

# SMT Solver raSAT for Polynomial Constraints

Vu Xuan Tung<sup>1</sup>, To Van Khanh<sup>2</sup>, and Mizuhito Ogawa<sup>1</sup>

<sup>1</sup> Japan Advanced Institute of Science and Technology  
{tungvx,mizuhito}@jaist.ac.jp

<sup>2</sup> University of Engineering and Technology, Vietnam National University, Hanoi  
khanhvt@vnu.edu.vn

**Abstract.** This paper presents an SMT solver **raSAT** for polynomial constraints, which aims to handle them over both reals and integers with unified methodologies, (1) **raSAT loop** for inequations, which extends the *interval constraint propagation* with testing to accelerate SAT detection, and (2) the Intermediate Value Theorem for equations over reals.

## 1 Introduction

*Polynomial constraint solving* is to find an instance that satisfies given polynomial inequations/equations. Various techniques are implemented in SMT solvers. For solving over reals (*QF\_NRA*), techniques vary **QE-CAD** (RAHD [15], Z3 from 4.3 [10]), **Virtual substitution** (SMT-RAT [5], Z3 from 3.1), **Interval constraint propagation (ICP)** [1] (iSAT3 [6], dReal [8], RSolver [16]), and **Linearization** (CORD [7]). For that over integers (*QF\_NIA*), we have **Bit-blasting** (UCLID [3], MiniSmt [17]) and **Linearization** (Barcelogic [2]).

This paper presents an SMT solver **raSAT** (refinement of approximations for SAT) for polynomial constraints over both reals and integers. For inequations, it applies a simple iterative approximation refinement, **raSAT loop**, which extends ICP with testing to accelerate SAT detection. For equations, a non-constructive reasoning based on the Intermediate Value Theorem is employed (Section 4).

In a floating point arithmetic, round-off errors may violate the soundness. To get rid of such traps, we apply the outward rounding [9] in an interval arithmetic. We also integrate **iRRAM**<sup>3</sup>, which guarantees the round-off error bounds, to confirm that a detected SAT instance is really SAT.

Currently, **raSAT** applies an incremental search, *incremental widening* and *deepening*, and an SAT-directed strategy based on measures, *SAT likelihood* and *sensitivity* (Section 3), but does not have UNSAT-directed strategies, e.g., UNSAT core. Note that the sensitivity works only with Affine intervals [14].

At the SMT Competition 2015, **raSAT** participated two categories of main tracks, *QF\_NRA* and *QF\_NIA*. **raSAT** is originally developed for *QF\_NRA*, however *QF\_NIA* is fairly easy to adapt, i.e., stop interval decompositions when the width becomes smaller than 1, and generate integer-valued test instances.

---

<sup>3</sup> <http://irram.uni-trier.de>

As the overall rating (Main Track), **raSAT** is 8<sup>th</sup> among 19 SMT solvers<sup>4</sup>. The results are summarized as

- 3<sup>rd</sup> in *QF\_NRA*, **raSAT** solved 7952 over 10184 (where Z3 4.4 solves 10000).
- 2<sup>nd</sup> in *QF\_NIA*, **raSAT** solved 7917 over 8475 (where Z3 4.4 solves 8459; CVC4 (exp) solves 8277, but with one wrong detection).

## 2 ICP Overview and raSAT Loop

Our target problem is solving nonlinear constraints. First, we discuss on solving polynomial inequations, and that for equations are shown later in Section 4. Let  $\mathbb{R}$  be the set of real numbers and  $\mathbb{R}^\infty = \mathbb{R} \cup \{-\infty, \infty\}$ . The normal arithmetic on  $\mathbb{R}$  is extended to those on  $\mathbb{R}^\infty$  as in [13]. The set of all intervals is defined as  $\mathbb{I} = \{[l, h] \mid l \leq h \in \mathbb{R}^\infty\}$ . A box for a sequence of variables  $x_1, \dots, x_n$  is of the form  $B = I_1 \times \dots \times I_n$  where  $I_1, \dots, I_n \in \mathbb{I}$ .

