Development of SyNRAC—Formula Description and New Functions

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. In this paper we present newly developed functions in Maple-package SyNRAC, for solving real algebraic constraints derived from various engineering problems. The current version of SyNRAC provides quantifier elimination (QE) for the quadratic case and an environment dealing with first-order formulas over the reals (including new simplifiers of formulas) on Maple.

1 Introduction

We presented Maple-package SyNRAC for solving real algebraic constraints in 2003 [1]. SyNRAC stands for a Symbolic-Numeric toolbox for Real Algebraic Constraints and is aimed to be a comprehensive toolbox composed of a collection of symbolic, numeric, and symbolic-numeric solvers for real algebraic constraints derived from various engineering problems.

In this paper we show the current status of development of SyNRAC. In the previous version of SyNRAC [1] the following algorithms were available

- a special QE by the Sturm-Habicht sequence for sign definite condition,
- a special QE by virtual substitution for linear formulas,
- some naive simplifications of quantifier-free formulas.

Besides, the current version of SyNRAC provides the following:

- an environment dealing with first-order formulas over the reals,
- a special QE by virtual substitution for quadratic formulas,
- some new standard simplifiers of formulas.

Since we firstly presented SyNRAC, we have introduced some new operational symbols and fixed a notation system for expressing formulas. We are now developing our tool under the basis of the new environment. The QE algorithms previously equipped have also been reimplemented after the latest setting. These new features greatly extend the applicability and tractability of SyNRAC for solving real algebraic constraints in engineering. The current notation for first-order logic over the reals is much easier to read than the previous one. This helps users describe mathematical formulas for various types of real algebraic constraints. A

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3039, pp. 286–294, 2004.

special QE method for quadratic formulas widens the application areas of SyNRAC in actual problems (see [2]). The simplifiers can reduce the size of a given formula. This contributes not only to improve recognition of formulas but also to remarkably improve the efficiency of special QE procedures based on virtual substitution.

Furthermore, using SyNRAC as a kernel, we are now pushing the further development of design tools based on computer algebra (in particular, QE) in various application fields: One successful attempt is the development of a toolbox for *parametric robust control design* on MATLAB [3] based on the authors' previous works concerning QE-based robust control design [4,5,6,7].

2 A New Environment for First-Order Formulas over the Reals

When we say a real algebraic constraint, what we have in mind is a first-order formula over the reals. We describe what type of formulas we are dealing with and how they are expressed in SyNRAC.

An *atomic formula* is an equality or inequality $f(x_1, \ldots, x_n)\ \rho\ g(x_1, \ldots, x_n)$, where f and g are polynomials in a finite number of indeterminates over $\mathbb{Q}$ and ρ is one of the relational operators $\{=, \neq, \leq, <\}$. A *formula* is a string obtained by appropriately arranging atomic formulas, logical operators, or existential/universal quantifiers. Here is an example of existential formulas with respect to x, y, and z

$$\exists x \exists y \exists z\ (f_1 \wedge f_2 \wedge (h_1 \vee h_2) \wedge f_3) \Longrightarrow \neg(g_1 \wedge g_2)\ ,$$

where f_i, g_i, and h_i are atomic formulas.

To express formulas in SyNRAC, we need to prepare and fix notational symbols for $\exists$, $\forall$, $\wedge$, $\vee$, $\neg$, and so forth. In the earlier stages of implementation, we were using relational and logical operators bundled in Maple. As we proceeded, it turned out that some of the Maple's operators are unsuitable for our purpose. Let us show a simple example. Let x be just an indeterminate. The `evalb` command, which evaluates a relation in Boolean context, in Maple returns `false` when $x = 0$ is input. This behavior does not meet our expectation, because we want to remain $x = 0$ unchanged unless x is assigned a value.

To avoid such reactions, we have introduced a user-defined operator `&=`[1] and replaced it for the Maple's equality symbol '='. To maintain consistency, the other relational operators are redefined by adding "`&`" at the forefront of the respective commands (see Table 1). Some of them are just an alias for the Maple's corresponding command. Logical operators and quantifier symbols have also been redefined in the same way as in Tables 2 and 3. In SyNRAC, the atomic formula $x^2 - 2y^2 - 3z^2 \leq xy - 6yz - z + 7$ is expressed in

[1] A Maple user can form a neutral operator symbol by using `&name` (the ampersand character "`&`" followed by one or more characters).

