

raSAT: an SMT Solver for Polynomial Constraints over Reals

Vu Xuan Tung¹, To Van Khanh², and Mizuhito Ogawa¹

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

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

Abstract. This paper presents an SMT solver **raSAT** for polynomial inequalities. It consists of a simple iterative approximation refinement, **raSAT** loop, which is an extension of the standard ICP (Interval Constraint Propagation) with testing, aiming to accelerate SAT detection. If it fails to decide, input intervals are refined by decomposition.

ICP is robust for large degrees, but the number of boxes (products of intervals) may exponentially explode with respect to the number of variables. For boosting SAT detection, we design strategies, namely *SAT likelihood* and *sensitivity*, on the choice of a variable to decompose and a box to explore.

At last, **raSAT** loop is extended for polynomial equations based on the Intermediate Value Theorem. This extension is again evaluated on Zankl, Meti-tarski, and Keymaera families. A simple modification of **raSAT** loop to handle mixed integers will be also presented.

1 Introduction

Polynomial constraint solving over reals (resp. integers) is to find an instance from reals (resp. integers) that satisfies given polynomial inequality/equality. Solving polynomial constraints on reals is decidable [21], though that on integers is undecidable (*Hilbert's 10th problem*). Quantifier Elimination by Cylindrical Algebraic Decomposition (QE-CAD) [4] is a well known technique, and implemented in Mathematica, Maple/SynRac, Reduce/Redlog, QEPCAD-B, and recently in some SMT solvers [13, 6]. QE-CAD solves more than the satisfiability, and is DEXPTIME. By restricting on the satisfiability, *Variant quantifier elimination* [12] reduces to polynomial optimization problems, which are solved by Groebner basis in EXPTIME.

A practical alternative is Interval Constraint Propagation (ICP) [2] which is implemented in **iSAT3** [7], **dReal** [10], and **RSolver** [20]. ICP is an over-approximation by an interval arithmetic, and iteratively refines by interval decomposition.

This paper presents an SMT solver **raSAT** (refinement of approximations for SAT) for polynomial constraints over reals. It consists of a simple iterative approximation refinement, **raSAT** loop, which adds testing to boost SAT

detection to the standard ICP, aiming to accelerate SAT detection. If both the estimations by an Interval Arithmetic and Testing fail, input intervals are refined by decomposition. The features of **raSAT** are,

- **raSAT** loop, which adds testing to boost SAT detection to a standard ICP,
- various Interval Arithmetic support, e.g., Affine Intervals [15, 17, 14],
- sound use of floating point arithmetic, i.e., outward rounding in interval arithmetic [11], and confirmation of an SAT instance by an error-bound guaranteed floating point package **irram**³.

ICP and **raSAT** loop are robust for large degrees, but the number of boxes (products of intervals) grows exponentially. First, we target on polynomial inequalities, and design SAT detection-directed strategies on the choice of a variable to decompose and a box to explore, and a variable to generate multiple test cases. They are based on heuristic measures, *SAT likelihood* and *sensitivity*. Another strategy of **raSAT** is an incremental avoiding local optimal.

- **Incremental widening.** Starting **raSAT** loop with a smaller initial interval, and if it is UNSAT, enlarge the input intervals and restart.
- **Incremental deepening.** Starting with the bound that each interval will be decomposed no smaller than it. If neither SAT nor UNSAT is detected, set a smaller bound and restart.

The combinations are examined on Zankl, Meti-tarski and Keymaera benchmarks from QF_NRA of SMT-LIB, to find clear differences from random choices. We also show two extensions, (1) handling polynomial equations by using the Intermediate Value Theorem, and (2) polynomial constraints over integers (e.g., AProVE benchmark in QF_NIA). These results are also compared with **Z3 4.3**, **dReal-2.15.01** and **iSAT3**.

1.1 Polynomial Constraints Over Integers

The **raSAT** loop is easily modified to QF_NIA (nonlinear arithmetic over integer numbers) from QF_NRA. We obtain SAT detection over integers by setting $\gamma_0 = 1$ in the incremental deepening in Section 3.1 and restricting test data generation on integer numbers, where UNSAT detection is the same as for QF_NIA benchmarks.

1.2 Results

We implement **raSAT** loop as an SMT solver **raSAT**, based on MiniSat 2.2 as a backbone SAT solver and the library in [1] for outward rounding in the IAs. **raSAT** can be downloaded from <http://www.jaist.ac.jp/~s1310007/raSAT/>.