**Definition 1.** A polynomial inequality constraint is

$$\psi(x_1, \dots, x_n) = \bigwedge_{j=1}^m p_j(x_1, \dots, x_n) > 0$$

where  $p_j(x_1, \dots, x_n) > 0$  is an atomic polynomial inequation (API). When  $x_1, \dots, x_n$  are clear from the context, we denote  $\psi$  for  $\psi(x_1, \dots, x_n)$ ,  $p_j$  for  $p_j(x_1, \dots, x_n)$ , and  $\text{var}(p_j)$  for the set of variables appearing in  $p_j$ .

As an SMT problem,  $\psi$  is satisfiable (SAT) if there exists an assignment on variables that makes it *true*. Otherwise,  $\psi$  is said to be unsatisfiable (UNSAT). Let  $\mathbb{S}(\psi) = \{(r_1, \dots, r_n) \in \mathbb{R}^n \mid \psi(r_1, \dots, r_n) = \text{true}\}$ .

---

**Algorithm 1** ICP starting from the initial box  $B_0 = I_1 \times \dots \times I_n$

---

```

1:  $S \leftarrow \{B_0\}$  ▷ Set of boxes
2: while  $S \neq \emptyset$  do
3:    $B \leftarrow S.\text{choose}()$  ▷ Get one box from the set
4:    $B' \leftarrow \text{prune}(B, \psi)$ 
5:   if  $B' = \emptyset$  then ▷ The box does not satisfy the constraint
6:      $S \leftarrow S \setminus \{B\}$ 
7:     continue
8:   else if  $B'$  satisfies  $\psi$  by using IA then
9:     return SAT
10:  else ▷ IA cannot conclude the constraint  $\implies$  Refinement Step
11:     $\{B_1, B_2\} \leftarrow \text{split}(B')$  ▷ split  $B'$  into two smaller boxes  $B_1$  and  $B_2$ 
12:     $S \leftarrow (S \setminus \{B\}) \cup \{B_1, B_2\}$ 
13:  end if
14: end while
15: return UNSAT

```

---

<sup>4</sup> <http://smtcomp.sourceforge.net/2015/results-competition-main.shtml>

## 2.1 ICP Overview

Starting with a box  $B$ , ICP [1] tries to prove UNSAT/SAT of  $\psi$  inside  $B$  by an interval arithmetic (IA). If it fails, it iteratively decomposes boxes. Algorithm 1 describes the basic ICP for solving polynomial inequations where two functions  $prune(B, \psi)$  and  $split(B)$  satisfy the following properties.

- If  $B' = prune(B, \psi)$ , then  $B' \subseteq B$  and  $B' \cap \mathbb{S}(\psi) = B \cap \mathbb{S}(\psi)$ .
- If  $\{B_1, B_2\} = split(B)$ , then  $B = B_1 \cup B_2$  and  $B_1 \cap B_2 = \emptyset$ .

ICP concludes SAT (line 8) only when it finds a box in which the constraint becomes valid by an IA. Although the number of boxes increases exponentially, ICP always detects SAT of the inequations  $\psi$  if  $I_1, \dots, I_n$  are bounded (Fig. 1a). However, ICP may miss to detect UNSAT in cases of *kissing* and *convergence* (Fig. 1c,d).

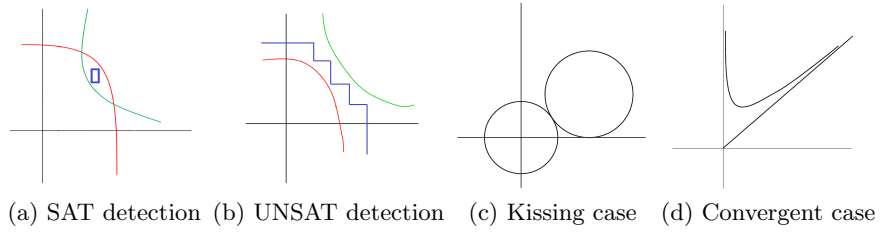


Fig. 1: Scenarios of solving polynomial inequations with ICP

## 2.2 raSAT Loop

ICP is extended to **raSAT** loop [11], which is displayed in Algorithm 2.

