raSAT : SMT Solver for Polynomial Constraints

Vu Xuan Tung[1], To Van Khanh[2], and Mizuhito Ogawa[1]

[1] Japan Advanced Institute of Science and Technology
{tungvx,mizuhito}@jaist.ac.jp
[2] University of Engineering and Technology, Vietnam National University, Hanoi
khanhtv@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, e.g., for reals (*QF_NRA*), **Cylindrical algebraic decomposition** (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 **CORDIC encoding** (CORD[7]). For integers (*QF_NIA*), we have **Bit-blasting** (UCLID[3], MiniSmt [17]) and **Linearization** (Barcelogic[2]).

This paper presents an SMT solver **raSAT**[3] (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 boost 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**[4], 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.

[3] Available at `http://www.jaist.ac.jp/~s1310007/raSAT/index.html`
[4] Available at `http://irram.uni-trier.de`

As the overall rating (Main Track), **raSAT** is 8^{th} among 19 SMT solvers[5]. The results are summarized as

- 3^{rd} in *QF_NRA*, **raSAT** solved 7952 over 10184 (where Z3 4.4 solves 10000).
- 2^{nd} 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 $x_1, \cdots, x_n$ is of the form $B = I_1 \times \cdots \times I_n$ where $I_1, \cdots, I_n \in \mathbb{I}$.

Definition 1. *A polynomial inequality constraint is*

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

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

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

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

$S \leftarrow \{B_0\}$ ▷ Set of boxes
while $S \neq \emptyset$ **do**
  $B \leftarrow S.choose()$ ▷ Get one box from the set
  $B' \leftarrow prune(B, \psi)$
  if $B' = \emptyset$ **then** ▷ The box does not satisfy the constraint
    $S \leftarrow S \setminus \{B\}$
    continue
  else if B' satisfies ψ by using *IA* **then**
    return SAT
  else ▷ *IA* cannot conclude the constraint $\Longrightarrow$ *Refinement* Step
    $\{B_1, B_2\} \leftarrow split(B')$ ▷ split B' into two smaller boxes B_1 and B_2
    $S \leftarrow (S \setminus \{B\}) \cup \{B_1, B_2\}$
  end if
end while
return UNSAT

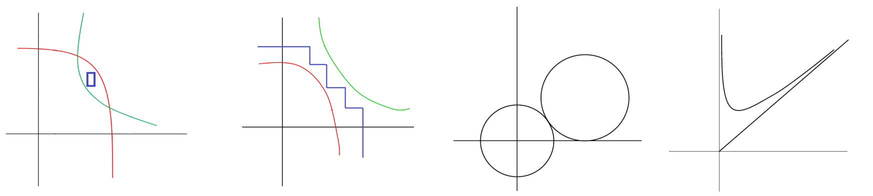
5 `http://smtcomp.sourceforge.net/2015/results-competition-main.shtml`

2.1 ICP Overview

Starting with a box B, ICP [1] tries to prove UNSAT/SAT of ψ inside B by an interval arithmetic (IA). If it fails, it iteratively decomposes boxes, and IA and constraint propagation are applied. 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 (IA-valid) by an IA. Although the number of boxes to be checked may be exponentially many, ICP always detects SAT of the inequations ψ if $I_1, \cdots, I_n$ are bounded (Fig. 1a). ICP can detect UNSAT (Fig. 1b); however ICP may miss to detect UNSAT in *kissing* or *convergent* cases (Fig. 1c,d).



(a) SAT detection (b) UNSAT detection (c) Kissing case (d) Convergent case

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 ϵ'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 ϵ_+ (or ϵ_-), 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_2 does $-3 - \epsilon_1 - 2\epsilon_2 + 3\epsilon_+ + 3\epsilon_\pm$ $(= [-9, 6])$, and CAI does $[-4, -\frac{11}{4}] + [-\frac{1}{4}, 0]\epsilon_1 - 2\epsilon_2 + \mathbf{3}|\epsilon_1| + [-2, 2]\epsilon_\pm$ $(= [-8, 4.5])$.