³ <http://irram.uni-trier.de>

1.3 Results on SMT-COMP 2015

raSAT participated the SMT Competition 2015 on two logic divisions, namely QF_NRA and QF_NIA. The result is available at <http://smtcomp.sourceforge.net/2015/results-toc.shtml>. Briefly summarized,

- on QF_NRA, **raSAT** solved 7952 benchmarks over 10184, ending with the third place; and
- on QF_NIA, **raSAT** solved 7917 benchmarks over 8475, ending with the second place.

By the Competition-Wide Scoring, **raSAT** ranked 8th among 19 solvers.

1.4 Related Work

There are many techniques appearing in various SMT solvers.

QE-CAD RAHD [19] and Z3 4.3 (nlsat in [13]) include QE-CAD.

Virtual substitution (VS) SMT-RAT toolbox [6] combines VS, incremental DPLL, and eager theory propagation. Z3 3.1 combines VS, ICP, and linearization.

Bit-blasting UCLID [3] and MiniSmt [22] give a bound on the number of bits to encode integers and rationals, respectively.

Linearization CORD [8] linearizes multiplications of reals by CORDIC encoding. Linearization suffers from the increase of the polynomial degrees.

ICP-based SMT solvers are iSAT3 and dReal, adding to **raSAT**.

iSAT3 has tighter integration between DPLL procedure [18] and ICP. Fresh variables are introduced to decompose a polynomial to atomic representations, and each of them is assigned to an atomic proposition. A data structure is prepared to store intervals such that they correspond to the decision level one in DPLL. Its unit propagation is strengthened by combining with eager theory propagation. In a clause, if all except one literals are falsified, the remaining literal causes unit propagation and it becomes a candidate when next decomposition occurs. Note that **iSAT3** uses only CI as an IA.

dReal has different judgments on SAT, called δ -SAT, which allows the deviation of the width δ . Thus, δ -SAT does not imply really SAT. This is the reason why dReal quite often concludes SAT (actually δ -SAT) for UNSAT problems in SMT-LIB benchmarks. With the weakening of SAT to δ -SAT, it obtains the completeness of δ -SAT and δ -UNSAT [9]. Note that **dReal** uses only CI as an IA, and lazy theory propagation as **raSAT**.

2 ICP overview and raSAT loop

Our target problem is solving nonlinear constraints. We mainly discuss on polynomial inequalities, and later in Section 4, we show an extension to cover polynomial equations based on the Intermediate Value Theorem. From now on, we

use the following notations without redefine it. Let \mathbb{R} be the set of real numbers, $\mathbb{R}^\infty = \mathbb{R} \cup \{-\infty, \infty\}$. The normal arithmetics on \mathbb{R} is extended to those on \mathbb{R}^∞ as in [16]. 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 inequality (API). When x_1, \dots, x_n are clear from the context, we write ψ as $\psi(x_1, \dots, x_n)$ and similarly for $p_j(x_1, \dots, x_n)$ with $j = 1, \dots, m$. We denote the set of variables appearing in p_j by $\text{var}(p_j)$.

As an SMT problem, ψ is satisfiable (SAT) if there exist an assignment from variables to real (integer) numbers such that makes it become *true*. Otherwise, ψ is said to be unsatisfiable (UNSAT).

We denote $\mathbb{S}(\psi) = \{(r_1, \dots, r_n) \in \mathbb{R}^n \mid \psi(r_1, \dots, r_n) = \text{true}\}$ as the set of solutions of ψ .

Intuitively, starting with a box B , ICP [2] tries to prove the unsatisfiability/satisfiability of ψ inside B by iteratively applying Interval Arithmetic and decomposing boxes. In the next sections, we are going to review ICP, and then introduce **raSAT** loop [14] as an extension of ICP.

2.1 ICP overview

Algorithm 1 describes the basic ICP for solving polynomial inequalities where two functions $\text{prune}(B, \psi)$ and $\text{split}(B)$ satisfy the following properties.