Table 1. The relational operators in SyNRAC

Operator	$=$	$\neq$	$\leq$	$<$	$\geq$	$>$
Notation	`&=`	`&<>`	`&<=`	`&<`	`&>=`	`&>`

Table 2. The logical operators in SyNRAC

Operator	$\wedge$	$\vee$	$\neg$	$\Longrightarrow$	$\Longleftarrow$	$\Longleftrightarrow$
Notation	`&and`	`&or`	`&not`	`&impl`	`&repl`	`&equiv`

Table 3. The quantifiers in SyNRAC

Operator	$\exists x_1 \cdots \exists x_n \varphi$	$\forall x_1 \cdots \forall x_n \varphi$
Notation	`&Ex`$([x_1, \ldots, x_n], \varphi)$	`&All`$([x_1, \ldots, x_n], \varphi)$

`(x^2-2*y^2-3*z^2) &<= (x*y-6*y*z-z+7)` .[2]

The example formula above is expressed in the following:

`&Ex([x,y,z], (`f_1 `&and` f_2 `&and (`h_1 `&or` h_2`) &and` f_3`) &impl &not(`g_1 `&and` g_2`))` .

The operators `&and` and `&or` can also be used as a prefix operator, taking a list of operands as an argument. The expression `&and([`f_1`,` f_2`, ...,` f_n`])` is equivalent in SyNRAC to f_1 `&and` f_2 `&and` $\cdots$ `&and` f_n.

According to these notational rules, QE algorithms has been (re)implemented in SyNRAC. In addition, several basic utility functions on formulas are provided in SyNRAC, for example, functions for counting the number of atomic formulas, extracting atomic formulas from a formula as a list, and so on. Moreover, some computations for the disjunctive normal form[3] are also available.

3 Solving Quadratic Algebraic Constraints over the Reals

Here we briefly explain a special QE by virtual substitution of parametric test points that is applicable to formulas in which the quantified variables appear at most quadratically (see [8] for details). We call a formula whose atomic subformulas are at most quadratic (linear) with respect to its quantified variables a *quadratic* (*linear*) formula, respectively.

Let

$$\psi(p_1, \ldots, p_m) \equiv Q_1 x_1 \cdots Q_n x_n \varphi(p_1, \ldots, p_m, x_1, \ldots, x_n)$$

[2] The polynomials both sides should be enclosed within parentheses since the user-defined operator `&name` in Maple has higher priority than the basic arithmetic operators. In the examples in the present paper, we leave them out when too convoluted.

[3] A formula is called a *disjunctive normal form* if it is a disjunction (a sequence of $\vee$'s) consisting of one or more disjuncts, each of which is a conjunction (a sequence of $\wedge$'s) of one or more atomic formulas.

be a linear or quadratic formula, where $Q_i \in \{\forall, \exists\}$ and φ is a quantifier-free formula. By using the equivalence $\forall x\varphi(x) \Longleftrightarrow \neg(\exists x \neg\varphi(x))$, we can change the formula into an equivalent one of the form $(\neg)\exists x_1 \cdots (\neg)\exists x_n\ (\neg)\varphi$. The negation '$\neg$' that precedes a quantifier-free formula can be easily eliminated (use De Morgan's law and rewrite the atomic subformulas), which is not essential part of QE. Therefore we may focus our attention on an *existential formula,* i.e., a formula of the form $\exists x_1 \cdots \exists x_n \varphi(p_1, \ldots, p_m, x_1, \ldots, x_n)$. Furthermore, it is sufficient to show how to eliminate $\exists x$ in $\exists x\varphi$, since all the quantifiers in the formula can be eliminated by removing one by one from the innermost one.