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

while Π is satisfiable **do** ▷ Some more boxes exist
  $\pi = \{x_i \in I_{ik} \mid i \in \{1, \cdots, n\}, k \in \{1, \cdots, i_k\}\} \leftarrow$ a solution of Π
  $B \leftarrow$ the box represented by $\bigwedge_{i=1}^{n} \bigwedge_{k=1}^{i_k} x_i \in I_{ik}$
  if B does not satisfy ψ by using IA **then**
    $\Pi \leftarrow \Pi \wedge \neg(\bigwedge_{i=1}^{n} \bigwedge_{k=1}^{i_k} x_i \in I_{ik})$
  else if B satisfies ψ by using IA **then**
    return SAT
  else if B satisfies ψ by using *testing* **then** ▷ Different from ICP
    return SAT
  else ▷ Neither IA nor *testing* conclude the constraint $\Longrightarrow$ *Refinement* Step
    choose $(x_i \in I_{ik}) \in \pi$ such that $\forall k_1 \in \{1, \cdots, i_k\} I_{ik} \subseteq I_{ik_1}$
    $\{I_1, I_2\} \leftarrow split(I_{ik})$ ▷ split I_{ik} into two smaller intervals I_1 and I_2
    $\Pi \leftarrow \Pi \wedge (x_i \in I_{ik} \leftrightarrow (x_i \in I_1 \vee x_i \in I_2)) \wedge \neg(x_i \in I_1 \wedge x_i \in I_2)$
  end if
end while
return UNSAT

3 SAT directed Strategies of raSAT

Performance of ICP is affected by the number of variables, since the initial box $I_1 \times \cdots \times I_n$ may be decomposed into exponentially many boxes. Thus, strategies for controlling a search among boxes and selecting a box / a variable to decompose are crucial for practical efficiency.

3.1 Incremental Search

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

Incremental Widening. Given $0 < \delta_0 < \delta_1 < \cdots < \delta_k = \infty$, *incremental widening* starts with $B_0 = [-\delta_0, \delta_0] \times \cdots \times [-\delta_0, \delta_0]$, and if ψ remains UNSAT, then enlarge the box to $B_1 = [-\delta_1, \delta_1] \times \cdots \times [-\delta_1, \delta_1]$. 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 and breadth first searches among decomposed boxes, **raSAT** applied *incremental deepening*. Let $\gamma_0 > \gamma_1 > \cdots > 0$. It applies a threshold γ, such that no more decomposition occurs when the size of a box becomes smaller than γ. γ is initially set to γ_0. If neither SAT nor UNSAT is detected, **raSAT** restarts with the threshold γ_1 (Fig. 2). This continues until either timeout or the result obtained.

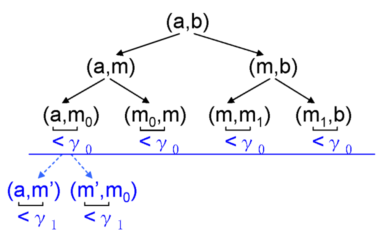


Fig. 2: Incremental Deepening

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 the *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 $[c_1, d_1]\epsilon_1 + \cdots + [c_n, d_n]\epsilon_n$. Then, $range(g_j, B)$ is evaluated by instantiating $[-1, 1]$ to ϵ_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 likelyhood*.
2. Choice of a variable with the largest *sensitivity* in the API.
3. Choice of a box with the largest *SAT likelyhood*, 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 δ-SAT (SAT under tolerance of the width δ); thus **dReal** sometimes answers SAT even for UNSAT problems. The timeout is set to 500 seconds (on Intel Xeon E5-2680v2 2.80GHz and 4GB of RAM), and the time shows the total of successful cases. Matrix-1, Matrix-2∼5, and Meti-Tarski have 53, 96, and 4884 inequality problems resp.

Benchmark	*Random*		**raSAT**		**iSAT3**		**dReal**	
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∼5 (SAT)	1	13.43(s)	11	1264.77	3	196.40	19*	2708.89
Matrix-2∼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 δ-SAT)

We observe that the strategy is effective for SAT-detection in large problems, like Matrix-2∼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∼5, the differences appear that **Z3 4.4** solely solves Matrix-3-7, 4-12, and 5-6 (which have 75, 200, and 258 variables), and **raSAT** solely solves Matrix-2-3, 2-8, 3-5, 4-3, and 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.