Its implementation **raSAT** adapts various IAs including Affine Intervals (AI) [4, 14, 11] and Classical Interval (CI) [13]. An AI introduces noise symbols  $\epsilon$ 's, which are interpreted to values in  $[-1, 1]$ . AIs vary depending on the treatments of the multiplication among noise symbols. For the multiplication of the same noise symbols,  $AF_2$  [12] describes by  $\epsilon_+$  (or  $\epsilon_-$ ), which is interpreted in  $[0, 1]$  (or  $[-1, 0]$ ), and  $CAI$  [11] describes  $\epsilon\epsilon = |\epsilon| + [-\frac{1}{4}, 0]$ . Mostly, the product of different noise symbols are simply regarded as any value in  $[-1, 1]$  (e.g.,  $\epsilon_{\pm}$ ).

Although precision is incomparable, an AI partially preserves the dependency among values, which is lost in CI. For instance, consider  $x \in [2, 4] = 3 + \epsilon$ . Then,  $x - x$  is evaluated to  $[-2, 2]$  by CI, where 0 by an AI.

*Example 1.* Let  $g = x^3 - 2xy$ ,  $x = [0, 2] = 1 + \epsilon_1$ , and  $y = [1, 3] = 2 + \epsilon_2$ . *CI* estimates the range of  $g$  as  $[-12, 8]$ , *AF<sub>2</sub>* does  $-3 - \epsilon_1 - 2\epsilon_2 + 3\epsilon_+ + 3\epsilon_{\pm}$  ( $= [-9, 6]$ ), and *CAI* does  $[-4, -\frac{11}{4}] + [-\frac{1}{4}, 0]\epsilon_1 - 2\epsilon_2 + 3|\epsilon_1| + [-2, 2]\epsilon_{\pm}$  ( $= [-8, 4.5]$ ).

## 3 SAT directed Strategies of raSAT

Performance of ICP is affected by the number of variables because the initial box  $I_1 \times \dots \times I_n$  will be decomposed into exponentially many boxes.

---

**Algorithm 2** **raSAT** loop starting from the initial box  $\Pi = \bigwedge_{i=1}^n x_i \in I_i^0$

---

```

1: while  $\Pi$  is satisfiable do ▷ Some more boxes exist
2:    $\pi = \{x_i \in I_{ik} \mid i \in \{1, \dots, n\}, k \in \{1, \dots, i_k\}\} \leftarrow$  a solution of  $\Pi$ 
3:    $B \leftarrow$  the box represented by  $\bigwedge_{i=1}^n \bigwedge_{k=1}^{i_k} x_i \in I_{ik}$ 
4:   if  $B$  does not satisfy  $\psi$  by using IA then
5:      $\Pi \leftarrow \Pi \wedge \neg(\bigwedge_{i=1}^n \bigwedge_{k=1}^{i_k} x_i \in I_{ik})$ 
6:   else if  $B$  satisfies  $\psi$  by using IA then
7:     return SAT
8:   else if  $B$  satisfies  $\psi$  by using testing then ▷ Different from ICP
9:     return SAT
10:  else ▷ Neither IA nor testing conclude the constraint  $\implies$  Refinement Step
11:    choose  $(x_i \in I_{ik}) \in \pi$  such that  $\forall k_1 \in \{1, \dots, i_k\} I_{ik} \subseteq I_{ik_1}$ 
12:     $\{I_1, I_2\} \leftarrow \text{split}(I_{ik})$  ▷ split  $I_{ik}$  into two smaller intervals  $I_1$  and  $I_2$ 
13:     $\Pi \leftarrow \Pi \wedge (x_i \in I_{ik} \leftrightarrow (x_i \in I_1 \vee x_i \in I_2)) \wedge \neg(x_i \in I_1 \wedge x_i \in I_2)$ 
14:  end if
15: end while
16: return UNSAT

```

---

### 3.1 Incremental Search

Two incremental search strategies are prepared in **raSAT**.

**Incremental Widening.** Given  $0 < \delta_0 < \delta_1 < \dots < \delta_k = \infty$ , *incremental widening* starts with  $B_0 = [-\delta_0, \delta_0] \times \dots \times [-\delta_0, \delta_0]$ , and if  $\psi$  remains UNSAT, then enlarge the box to  $B_1 = [-\delta_1, \delta_1] \times \dots \times [-\delta_1, \delta_1]$ . This continues until either timeout or the result obtained. In **raSAT**,  $AF_2$  is used for  $B_i$  if  $i \neq k$ , and  $CI$  is used for  $\delta_k = \infty$ .

