package for solving real algebraic constraints derived from various engineering problems. The current version of SyNRAC has added quantifier elimination (QE) by cylindrical algebraic decomposition (CAD), a general QE procedure. We also show a visualization tool for representing the possble region of an output quantifier-free formula for the two-dimensional case.

1 Introduction

Nowadays symbolic computation methods have been widely applied to solving scientific and engineering problems, which has been caused by the efficient symbolic algorithms introduced or improved for these few decades and by the advancement of computer technology that has hugely increased the CPU power and memory capacity.

We have been developing a Maple toolbox, called SyNRAC, for solving real algebraic constraints, or first-order formulas over the reals. SyNRAC stands for a Symbolic-Numeric toolbox for Real Algebraic Constraints and is aimed at being a comprehensive toolbox including a collection of symbolic, numerical, and symbolic-numeric solvers for real algebraic constraints derived from engineering problems and the biological sciences. Our main method is quantifier elimination (QE), which is a procedure removing the quantified variables from a given formula to return a quantifier-free equivalent.

In this paper we present newly implemented procedures in SyNRAC. In [1] two types of special QE methods as well as some simplification procedures of quantifier-free formulas had been implemented in SyNRAC. In [2] QE by virtual substitution for weakly parametric linear formulas and a preliminary version of QE by CAD were implemented. The newly implemented functions in the current version of SyNRAC are following:

- general QE by cylindrical algebraic decomposition
- visualization of a resulting quantifier-free formula (plotting of 2-D CAD)

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part II, LNCS 3992, pp. 462–469, 2006.

Against the inherent computational complexity of QE based on a CAD algorithm, several researchers have focused on QE algorithms specialized to particular types of input formulas; see [3, 4, 5, 6, 7]. This direction is quite promising in practice since a number of important problems in engineering have been successfully reduced to a certain type of input formulas and resolved by using specialized QE algorithms. For the concrete applications of these schemes, see [8, 9, 10, 11, 12, 13]. However, there still remain many significant problems in engineering that cannot be recast as such particular formulas. Therefore, it is strongly desired to develop an efficient algorithm to realize CAD. These new features extend the applicability and tractability of SyNRAC for solving real algebraic constraints in science and engineering.

This paper is organized as follows. We explain some changes in expressing formulas in SyNRAC in Section 2. We briefly describe CAD in Section 3 and in Section 4 we show some exapmle commands of QE by CAD and our plan for combining QE by CAD with numerical methods. In Section 5, a tool for visualizing two-dimensional possible region are shown. We state a conclusion and our future work in Section 6.

2 Formula Description on Maple 9.5

SyNRAC has been developed on the Maple software. When the SyNRAC project started, we were doing our implementation on Maple 8. We were using relational and logical operators bundled in Maple in the very first stage, but it turned out that some of those operators were unsuitable for our purpose. Here is a simple example. Let x be just an indeterminate (called a name in Maple). The `evalb` command, which evaluates a Maple expression in Boolean context, returns `false` if $x = 0$ is input. This behavior does not meet our expectation, because we want to remain $x = 0$ unchanged and undecided until some real value is assigned to x. We introduced a user-defined operator `&=` to avoid such a reaction and replaced it for the Maple's equal sign '='. We have also defined `&and` and `&or`, as taking a list of atomic formulas as an argument because binary logical operators `and` and `or` in Maple had a disadvantage when dealing with long formulas.

Since then Maple 9, Maple 9.5 and Maple 10 have been released. We are now developing SyNRAC on Maple 9.5. The most influential change is that the Logic Package has been introduced in Maple 9.5. (We have not used Maple 10 yet, but we hear that the Logic Package in Maple 10 is the same as in Maple 9.5.)

The package has its own set of logical operators such as `&and`, `&or`, and `&not`, as well as several commands for manipulating expressions. For example, `Normalize` converts the input into a disjunctive or conjunctive normal form and `BooleanSimplify` changes a given Boolean expression into a minimal sum of prime implicants. `Import` and `Export` transform a certain type of Maple expression into a form available in the Logic Package, which enables users to convert a Boolean expression represented by built-in logical operators into one suitable for the Logic Package, and vice versa.