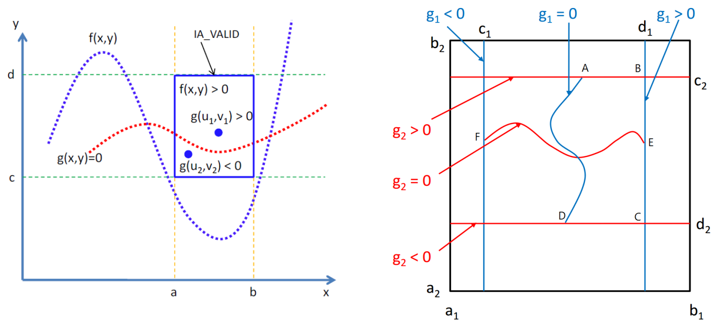
Lemma 1. *For* $\psi = \bigwedge_{j=1}^{m} g_j > 0 \wedge g = 0$. *Suppose a box* B *and let* $[l_g, h_g] = range(g, B)$,

(i) If either $l_g > 0$ *or* $h_g < 0$, *then* $g = 0$ *(thus* ψ*) is UNSAT in* B.

(ii) If $\bigwedge_{j=1}^{m} g_j > 0$ *is IA-valid in* B *and there are* $\boldsymbol{t}, \boldsymbol{t'} \in B$ *with* $g(\boldsymbol{t}) > 0$ *and* $g(\boldsymbol{t'}) < 0$, *then* ψ *is SAT.*

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

Example 3. Let $\psi = f(x, y) > 0 \wedge g(x, y) = 0$. 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.



(a) Single equation

(b) Multiple equations

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 $V = \{x_1, \cdots, x_n\}$ be the set of variables. For $V' = \{x_{i_1}, \cdots x_{i_k}\} \subseteq V$, we denote $B \downarrow_{V'}$ and $B \uparrow_{V'}$ for $\{(r_1, \cdots, r_n) \in B \mid r_i = l_i \text{ for } i = i_1, ..., i_k\}$ and $\{(r_1, \cdots, r_n) \in B \mid r_i = h_i \text{ for } i = i_1, ..., i_k\}$, respectively.

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

1. *$V_j(\neq \emptyset) \subseteq var(g_j)$,*
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 *$(V_{m+1}, \cdots, V_{m'})$ of $(g_{m+1}, \cdots, g_{m'})$ in B.*

Then, ψ has a SAT instance in B.

The idea is, from the Intermediate Value Theorem, each $j \in \{m+1, \cdots, m'\}$, 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' 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.

Fig. 3b illustrates 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. They must have an intersection somewhere.

The table below shows preliminary experiments on benchmarks with equality constraints. Note that Keymaera benchmark consists of 612 problems, but we do not know yet the exact numbers of SAT and UNSAT problems

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

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. Handbook of Constraint Programming, 571–604. Elsevier (2006)
[2] Bofill, M., Nieuwenhuis, R., Oliveras, A., Rodrguez-Carbonell, E., Rubio, A.: The BarceLogic SMT solver. CAV 2008, LNCS5123, 294–298. (2008)
[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, 358–372. (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. SAT 2012. LNCS 7317, 442–448. (2012)
[6] Frnzle, 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. FMCAD 2009. 61–68 IEEE. (2009)
[8] Gao, S., Kong, S., Clarke, E.: dReal: An SMT solver for nonlinear theories over the reals. CADE-24. LNCS 7898, 208–214. (2013)
[9] Hickey, T., Ju, Q., Van Emden, M.H.: Interval arithmetic: From principles to implementation. J. ACM 48(5), 1038–1068. (2001)
[10] Jovanovi, D., de Moura, L.: Solving non-linear arithmetic. IJCAR 2012, LNCS 7364, 339–354. (2012)
[11] Khanh, T.V., Ogawa, M.: {SMT} for polynomial constraints on real numbers. TAPAS 2012, ENTCS 289, 27–40. (2012)
[12] Messine, F.: Extensions of affine arithmetic: Application to unconstrained global optimization. J. UCS 8(11), 992-1015 (2002)
[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. SEFM 2009, 105–114, IEEE. (2009)
[15] Passmore, G.O., Jackson, P.B.: Combined decision techniques for the existential theory of the reals. CALCULEMUS. LNCS 5625, 122–137. (2009)
[16] Ratschan, S.: Efficient solving of quantified inequality constraints over the real numbers. ACM Trans. Comput. Logic 7(4), 723–748. (2006)
[17] Zankl, H., Middeldorp, A.: Satisfiability of non-linear (ir)rational arithmetic. LPAR-16, LNAI 6355, 481–500. (2010)