**Incremental Deepening.** To combine depth-first-search and breadth-first search among decomposed boxes, **raSAT** applied *incremental deepening*. Let  $\gamma_0 > \gamma_1 > \dots > 0$ . It applies a threshold  $\gamma$ , such that no more decomposition occurs when an interval becomes smaller than  $\gamma$ .  $\gamma$  is initially  $\gamma_0$ . If neither SAT nor UNSAT is detected, **raSAT** restarts with the threshold  $\gamma_1$  (Fig. 2 (b)). This continues until either timeout or the result obtained.

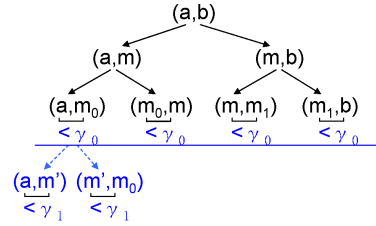


Fig. 2: Incremental Deepening

### 3.2 SAT Directed Heuristics Measure

To reduce an explosion of the number of decomposed boxes, **raSAT** prepares a strategy to select a variable at a box decomposition. (1) First, select a least likely satisfiable *API* with respect to *SAT*-

*likelihood*. (2) Then, choose a most likely influential variable in such API with respect to *sensitivity*. We assume an Affine interval (AI) for an interval arithmetic (IA).

In line 3 of Algorithm 2, let an AI estimate the range  $range(g_j, B)$  of each polynomial  $g_j$  in a box  $B$  as  $[c_1, d_1]\epsilon_1 + \dots + [c_n, d_n]\epsilon_n$ . Then,  $range(g_j, B)$  is obtained by instantiating  $[-1, 1]$  to  $\epsilon_i$ . We define

- The *SAT-likelihood* of an API  $g_j > 0$  is  $|I \cap (0, \infty)|/|I|$  for  $I = range(g_j, B)$ .
- The *sensitivity* of a variable  $x_i$  in an API  $g_j > 0$  is  $max(|c_i|, |d_i|)$ .

*Example 2.* In Example 1, SAT-likelihood of  $g$  is  $0.4 = \frac{6}{9-(-6)}$  by  $AF_2$  and  $0.36 = \frac{4.5}{4.5-(-8)}$  by  $CAI$ . The sensitivity of  $x$  is 1 by  $AF_2$  and  $3\frac{1}{4}$  by  $CAI$ , and that of  $y$  is 2 by both.

Our strategy consists of three steps:

1. Choice of an API with the least *SAT likelihood*.
2. Choice of a variable with the largest *sensitivity* in the API.
3. Choice of a box with the largest *SAT likelihood*, where the *SAT-likelihood* of a box  $B$  is the least *SAT-likelihood* among APIs on  $B$ .

The first two steps selects a variable to apply the interval decomposition and the testdata generation. After a box decomposition is applied, the last step compares *SAT-likelihood* of all boxes (including newly decomposed boxes) to select one to explore next. At the testdata generation, **raSAT** observes the sign of the coefficient of the noise symbol with the largest sensitivity. If positive, first take the upper bound; otherwise, the lower bound. The rest is generated randomly.

The strategy is evaluated on Zankl and Meti-Tarski families of  $QF\_NRA$ . As a comparison with other ICP-based solvers, we compare **raSAT** with a random choice (*Random*), **raSAT** with the strategy above (**raSAT**), **iSAT3**, and **dReal**.

Note that (1) **iSAT3** solves over  $[-1000, 1000]$  due to the system restriction, while all others solve over  $[-\infty, \infty]$ , and (2) SAT of **dReal** is  $\delta$ -SAT (SAT under tolerance of the width  $\delta$ ); thus **dReal** sometimes answers SAT even for UNSAT problems. The experiments are performed with Intel Xeon E5-2680v2 2.80GHz and 4GB of RAM. The timeout is set to 500 seconds, and the time shows the total of successful cases (either SAT or UNSAT).