The names `&and` and `&or` in the Logic Package are coincidentally the same as those defined in SyNRAC on Maple 8. One of the differences was that `&and` (or `&or`) in the Logic Package takes a sequence as input, while our counterpart had been defined as taking a list of atomic formulas as input, like `&and([a>0, b>0, c>0])`. We have modified our `&and` and `&or` to receive a sequence of atomic formulas as input so that we can apply commands in the Logic Package such as `Logic[Normalize]` to quantifier-free formulas in SyNRAC.

3 Cylindrical Algebraic Decomposition

Cylindrical algebraic decomposition (CAD) was discovered by Collins in 1973; see [14] for his monumental work. Collins also proposed a general QE algorithm based on CAD, which has provided a powerful method for solving real algebraic constraints.

Let A be a finite subset of $\mathbb{Z}[x_1, \ldots, x_n]$. An *algebraic decomposition* for A is a collection of mutually disjoint, semi-algebraic, A-invariant sets that partitions the Euclidean n-space E^n. To define the term *cylindrical,* we explain three parts of a CAD procedure—the projection phase, the base phase, and the lifting phase.

In the projection phase of a CAD, the PROJ function plays a central role. Let r be an integer greater than 1. PROJ maps a finite set of integral polynomials in r variables to a finite set of integral polynomials in $r-1$ variables: for $A_r \subset \mathbb{Z}[x_1, \ldots, x_r]$, $\mathrm{PROJ}(A_r) \subset \mathbb{Z}[x_1, \ldots, x_{r-1}]$. For a given $A \subset \mathbb{Z}[x_1, \ldots, x_n]$, we obtain a list

$$A = A_0 \overset{\mathrm{PROJ}}{\mapsto} A_1 \overset{\mathrm{PROJ}}{\mapsto} A_2 \overset{\mathrm{PROJ}}{\mapsto} \cdots \overset{\mathrm{PROJ}}{\mapsto} A_{n-1},$$

where $A_i \subset \mathbb{Z}[x_1, \ldots, x_{n-i}]$.

In the base phase we partition E^1 by using a set of univariate polynomials $A_{n-1} \subset \mathbb{Z}[x_1]$; we find all the real zeros of A_{n-1} and partition E^1 into A_{n-1}-invariant regions that consist of the zeros of A_{n-1} and the remaining open intervals. These points and intervals are called sections and sectors, respectively.

The lifting phase inductively constructs a decomposition of E^{i+1} from the decomposition of E^i, $i = 1, 2, \ldots, n-1$. Suppose D is a decomposition of E^i. A lifting of D is a decomposition $\bar{D}$ of E^{i+1} obtained by decomposing the space $R \times E^1$ by using A_{n-i-1} for each region $R \in D$ and putting all of them together. Let R be a region of a decomposition D of E^i. $R \times E^1$ is decomposed by the following; Take a point $(p_1, \ldots, p_i)$ in R and substitute it for $(x_1, \ldots, x_i)$ in each polynomial in A_{n-i-1} to obtain a set of univariate polynomials in x_{i+1}; Partition E^1 into, say, L_0, L_1, …, L_{2k+1} by using the roots of the polynomials in x_{i+1}; Regard $R \times L_0$, $R \times L_1$, …, $R \times L_{2k+1}$ as the resulting decomposition. The condition for this process to work is that A_{n-i-1} is *delineable* on R, in other words, every pair of polynomials in A_{n-i-1} has no intersections on R. In such a case the decomposition is independent of the choice of a sample point. A decomposition D of E^r is *cylindrical* if it is constructed by iterating the above lifting method, i.e., $r = 1$ and E^1 is decomposed as in the base phase, or $r > 1$ and D is a lifting of some cylindrical decomposition D' of E^{r-1}.

Given a formula φ one can construct a CAD for the polynomials of the atomic formulas in φ. The point for CAD-based QE is that the truth value of φ is determined regionwise because each region in the CAD is A-invariant. See [14] for details.

It is the PROJ function that is crucial in a CAD procedure. The fewer polynomials PROJ produces, the more efficient the CAD program becomes. But PROJ must be constructed to maintain the delineability and make the lifting phase possible. Some improvements in the projection phase of CAD are found in [15, 16, 17]. In the next section, we show some QE commands based on CAD. Our present implementation of QE by CAD is based on Collins's original method; there is still a long way to catch up with latest tools such as QEPCAD.