Now our main purpose is to eliminate the quantified variable $\exists x$ in

$$\psi'(p_1, \ldots, p_m) \equiv \exists x\ \varphi(p_1, \ldots, p_m, x)$$

with $\varphi(p_1, \ldots, p_m, x)$ quantifier-free and quadratic, and obtain an equivalent quantifier-free formula $\psi'(p_1, \ldots, p_m)$. For fixed real values $q_1, \ldots, q_m$ for the parameters $p_1, \ldots, p_m$, all polynomials appearing in $\varphi(x)$ are linear or quadratic. Therefore, the set $M = \{r \in \mathbb{R} | \varphi(q_1, \ldots, q_m, r)\}$ of real values r for x satisfying φ is a finite union of closed, open, or half-open intervals over $\mathbb{R}$. The endpoints of these intervals are among $\pm\infty$ and the real zeros of atomic formulas in φ. Then candidate terms, say, $t_1, \ldots, t_k$, for those zeros can be constructed by the solution formulas for linear or quadratic equations.

If φ does not contain any strict inequalities, all the intervals composing M are either unbounded or closed. In the closed case such an interval contains its real endpoint. So M is nonempty if and only if the substitution of $\pm\infty$ or of one of the candidate solutions t_j for x satisfies φ. Let S be the candidate set $S = \{t_1, \ldots, t_k, \pm\infty\}$. Such a set is called an *elimination set* for $\exists x\varphi$. We obtain a quantifier-free formula equivalent to $\exists x\varphi$ by substituting all candidates in S into φ *disjunctively:*

$$\exists x\varphi \Longleftrightarrow \bigvee_{t \in S} \varphi(x//t)\ .$$

We note that there is a procedure assigning the expression $\varphi(x/t)$ obtained from φ by substituting t for x an equivalent formula [8]. We denote the resulting formula by $\varphi(x//t)$.

If φ contains strict inequalities, we need to add to S other candidates of the form $s \pm \epsilon$, where s is a candidate solution for some left-hand polynomial in a strict inequality and ϵ is a positive infinitesimal.

For improving the efficiency of this method, the following two points are crucial: (i) refining the elimination set S by a scrupulous selection of a smaller number of candidates in S; (ii) integrating with sophisticated simplifications of quantifier-free formulas. SyNRAC now employs three types of elimination sets proposed in [9]. Simplifications in SyNRAC are discussed in the next section.

Moreover, (heuristic) techniques for decreasing the degree during elimination are important for raising the applicability of quadratic QE, because after one quantifier is eliminated for a quadratic case the degree of other quantified variables may increase. Only simple degree-decreasing functions are implemented in the current version of SyNRAC.

4 Simplification

In the present paper, the term simplification is used for simplification of quantifier-free formulas. When a quantifier is eliminated in a given first-order formula with a special QE procedure, its quantifier-free part usually gets larger. During a QE algorithm, formulas under manipulation tend to get extremely large, deeply nested and highly redundant. That is why simplification procedures, which equivalently change a quantifier-free formula into more concise one, are important. Utilizing simplification algorithms combined with a special QE algorithm contributes to improve not only readability of the resulting formula but efficiency of the computation.

As for simplification, Maple, on which we implement our toolbox SyNRAC, can simplify certain formulas. By using Maple's `evalb` command for the inequality $3 < 5$, the value `true` are obtained. But it does not work for, say, '$x < 3$ `and` $x < 5$'; the `evalb` command does nothing and just returns '$x < 3$ `and` $x < 5$', not '$x < 5$'. Dolzmann and Sturm [10] summarize the rule for simplifying such formulas, to be precise, the formula '$f\ \rho_1\ 0$ `and`/`or` $g\ \rho_2\ 0$', where f and g differ only by a constant c, and ρ_1 and ρ_2 are an (in)equality. They called these laws *ordering theoretical smart simplification* when $c = 0$, i.e., $f = g$ and *additive smart simplification* when $c \neq 0$, respectively.

Automatic formula simplifiers are implemented in REDLOG[4] and QEPCAD[5] (see [13,10] for possible simplifications). Several simplification rules including ordering theoretical and additive smart simplification are implemented in SyNRAC, which greatly increase the efficiency of our QE commands. These rules dramatically work especially when the number of quantified variables are large.

5 Commands in SyNRAC

In this section we show some computational examples to illustrate how commands in SyNRAC are used.[6]

First, you need to load the packages:

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

You can use `qe_sdc` to solve the formula $\forall x > 0,\ f(x) > 0$, called the *sign definite condition* (SDC). The first argument of `qe_sdc` is polynomial f and the second is the variable to be eliminated. The next example shows how to use the command to solve the problem $\forall x > 0,\ a_2x^2 + a_1x + a_0 > 0$,

```
> qe_sdc(a2*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 )
  time = 0.02, bytes = 123614
```

[4] REDLOG is a QE package based on virtual substitution on REDUCE.