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

Since ICP concludes SAT (line 8) only when it finds a box in which the constraint becomes valid by IA, it almost cannot conclude the satisfiability of equations. In contrast, although the number of boxes increases exponentially, ICP always detects SAT of φ if I_1, \dots, I_n are bounded. However, ICP may miss to detect UNSAT. Limitations for detecting UNSAT come from the *kissing* and *convergent* cases in Fig. 1. The left shows a kissing case $x^2 + y^2 < 2^2 \wedge (x-4)^2 + (y-3)^2 < 3^2$ such that $\mathbb{S}(-x^2 - y^2 + 2^2 > 0) \cap \mathbb{S}(-(x-4)^2 - (y-3)^2 + 3^2 > 0) = \{(1.6, 1.2)\}$. Thus, it cannot be separated by the covering by (enough small) boxes. The right shows a convergent case $y > x + \frac{1}{x} \wedge y < x \wedge x > 0$, i.e., $xy > x^2 + x \wedge y < x \wedge x > 0$. The latter does not appear if all intervals are bounded.

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:    $B \leftarrow S.choose()$  ▷ Get one box from the set
4:    $B' \leftarrow prune(B, \psi)$ 
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:     $\{B_1, B_2\} \leftarrow split(B')$  ▷ 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

```

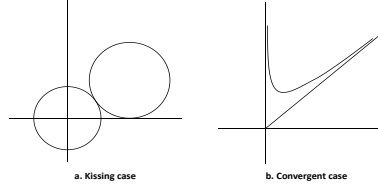


Fig. 1: Limitations for detection of UNSAT

2.2 raSAT loop

ICP is extended to **raSAT** loop [14] which is displayed in Algorithm 2. Its implementation **raSAT** adapts various IAs including Affine Intervals [5] and Classical Interval (CI) [16]. Although precision is incomparable, Affine Interval partially preserves the dependency among values, which are lost in CI. For instance, $x - x$ is evaluated to $[-2, 2]$ for $x \in [2, 4]$ by CI, but 0 by an Affine Interval. More details can be found in [14].

Example 1. Let $g = x^3 - 2xy$ with $x = [0, 2]$ ($x = 1 + \epsilon_1$) and $y = [1, 3]$ ($y = 2 + \epsilon_2$), we have,

- AF_2 estimates the range of g as $-3 - \epsilon_1 - 2\epsilon_2 + 3\epsilon_+ + 3\epsilon_\pm$, thus $[-9, 6]$,
- CAI estimates the range of g as $[-4, -\frac{11}{4}] + [-\frac{1}{4}, 0]\epsilon_1 - 2\epsilon_2 + 3|\epsilon_1| + [-2, 2]\epsilon_\pm$, thus $[-8, 4.5]$.

3 SAT directed Strategies of raSAT

During the search, the initial box $I_1 \times \dots \times I_n$ is decomposed into exponentially many boxes. As a result, performance of ICP is affected by the number of variables. The detection of UNSAT requires exhaustive search on all boxes, and thus

finding a small UNSAT core is the key. For SAT detection, the keys will be a strategic control not to fall into local optimal and a strategy to choose the most likely influential decomposition/box.

3.1 Incremental search

Two incremental search strategies are prepared in **raSAT**, (1) *incremental widening*, and (2) *incremental deepening*.

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 ψ stays UNSAT, then enlarge the box to $B_1 = [-\delta_1, \delta_1] \times \dots \times [-\delta_1, \delta_1]$. This continues until either SAT, timeout, UNSAT of $B_k = [-\delta_k, \delta_k] \times \dots \times [-\delta_k, \delta_k]$. In **raSAT**, AF_2 is used for B_i if $i \neq k$, and CI is used otherwise.

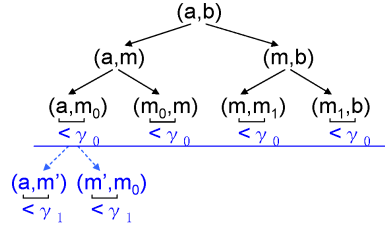


Fig. 2: Incremental Deepening

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 γ , such that no more decomposition occurs when a box becomes smaller than γ . γ is initially $\gamma = \gamma_0$. If neither SAT nor UNSAT is detected, **raSAT** restarts with the threshold γ_1 . This continues until either SAT, timeout, or a given bound of repetition is reached (Fig. 2 (b)).

3.2 SAT directed heuristics measure

In **raSAT**, a strategy to select a variable to decompose is in the following two steps. (1) First select a least likely satisfiable *API* using *SAT-likelihood*, and (2) then choose a most likely influential variable in the selected *API* using variables *sensitivity*.