4 Quantifier Elimination Based on CAD

4.1 Commands for QE by CAD

Here we show an example of the QE by CAD command in SyNRAC. Consider $\psi = \exists y\ (x^2 + y^2 \leq 1\ \wedge\ x^3 - y^2 < 0)$. To solve this formula, we first construct a CAD for $A = \{x^2+y^2-1, x^3-y^2\} \subset \mathbb{Z}[x, y]$. The graph of these two polynomials is shown in Fig. 1. The `Projection` command repeats PROJ and returns `P[1]`$=$ $\mathrm{PROJ}^0(A) = A$ and `P[2]`$= \mathrm{PROJ}(A) = \{x^2 - 1, x^3 + x^2 - 1, x\}$.

The `Base` command partitions E^1 by using `P[2]` and returns a list of points that represent respective sections or sectors. A rational point is taken as a sample point for a sector, and a unique vanishing polynomial and an isolated interval are taken for a section. There are four real roots (sections) in `P[2]` and they make five open intervals (sectors).

```
> Base(P[2], x);

  [-2, &algn(x+1,-1,-1), -1/2, &algn(x,0,0), 3/8,
   &algn(x^3+x^2-1,3/4,7/8), 15/16, &algn(x-1,1,1), 2]
```

where `&algn(`$f(x), l, r$`)` represents the algebraic number that is a unique root of $f(x)$ lying between the two rational numbers l and r $(l \leq r)$.

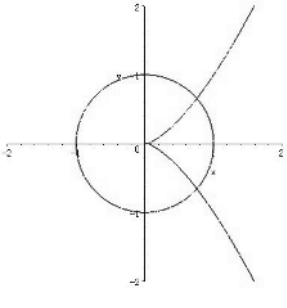

Fig. 1. The graph of A

The `Lifting` command makes a stack for each section or sector. Out of the nine regions, we have the fifth one displayed. The fifth region is a sector with a rational sample point `[3/8]` and the stack on it is represented in a list of nine sample points of sections/sectors.

```
> L:=Lifting(P, [x,y]):
> op(L[5]);

  [3/8, -2], [3/8, &algn(64*y^2-55,-1,-1/2)], [3/8, -1/2],
  [3/8, &algn(512*y^2-27,-1/2,0)],[3/8, 0],[3/8, &algn(512*y^2-27,0,1/2)],
  [3/8, 1/2], [3/8, &algn(64*y^2-55,1/2,1)], [3/8, 2]
```

And the `QEbyCAD` command returns a quantifier-free equivalent to the input formula ψ by using the sample points:

```
> QEbyCAD(&Ex([y],&and(x^2 + y^2 <= 1, x^3 - y^2 < 0)));

  &or(&and(0 < -x^2+1, 0 < -x, 0 <= x+1, x <= 0),
      &and(x^2-1 = 0, 0 <= x+1, x <= -1), &and(x = 0, 0 <= x, x <= 0),
      &and(0 < x, 0 <= x, 0 < -x^3-x^2+1, 8*x <= 7))
```

4.2 Advantage of QE Based on CAD

Though QE by CAD has a bad computational complexity (doubly exponential in terms of the number of variables), it has some advantages against specialized QE algorithms that have already been implemented in SyNRAC, such as QE by the Sturm-Habicht sequence for positive definiteness of an univariate polynomial with parametric coefficients and QE by virtual substitution for linear and quadratic formulas with respect to quantified variables:

- QE by CAD is a general-purpose QE algorithm and can handle any first-order formulas.
- Moreover, one of the big advantages of QE by CAD is its ability to produce simple solution formula.

Large quantifier-free formulas can be simplified well by using QE by CAD due to the second advantage. QE by CAD enables us to simplify the output formulas generated by a specialized QE algorithm, which are mostly too large to understand (see `http://www.cs.usna.edu/{~}qepcad/SLFQ/Home.html`). For a very large formula, repeated operations of QE by CAD to its subformulas may succeed when a direct application of QE by CAD to the entire formula fails.