[5] QEPCAD is a general QE package that is applicable to all first-order formulas based on cylindrical algebraic decomposition (CAD) [11,12].

[6] All computations were executed on a Pentium III 1 GHz processor.

By using `qe_lin` command, you can solve the existential linear QE problem. This command takes two arguments; the former is a list of quantified variables and the latter a quantifier-free formula. In the following example, `qe_lin` eliminates the two quantified variables in $\exists x \exists y(y > 2x + 3 \wedge x > 0 \wedge y < s)$ and returns a condition with regard to s.

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

                                  -1/2*s &< -3/2
  time = 0.03, bytes = 144686
```

The `qe_quad` command can deal with quadratic QE problems. You can solve the quadratic QE problem $\exists x \exists y(x^2 - 4x - 5 \leq y \wedge 3 \leq x \wedge y \leq -5s + 6)$ as follows:

```
> qe_quad(&Ex([x,y], &and[(x^2-4*x-5)&<=y, 3&<=x, y&<=(-5*s+6)]));

                                  -14+5*s &<= 0
  time = 0.03 sec, bytes = 233514
```

The two examples below show that if a decision problem is given, i.e., the input contains no free variables, each command returns the `true` or `false` value:

```
> qe_sdc(x^5-x^2+3*x-9,x);

                                    false
  time = 1.11, bytes = 8774262

> qe_lin(&Ex([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
```

A sample point is one that makes the formula true.
By calling the `qfsimple` command, you can simplify quantifier-free formulas with ordering theoretical and additive smart simplification.

```
> qfsimple((x&<5 &and x&<c &and x&>=10) &or (x&<=3 &and x&<=5 &and x&>=-5
            &and x&<>3) &or (x&>7 &and x&<=d));

         (-3+x &<= 0 &and -5-x &<= 0) &or (-x &< -7 &and -d+x &<= 0)
  time = 0.00, bytes = 44974
```

The `substsimple` command simplifies quantifier-free formulas by making use of simple atomic equations. This command repeats the following two procedures: (i) solving the linear atomic equations with only one variable in each conjunctive formula and substituting its solution for the variable as far as its influence goes; (ii) calling the `qfsimple` command and simplifying the resulting formula. These are redone until such linear equations run out.
In the next example, z in the input formula is firstly substituted by $3/2$ except in the 4th atomic one, and then by using the 1st equation in the resulting formula, x is replaced by $3/5$ in three places.

```
> substsimple(5*x&=2*z &and 9&>=3*y-x &and x+4*y+z&>0 &and 2*z-3&=0
              &and 5*x+2*y&<=z+3);
```

```
          x &= 3/5 &and -40*y &< 21 &and z &= 3/2 &and -3+4y &<= 0
time = 0.00, bytes = 97406
```

6 Examples

We show two example problems from mathematical programming and solve them with SyNRAC.

Example 1 First consider the following convex quadratic programming:
minimize $x_1^2 + x_1x_2 + 2x_2^2$,
subject to $x_1 + 4x_2 \geq 16$, $3x_1 + 2x_2 \geq 18$, $x_1 \geq 0$, $x_2 \geq 0$.

To obtain a description of the first-order formula, we add an unqualified variable z and express the problem in

$$\exists x_1 \exists x_2 (z - (x_1^2 + x_1x_2 + 2x_2^2) \geq 0 \wedge x_1 + 4x_2 \geq 16 \wedge 3x_1 + 2x_2 \geq 18 \wedge x_1 \geq 0 \wedge x_2 \geq 0) .$$

Eliminating the quantified variables x_1 and x_2, we can obtain a condition on z, from which we would obtain the range of the objective function. Quantifier elimination procedure in SyNRAC outputs the condition below in 1.78 sec:

