数理解析研究所講究録
1514巻 2006年 81-86

A Quantifier Elimination Procedure Based on Cylindrical Algebraic Decomposition in SyNRAC

屋並 仁史
YANAMI HITOSHI*
(株)富士通研究所/(独)科学技術振興機構
FUJITSU LABORATORIES LTD./CREST, JST†

穴井 宏和
ANAI HIROKAZU
(株)富士通研究所/(独)科学技術振興機構
FUJITSU LABORATORIES LTD./CREST, JST‡

Abstract

We have been developing SyNRAC, a toolbox on Maple for solving real algebraic constraints derived from various engineering problems. SyNRAC is used as a core engine on which MATLAB toolboxes are built. We have implemented quantifier elimination (QE) by Collins in which cylindrical algebraic decomposition (CAD) plays a central role. A visualization tool for representing the possible region of an output quantifier-free formula for the two-dimensional case is also added in SyNRAC.

1 Introduction

Symbolic computation methods have been more widely applied to solving scientific and engineering problems, due to the efficient symbolic algorithms introduced or improved for these few decades and to 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. Among them a CAD-based QE procedure is our first attempt to implement a general QE algorithm. 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

*This work has been partially supported by CREST, Japan Science and Technology Agency.
†yanami@labs.fujitsu.com
‡anai@jp.fujitsu.com

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)

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 briefly describe CAD in Section 2 and in Section 3 we show some example commands of QE by CAD. In Section 4, a tool for visualizing two-dimensional possible region are shown. We state a conclusion and our future work in Section 5.

2 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 $\mathbf{Z}[x_1, \ldots, x_n]$. 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 procedure, 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 $B \subset \mathbf{Z}[x_1, \ldots, x_r]$, $\mathrm{PROJ}(B) \subset \mathbf{Z}[x_1, \ldots, x_{r-1}]$ (Here B is thought of as a subset of $\mathbf{Z}[x_1, \ldots, x_{r-1}][x_r]$.) For a given $A \subset \mathbf{Z}[x_1, \ldots, x_n]$, 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 $A_i \subset \mathbf{Z}[x_1, \ldots, x_{n-i}]$.

The base phase partitions E^1 by using a set of univariate polynomials $A_{n-1} \subset \mathbf{Z}[x_1]$; it finds all the real zeros of A_{n-1} and partitions 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 that of E^i, $i = 1, \ldots, n-1$. Suppose D is a decomposition of E^i. A lifting of D is a decomposition $\tilde{D}$ of E^{i+1} obtained by decomposing the space $S \times E^1$ by using A_{n-i-1} for each region $S \in D$ and putting all of them together. Let S be a region of a decomposition D of E^i. $S \times E^1$ is decomposed by the following; Take a point $(p_1, \ldots, p_i)$ in S and substitute it for $(x_1, \ldots, x_i)$ in each polynomial in A_{n-i-1} to obtain a set A'_{n-i-1} of univariate polynomials in x_{i+1}; Partition E^1 into A'_{n-i-1}-invariant regions, say, L_1, L_2, ..., L_{2k+1} by using the zeros of A'_{n-i-1}; Regard $S \times L_1$, $S \times L_2$, ..., $S \times L_{2k+1}$ as the resulting decomposition. The condition for this

process to work is that every polynomial in A_{n-i-1} is *delineable* on S, roughly speaking, no polynomial in A_{n-i-1} has vertical tangents and self intersections, and none of the pairs of polynomials in A_{n-i-1} has intersections on $S \times \mathbb{R}$. 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}. In every lifting step, D contains a decomposition of the 'cylinder' $S \times E^1$ for every region S of D'.

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.

3 Quantifier elimination based on CAD

Here we show an example of the CAD-based QE 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 $A = \{x^2+y^2-1, x^3-y^2\} \subset \mathbf{Z}[x, y]$. 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]`$= \mathrm{PROJ}(A) = \{x^2 - 1, x^3 + x^2 - 1, x\}$.

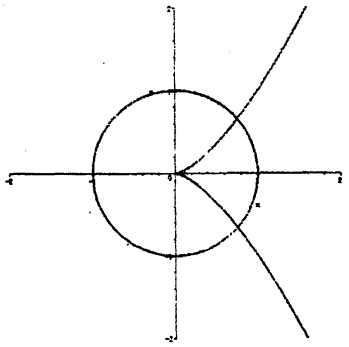

Figure 1: The graph of A

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 ($l \leq r$).

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 Visualization

Here we show a newly introduced function for visualizing output of QE computation, in particular, in the two-dimensional case. By QE computation 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 `[xmin, xmax]` × `[ymin, 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.

5 Conclusion and Future Work

Newly developed functions in Maple-package SyNRAC have been presented. The current version of SyNRAC provides QE by CAD and visualization of 2-D resulting quantifier-free formulas. The new

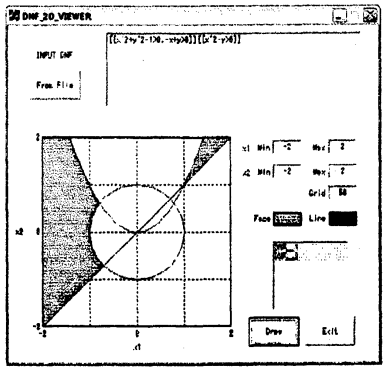

Figure 2: 2-D visualization

features greatly extend the applicability and tractability of SyNRAC for solving real algebraic constraints in engineering.

Our CAD-based QE procedure can be improved by introducing more up-to-date methods. We are planning to combining our procedure with some numerical methods to further improve efficiency. And it is also important to implement more sophisticated simplification procedures that help obtain simpler output formulas. 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