In line 3 of Algorithm 2, an IA will estimate the ranges of polynomials in a box B . We denote the estimated range of g_j by $range(g_j, B)$. If the IA is an Affine Interval, the estimated range of g_j has the form $[c_1, d_1]\epsilon_1 + \dots + [c_n, d_n]\epsilon_n$ from which $range(g_j, B)$ is obtained by instantiating $[-1, 1]$ to ϵ_i . We have the following definitions.

- The *SAT-likelihood* of an API $g_j > 0$ is $|I \cap (0, \infty)|/|I|$ for $I = \text{range}(g_j, B)$.
- 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 f 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 AF_2 and CAI .

At the decomposition, **raSAT** also examines the choice of the box. We define the *SAT-likelihood* of a box B by the least SAT-likelihood of APIs. After decomposition, **raSAT** simply compares SAT-likelihood of newly decomposed boxes with the those of previous ones to select one box to explore.

Test case generation strategy The sensitivity of variables is also used for test case generation. That is, **raSAT** generates two test cases for the specified number of variables, and one for the rest. Such variables are selected from those with larger sensitivity. When two test cases are generated, **raSAT** also observes the sign of the coefficients of noise symbols. If positive, it takes the upper bound of possible values as the first test case; otherwise, the lower bound. The second test case is generated randomly.

The effects of designed strategies are examined on Zankl and Meti-Tarski families of QF_NRA, which are illustrated in Table 1. The experiments are done on a machine with Intel Xeon E5-2680v2 2.80GHz and 4 GB of RAM. The timeout is set to 500s, and time shows the total of successful cases (either SAT or UNSAT).

Benchmark	Random		Strategy	
Matrix-1 (SAT)	19	230.39 (s)	25	414.99(s)
Matrix-1 (UNSAT)	2	0.01(s)	2	0.01(s)
Matrix-2,3,4,5 (SAT)	1	13.43 (s)	11	1264.77(s)
Matrix-2,3,4,5 (UNSAT)	8	0.37(s)	8	0.38(s)
Meti-Tarski (SAT)	3451	895.14 (s)	3473	419.25 (s)
Meti-Tarski (UNSAT)	1060	233.46 (s)	1052	821.85 (s)

Table 1: Effectiveness of Heuristics

4 Extension for Equations Handling

In this section, we try a simple method based on the *Intermediate Value Theorem* which is illustrated by cases of single equation and multiple ones.

4.1 Single Equation

A single equation ($g = 0$) can be solved in a simple way by finding 2 test cases with $g > 0$ and $g < 0$, which implies $g = 0$ somewhere in between.

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 $l_g > 0$ or $h_g < 0$, then $g = 0$ is UNSAT in B and thus that is for ψ .
- (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 ψ 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.

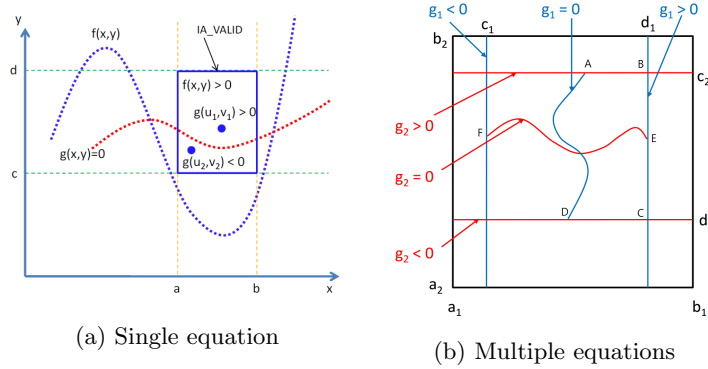


Fig. 3: Example on solving equations using the Intermediate Value Theorem

4.2 Multiple Equations

The above idea is extended for solving multiple equations. Consider m equations ($m \geq 1$): $\bigwedge_{j=1}^m g_j = 0$ and an box $B = [l_1, h_1] \times \cdots [l_n, h_n]$. Let $V = \{v_1, \dots, v_n\}$ be the set of variables. For $V' = \{x_{i_1}, \dots, x_{i_k}\} \subseteq V$, we denote $B \downarrow_{V'}$ and $B \uparrow_{V'}$ as $\{(r_1, \dots, r_n) \in B \mid r_i = l_i \text{ for } i = i_1, \dots, i_k\}$ 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) on 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 a box B , assume that

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

Then, ψ 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. Consider two equations $g_1(x, y) = 0$ and $g_2(x, y) = 0$, and assume that $(\{x\}, \{y\})$ is a check basis of (g_1, g_2) on $[c_1, d_1] \times [c_2, d_2]$ (Fig. 3b). Then, the blue line (null points of g_1) and the red line (null points of g_2) must have an intersection. We can explain this by Jordan curve theorem. Since $ABCD$ is a closed curve such that E is inner and F is outer, a continuous (red) line EF must have an intersection by Jordan curve theorem.