```
&or([46 - z &<= 0, &and([567 - 16 z &<= 0,
    &or([(46 - z) &= 0, &and([46 - z &<= 0, -162 + z &<= 0]),
    &and([-466 + z &<= 0, 2659 - 40 z &<= 0]), 2659 - 40 z &<= 0])]),
    &and([46 - z &<= 0, -256 + z &<= 0])])
```

A little computation tells us that this formula is equivalent to $z \geq 46$. Thus the minimum of the objective function $x_1^2 + x_1x_2 + 2x_2^2$ equals 46.

Example 2 Next we consider the following nonconvex programming:
minimize $x_1 + 3x_2$,
subject to $x_1^2 + x_2 - 4x_1 - 3 \geq 0$, $x_1^2 + 2x_2 - 12x_1 + 32 \geq 0$, $x_1 \geq 0$, $x_2 \geq 0$.

As in the first example, we rewrite the problem by using a slack variable z into

$$\exists x_1 \exists x_2 (z - (x_1 + 3x_2) \geq 0 \wedge x_1^2 + x_2 - 4x_1 - 3 \geq 0 \wedge x_1^2 + 2x_2 - 12x_1 + 32 \geq 0 \wedge x_1 \geq 0 \wedge x_2 \geq 0) .$$

Quantifier elimination procedure as well as simplification after QE outputs the condition below in 6.12 sec:

```
-155 + 25*42^(1/2) - z &<= 0
```

Thus the minimum of the objective function $x_1 + 3x_2$ is $-155 + 25\sqrt{42}$, or approximately 7.02.

7 Conclusion

We presented a newly developed functions in Maple-package SyNRAC. The current version of SyNRAC, in particular, provides quantifier elimination for quadratic case and some standard simplifiers of formulas over the new environment for

first-order formulas over the reals on Maple. The new features greatly extend the applicability and tractability of SyNRAC for solving real algebraic constraints in engineering. We are continually improving the efficiency of implemented algorithms and are going to implement other algorithms (including symbolic-numeric algorithms) for solving real algebraic constraints into SyNRAC.

Now we note that based on SyNRAC the development of a toolbox for parametric robust control design on MATLAB is ongoing.

We are aware that there is still a considerable way for SyNRAC to be a sophisticated symbolic-numeric tool. Hence we will keep progressing to bridge the gap. Our goal is to develop innovative symbolic-numeric methods and to build novel design tools via SyNRAC for various fields in engineering.

Acknowledgements. The authors would like to thank Volker Weispfenning for his invaluable advice.

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. 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
3. Sakabe, K., Yanami, H., Anai, H., Hara, S.: A MATLAB toolbox for parametric robust control system design based on symbolic computation. In: Bulletin (Kokyuroku) of RIMS (Research Institute for Mathematical Sciences, Kyoto Univ.) Workshop on Computer Algebra—Algorithms, Implementations and Applications 2003 (15-18 December 2003), (To appear)
4. 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
5. Anai, H., Hara, S.: Linear programming approach to robust controller design by a quantifier elimination. In: Proceedings of SICE Annual Conference 2002 (Osaka, Japan). (2002) 863–869
6. Anai, H., Hara, S.: A parameter space approach for fixed-order robust controller synthesis by symbolic computation. In: Proceedings of IFAC World Congress on Automatic Control b'02. (2002)
7. Anai, H., Yanami, H., Hara, S.: SyNRAC: a maple-package for solving real algebraic constraints toward a robust parametric control toolbox. In: Proceedings of SICE Annual Conference 2003 (Fukui, Japan). (2003) 1716–1721
8. Weispfenning, V.: Quantifier elimination for real algebra—the quadratic case and beyond. Applicable Algebra in Engineering Communication and Computing **8** (1997) 85–101
9. Loos, R., Weispfenning, V.: Applying linear quantifier elimination. The Computer Journal **36** (1993) 450–462 Special issue on computational quantifier elimination.
10. Dolzmann, A., Sturm, T.: Simplification of quantifier-free formulae over ordered fields. Journal of Symbolic Computation **24** (1997) 209–231

11. Collins, G.E.: Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In Brakhage, H., ed.: Automata Theory and Formal Languages. 2nd GI Conference. Volume 33 of Lecture Notes in Computer Science., Gesellschaft für Informatik, Springer-Verlag, Berlin, Heidelberg, New York (1975) 134–183
12. Collins, G.E., Hong, H.: Partial cylindrical algebraic decomposition for quantifier elimination. Journal of Symbolic Computation **12** (1991) 299–328
13. 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-Verlag (1998) 365–375