4.3 Symbolic-Numeric CAD

Unfortunately, QE by CAD is not practical on real computers, since a CAD procedure usually consists of many purely symbolic computations and has inherent bad computational complexity. Therefore, it is strongly desired to develop an efficient algorithm to realize CAD. One way of tackling this is, as was stated in the previous subsection, to consider a specialized algorithm for restricted input

formulas. Another effective way for more efficient CAD is achieved by utilizing numerical computation and by making use of the derived numerical information on algebraic numbers to the full extent—until it violates the correctness of the results—instead of by handling symbolic computation. So far there are some related works to introduce numerical computation into CAD construction [18].

This strategy greatly improves the efficiency, while this causes uncertainty of the computed results depending on accuracy of numerical computation. Hence we need an effective validation scheme of the results. Now we are developing a symbolic-numeric procedure using as small a number of symbolic computations as possible—they are used only when validating unreliable numerical results. Symbolic reconstruction procedure in the lifting phase is improved if we utilize a dynamic evaluation (DE) technique [19, 20, 21] combined with successive representations of algebraic extensions.

5 Visualization

Here we show a newly introduced function for visualizing output of QE computation, in particular, in the two-dimensional case. By QE computaiton we obtain a quantifier-free formula in the unquantified variables. We can use the command `DrawDnf2d(dnf,xmin,xmax,ymin,ymax,grid)` to visualize the CAD of the two-dimensional space with colored feasible regions if the output formula is given by a disjunctive normal form (DNF). The range where the result is shown is given by the rectangle $[\texttt{xmin}, \texttt{xmax}] \times [\texttt{ymin}, \texttt{ymax}]$. This process is done by analyzing the topology of the CAD, sweeping through from left to right with repeated increments of a stride, and marking the appropriate points to the model of the CAD. This does not change the topology of the model; it makes the picture look prettier. In the plotting where the view is $[-2, 2] \times [-2, 2]$, its stride is given by $1/grid$. The DNF given as $(x^2+y^2-1 > 0 \wedge -x+y > 0) \vee (x^2-y > 0)$ is drawn as follows;

```
> DrawDnf2d := define_external('DrawDnf2d','MAPLE','LIB'="DrawDnf2d.dll");
  DrawDnf2d := proc () option call_external;
  call_external(0, 22024192, true, args) end proc

> dnf := "[[x^2+y^2-1>0,-x+y>0]][[x^2-y>0]]";

> DrawDnf2d(dnf, -2, 2, -2, 2, 50);
  "ok..."
```

Our visualizer, as you see, is an external library written in C, hence we call it from maple via "DrawDnf2d.dll." Though QE by CAD can generate "adjacency information" for two-dimensional CADs which allows it to generate topologically correct plots, we do not care for the adjacency in this implementation.

6 Conclusion and Future Work

We have presented newly developed functions in Maple-package SyNRAC. The current version of SyNRAC provides QE by CAD and visualization of 2-D resulting

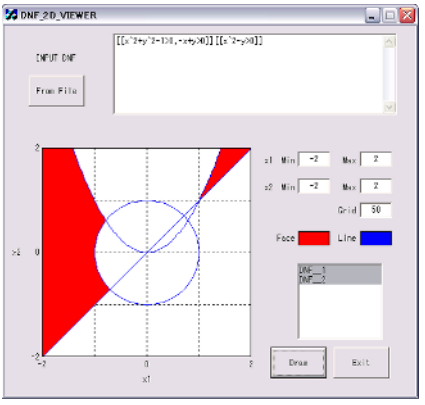

Fig. 2. 2-D visualization

quantifier-free formulas. The new features greatly extend the applicability and tractability of SyNRAC for solving real algebraic constraints in engineering.

We are planning to combine QE by CAD with some numerical methods to improve efficiency. We proceed to implement other known QE algorithms and improve them, and are setting about developing symbolic-numeric algorithms. We also plan to advance our toolbox for parametric robust control design on MATLAB using SyNRAC as a core engine [13], which we keep upgrading.

Acknowledgements. The authors thank Hong Myunghoon for the C code of the 2-D visualization tool in the present paper.