Benchmark	<i>Random</i>		<b>raSAT</b>		<b>iSAT3</b>		<b>dReal</b>	
Matrix-1 (SAT )	19	230.39(s)	25	414.99	11	4.68	46*	3573.43
Matrix-1 (UNSAT)	2	0.01(s)	2	0.01	3	0.00	0	0.00
Matrix-2,3,4,5 (SAT)	1	13.43(s)	11	1264.77	3	196.40	19*	2708.89
Matrix-2,3,4,5 (UNSAT)	8	0.37(s)	8	0.38	12	8.06	0	0.00
Meti-Tarski (SAT)	3451	895.14(s)	3473	419.25	2916	811.53	3523*	441.35
Meti-Tarski (UNSAT)	1060	233.46(s)	1052	821.85	1225	73.83	1197	55.39

(\* means  $\delta$ -SAT)

We observe that the strategy is effective for SAT-detection in large problems, like Matrix-2,3,4,5 (in Zankl benchmarks), which often have more than 50 variables (Meta-Tarski has mostly less than 10 variables, and Matrix-1 has mostly less than 30 variables). Comparing with the state-of-the-art tool **Z3 4.4** on Matrix-2,3,4,5, the differences appear that **Z3 4.4** solely solves Matrix-3-7/4-12/5-6 (which have 75, 200, and 258 variables), and **raSAT** solely solves Matrix-2-3/2-8/3-5/4-3/4-9 (which have 57, 17, 81, 139, and 193 variables).

## 4 Extension for Equations Handling

**Single Equation.** A single equation ( $g = 0$ ) can be solved by finding 2 test cases with  $g > 0$  and  $g < 0$ . Then,  $g = 0$  holds somewhere in between by the Intermediate Value Theorem.

**Lemma 1.** For  $\psi = \bigwedge_{j=1}^m g_j > 0 \wedge g = 0$ . Suppose a box  $B$  and let  $[l_g, h_g] = \text{range}(g, B)$ ,

- (i) If either  $l_g > 0$  or  $h_g < 0$ , then  $g = 0$  (thus  $\psi$ ) is UNSAT in  $B$ .
- (ii) If  $\bigwedge_{j=1}^m g_j > 0$  is IA-valid in  $B$  and there are  $\mathbf{t}, \mathbf{t}' \in B$  with  $g(\mathbf{t}) > 0$  and  $g(\mathbf{t}') < 0$ , then  $\psi$  is SAT.

If neither (i) nor (ii) holds, **raSAT** continues the decomposition.

*Example 3.* Let  $\varphi = f(x, y) > 0 \wedge g(x, y) = 0$ . Suppose we find a box  $B = [a, b] \times [c, d]$  such that  $f(x, y) > 0$  is IA-VALID in  $B$ . (Fig. 3a). In addition, if we find two points  $(u_1, v_1)$  and  $(u_2, v_2)$  in  $B$  such that  $g(u_1, v_1) > 0$  and  $g(u_2, v_2) < 0$ , then the constraint is satisfiable by Lemma 1.

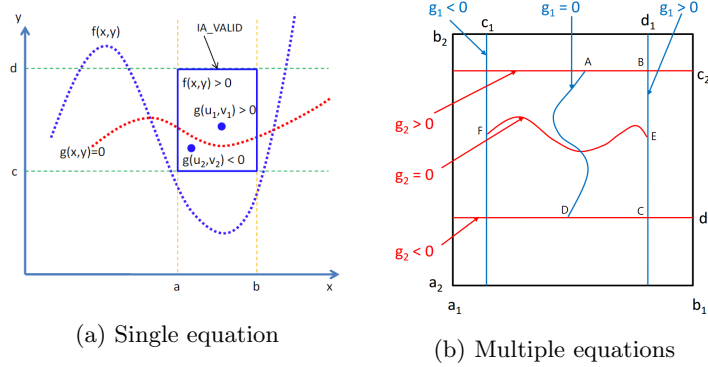


Fig. 3: Example on solving equations using the Intermediate Value Theorem

**Multiple Equations.** Multiple equations are solved by repeated applications of the Intermediate Value Theorem. Let  $\bigwedge_{j=1}^m g_j = 0$  and  $B = [l_1, h_1] \times \cdots [l_n, h_n]$  with  $m \leq n$ . Let  $V = \{x_1, \dots, x_n\}$  be the set of variables. For  $V' = \{x_{i_1}, \dots, x_{i_k}\} \subseteq V$ , we denote  $B \downarrow_{V'}$  and  $B \uparrow_{V'}$  for  $\{(r_1, \dots, r_n) \in B \mid r_i = l_i \text{ for } i = i_1, \dots, i_k\}$  and  $\{(r_1, \dots, r_n) \in B \mid r_i = h_i \text{ for } i = i_1, \dots, i_k\}$ , respectively.