5 Conclusion

This paper presented **raSAT** loop, which extends ICP with testing to accelerate SAT detection and implemented as an SMT solver **raSAT**. With experiments on benchmarks from QF_NRA of SMT-LIB, we found two heuristic measures, *SAT-likelihood* and *sensitivity*, which lead an effective strategy for SAT detection.

We have some ideas for the future works.

UNSAT core. Currently, **raSAT** focuses on SAT detection. For UNSAT detection, the target is to find a small UNSAT core in a large problem.

Equations handling. Section 4 shows incomplete equations handling, we would like to additionally apply Groebner basis to overcome the limitation that the number of variables is greater-than-equal to that of equations.

References

- [1] Alliot, J.M., Gotteland, J.B., Vanaret, C., Durand, N., Gianazza, D.: Implementing an interval computation library for OCaml on x86/amd64 architectures. ICFP, ACM (2012)
- [2] Benhamou, F., Granvilliers, L.: Continuous and Interval Constraints. Handbook of Constraint Programming (2006), pp.571–604
- [3] Bryant, R.E., Kroening, D., Ouaknine, J., Seshia, S.A., Strichman, O., Brady, B.: Deciding bit-vector arithmetic with abstraction. TACAS 2007. LNCS 4424, pp. 358–372 (2007)
- [4] Collins, G.: Quantifier elimination by cylindrical algebraic decomposition - twenty years of progress. Quantifier Elimination and Cylindrical Algebraic Decomposition, pp. 8–23 (1998)
- [5] Comba, J.L.D., Stolfi, J.: Affine arithmetic and its applications to computer graphics (1993)
- [6] Corzilius, F., Loup, U., Junges, S., brahm, E.: Smt-rat: An SMT-compliant non-linear real arithmetic toolbox. SAT 2012, vol. 7317, pp. 442–448
- [7] Frnzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. JSAT 1, 209–236 (2007)
- [8] Ganai, M., Ivancic, F.: Efficient decision procedure for non-linear arithmetic constraints using cordic. FMCAD 2009. pp. 61–68 (2009).
- [9] Gao, S., Avigad, J., Clarke, E.M.: Delta-complete decision procedures for satisfiability over the reals. IJCAR’12, pp. 286–300 (2012)
- [10] Gao, S., Kong, S., Clarke, E.: dReal: An SMT solver for nonlinear theories over the reals. CADE-24, LNCS 7898, pp. 208–214 (2013)
- [11] Hickey, T., Ju, Q., Van Emden, M.H.: Interval Arithmetic: From principles to implementation. J. ACM 48(5), 1038–1068 (Sep 2001)
- [12] Hong, H., Din, M.S.E.: Variant quantifier elimination. Journal of Symbolic Computation 47(7), 883 – 901 (2012)
- [13] Jovanovi, D., de Moura, L.: Solving non-linear arithmetic. Automated Reasoning, LNCS 7364, pp. 339–354 (2012)
- [14] Khanh, T.V., Ogawa, M.: SMT for polynomial constraints on real numbers. TAPAS 2012, ENTCS 289(0), pp.27 – 40 (2012)
- [15] Messine, F.: Extensions of affine arithmetic: Application to unconstrained global optimization, Journal of Universal Computer Science 8(11), pp.992–1015 (2002)
- [16] Moore, R.: Interval analysis. Prentice-Hall (1966)
- [17] Ngoc, D.T.B., Ogawa, M.: Overflow and roundoff error analysis via model checking. SEFM 2009. pp. 105–114 (2009)
- [18] Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Abstract dpll and abstract dpll modulo theories. LPAR04, LNAI 3452. pp. 36–50 (2005)
- [19] Passmore, G.O., Jackson, P.B.: Combined decision techniques for the existential theory of the reals. CALCULEMUS. LNCS 5625, pp. 122–137 (2009)
- [20] Ratschan, S.: Efficient solving of quantified inequality constraints over the real numbers. TOCL 7(4), 723–748 (2006)
- [21] Tarski, A.: A decision method for elementary algebra and geometry. Quantifier Elimination and Cylindrical Algebraic Decomposition, pp. 24–84 (1998)
- [22] Zankl, H., Middeldorp, A.: Satisfiability of non-linear (ir)rational arithmetic. LPAR, LNCS 6355, pp. 481–500 (2010)

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

```