References

1. Anai, H., Yanami, H.: SyNRAC: A maple-package for solving real algebraic constraints. In: Proceedings of International Workshop on Computer Algebra Systems and their Applications (CASA) 2003 (Saint Petersburg, Russian Federation), P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2657, Springer (2003) 828–837
2. Yanami, H., Anai, H.: Development of synrac – formula description and new functions. In: Proceedings of International Workshop on Computer Algebra Systems and their Applications (CASA) 2004 : ICCS 2004, LNCS 3039, Springer (2004) 286–294
3. Weispfenning, V.: The complexity of linear problems in fields. Journal of Symbolic Computation **5** (1988) 3–27
4. Loos, R., Weispfenning, V.: Applying linear quantifier elimination. The Computer Journal **36** (1993) 450–462 Special issue on computational quantifier elimination.
5. Weispfenning, V.: Quantifier elimination for real algebra—the quadratic case and beyond. Applicable Algebra in Engineering Communication and Computing **8** (1997) 85–101
6. Hong, H.: Quantifier elimination for formulas constrained by quadratic equations. In Bronstein, M., ed.: Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 93), Kiev, Ukraine, ACM, ACM Press (1993) 264–274

7. González-Vega, L.: A combinatorial algorithm solving some quantifier elimination problems. In Caviness, B., Johnson, J., eds.: Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts and Monographs in Symbolic Computation. Springer, Wien, New York (1998) 365–375
8. Weispfenning, V.: Applying quantifier elimination to problems in simulation and optimization. Technical Report MIP-9607, FMI, Universität Passau, D-94030 Passau, Germany (1996) To appear in the Journal of Symbolic Computation.
9. Sturm, T., Weispfenning, V.: Rounding and blending of solids by a real elimination method. In Sydow, A., ed.: Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling, and Applied Mathematics (IMACS 97). Volume 2., Berlin, IMACS, Wissenschaft & Technik Verlag (1997) 727–732
10. Dolzmann, A., Sturm, T., Weispfenning, V.: Real quantifier elimination in practice. In Matzat, B.H., Greuel, G.M., Hiss, G., eds.: Algorithmic Algebra and Number Theory. Springer, Berlin (1998) 221–247
11. González-Vega, L.: Applying quantifier elimination to the Birkhoff interpolation problem. Journal of Symbolic Computation (1996) To appear.
12. Anai, H., Hara, S.: Fixed-structure robust controller synthesis based on sign definite condition by a special quantifier elimination. In: Proceedings of American Control Conference 2000. (2000) 1312–1316
13. Anai, H., Yanami, H., Sakabe, K., Hara, S.: Fixed-structure robust controller synthesis based on symbolic-numeric computation: design algorithms with a cacsd toolbox (invited paper). In: Proceedings of CCA/ISIC/CACSD 2004 (Taipei, Taiwan). (2004) 1540–1545
14. Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In Caviness, B., Johnson, J., eds.: Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts and Monographs in Symbolic Computation. Springer, Wien, New York (1998) 85–121
15. Hong, H.: An improvement of the projection operator in cylindrical algebraic decomposition. In Caviness, B., Johnson, J., eds.: Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts and Monographs in Symbolic Computation. Springer, Wien, New York (1998) 166–173
16. McCallum, S.: An improved projection operation for cylindrical algebraic decomposition. In Caviness, B., Johnson, J., eds.: Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts and Monographs in Symbolic Computation. Springer, Wien, New York (1998) 242–268
17. Brown, C.W.: Improved projection for cylindrical algebraic decomposition. Journal of Symbolic Computation **32** (2001) 447–465
18. Anai, H., Yokoyama, K.: Numerical cylindrical algebraic decomposition with certificated reconstruction. In: Proceedings of 11th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics (Fukuoka, Japan). (2004) 31
19. Duval, D.: Algebraic numbers : an example of dynamic evaluation. Journal of Symbolic Computation **18** (1994) 429–445
20. Duval, D., González-Vega, L.: Dynamic evaluation and real closure. Math. Comput. Simulation **42** (1996) no. 4-6, 551–560
21. Hong, H.: An efficient method for analyzing the topology of plane real algebraic curves. Selected papers presented at the international IMACS symposium on Symbolic computation, new trends and developments. Elsevier Science Publishers B. V., Springer, Amsterdam, The Netherlands (1996) 571-582