**Definition 2.** A sequence  $(V_1, \dots, V_m)$  of subsets of  $V$  is a check basis of  $(g_1, \dots, g_m)$  in a box  $B$ , if, for each  $j, j'$  with  $1 \leq j, j' \leq m$ ,

1.  $V_j (\neq \emptyset) \subseteq \text{var}(g_j)$ ,
2.  $V_j \cap V_{j'} = \emptyset$  if  $j \neq j'$ , and
3. either  $g_j > 0$  on  $B \uparrow_{V_j}$  and  $g_j < 0$  on  $B \downarrow_{V_j}$ , or  $g_j < 0$  on  $B \uparrow_{V_j}$  and  $g_j > 0$  on  $B \downarrow_{V_j}$ .

**Lemma 2.** For a polynomial constraint containing multiple equations

$$\psi = \bigwedge_{j=1}^m g_j > 0 \wedge \bigwedge_{j=m+1}^{m'} g_j = 0$$

and  $B = [l_1, h_1] \times \cdots [l_n, h_n]$ , assume that

1.  $\bigwedge_{j=1}^m g_j > 0$  is IA-valid in  $B$ , and
2. there is a check basis  $(V_{m+1}, \dots, V_{m'})$  of  $(g_{m+1}, \dots, g_{m'})$  in  $B$ .

Then,  $\psi$  has a SAT instance in  $B$ .

The idea is, from the Intermediate Value Theorem, each  $j \in \{m+1, \dots, m'\}$ ,  $g_j$  has a  $n - |V_j|$  dimensional surface of null points of  $g_j$  between  $B \uparrow_{V_j}$  and  $B \downarrow_{V_j}$ . Since  $V_j$ 's are mutually disjoint (and  $g_j$ 's are continuous), we have the intersection of all such surfaces of null points with the dimension  $n - \sum_{j=m+1}^{m'} |V_j|$ . Thus, this method has a limitation that the number of variables must be greater than or equal to the number of equations.

*Example 4.* Fig. 3b illustrate Lemma 2 for  $m = 0$  and  $m' = n = 2$ . The blue and red lines represent the null points of  $g_1(x, y)$  and  $g_2(x, y)$  in  $[c_1, d_1] \times [c_2, d_2]$ , respectively.

Benchmark	raSAT				Z3 4.3				iSAT3			
	SAT		UNSAT		SAT		UNSAT		SAT		UNSAT	
Zankl (15)	<b>11</b>	0.07 (s)	<b>4</b>	0.17 (s)	<b>11</b>	0.17 (s)	<b>4</b>	0.02 (s)	0	0.00 (s)	<b>4</b>	0.05 (s)
Meti-Tarski (3528/1573)	875	174.90 (s)	781	401.15 (s)	<b>1497</b>	21.00 (s)	<b>1115</b>	74.19 (s)	1	0.28 (s)	1075	22.6 (s)
Keymaera (612)	0	0.00 (s)	312	66.63 (s)	0	0.00 (s)	<b>610</b>	2.92 (s)	0	0.00 (s)	226	1.63 (s)

Table 1: Comparison among SMT solvers with equations

## 5 Conclusion

This paper presented an SMT solver **raSAT** for polynomial constraints. There are lots of future works, including

- Support mixed integers (QF\_NIRA).
- Implement UNSAT-directed strategies, e.g. UNSAT core.
- Set search bounds based on QE-CAD.
- Apply Groebner basis to overcome current limitation that the number of variables must be greater than or equal to the number of equations.

## References

- [1] Benhamou, F., Granvilliers, L.: Continuous and Interval Constraints. In: F. Rossi, P.v.B., Walsh, T. (eds.) *Handbook of Constraint Programming*, pp. 571–604. Elsevier (2006)
- [2] Bofill, M., Nieuwenhuis, R., Oliveras, A., Rodriguez-Carbonell, E., Rubio, A.: The Barcelogic SMT Solver. In: Gupta, A., Malik, S. (eds.) *Computer Aided Verification, Lecture Notes in Computer Science*, vol. 5123, pp. 294–298. Springer Berlin Heidelberg (2008), [http://dx.doi.org/10.1007/978-3-540-70545-1\\_27](http://dx.doi.org/10.1007/978-3-540-70545-1_27)
- [3] Bryant, R.E., Kroening, D., Ouaknine, J., Seshia, S.A., Strichman, O., Brady, B.: Deciding bit-vector arithmetic with abstraction. In: *IN PROC. TACAS 2007. Lecture Notes in Computer Science*, vol. 4424, pp. 358–372. Springer (2007)
- [4] Comba, J.L.D., Stolfi, J.: *Affine Arithmetic and its Applications to Computer Graphics* (1993)
- [5] Corzilius, F., Loup, U., Junges, S., brahm, E.: SMT-RAT: An SMT-Compliant Nonlinear Real Arithmetic Toolbox. In: *Theory and Applications of Satisfiability Testing SAT 2012*, vol. 7317, pp. 442–448. Springer Berlin Heidelberg (2012)
- [6] Fränzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *Journal on Satisfiability, Boolean Modeling and Computation* 1, 209–236 (2007)
- [7] Ganai, M., Ivancic, F.: Efficient decision procedure for non-linear arithmetic constraints using CORDIC. In: *Formal Methods in Computer-Aided Design, 2009. FMCAD 2009*. pp. 61–68 (Nov 2009)
- [8] Gao, S., Kong, S., Clarke, E.: dReal: An SMT Solver for Nonlinear Theories over the Reals. In: Bonacina, M. (ed.) *Automated Deduction CADE-24, Lecture Notes in Computer Science*, vol. 7898, pp. 208–214. Springer Berlin Heidelberg (2013)
- [9] Hickey, T., Ju, Q., Van Emden, M.H.: Interval Arithmetic: From Principles to Implementation. *J. ACM* 48(5), 1038–1068 (Sep 2001)
- [10] Jovanovi, D., de Moura, L.: Solving Non-linear Arithmetic. In: Gramlich, B., Miller, D., Sattler, U. (eds.) *Automated Reasoning, Lecture Notes in Computer Science*, vol. 7364, pp. 339–354. Springer Berlin Heidelberg (2012)
- [11] Khanh, T.V., Ogawa, M.: SMT for Polynomial Constraints on Real Numbers. *Electronic Notes in Theoretical Computer Science* 289(0), 27 – 40 (2012), third Workshop on Tools for Automatic Program Analysis (TAPAS’ 2012)
- [12] Messine, F.: *Extensions of Affine Arithmetic: Application to Unconstrained Global Optimization*
- [13] Moore, R.: *Interval analysis*. Prentice-Hall series in automatic computation, Prentice-Hall (1966)



- [14] Ngoc, D.T.B., Ogawa, M.: Overflow and Roundoff Error Analysis via Model Checking. In: Proceedings of the 2009 Seventh IEEE International Conference on Software Engineering and Formal Methods. pp. 105–114. SEFM '09, IEEE Computer Society, Washington, DC, USA (2009)
- [15] Passmore, G.O., Jackson, P.B.: Combined Decision Techniques for the Existential Theory of the Reals. In: CALCULEMUS. Lecture Notes in Computer Science, vol. 5625, pp. 122–137. Springer-Verlag (2009)
- [16] Ratschan, S.: Efficient Solving of Quantified Inequality Constraints over the Real Numbers. *ACM Trans. Comput. Logic* 7(4), 723–748 (Oct 2006)
- [17] Zankl, H., Middeldorp, A.: Satisfiability of Non-linear (Ir)rational Arithmetic. In: Clarke, E., Voronkov, A. (eds.) *Logic for Programming, Artificial Intelligence, and Reasoning*. Lecture Notes in Computer Science, vol. 6355, pp. 481–500. Springer Berlin Heidelberg (2010)