SMT Solver raSAT 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 both over reals and integers with unified methodologies, (1) **raSAT** *loop* for inequalities, 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 over reals (resp. integers) is to find an instance from reals (resp. integers) that satisfies given polynomial inequalities/equations. Various techniques are employed by SMT solvers. For solving over reals (referred as *QF_NRA*), techniques include **QE-CAD** (RAHD [13], Z3 from 4.3 [9]), **Virtual substitution (VS)** (SMT-RAT [4], Z3 from 3.1), **Interval constraint propagation (ICP)** [1] (iSAT3 [5], dReal [7], RSolver [14]), and **Linearization** (CORD [6]). For that over integers (referred as *QF_NIA*), **Bit-blasting** (UCLID [2], MiniSmt [15]) and **Linearization** (Barcelogic [**?**]) are prominent.

This paper presents an SMT solver **raSAT** (refinement of approximations for SAT) for polynomial constraints over both reals and integers. For inequalities, 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).

Because round-off errors affect the soundness of floating point arithmetic, outward rounding [8] is applied in interval computation. We further integrate **iRRAM**[3], which guarantees the round-off error bounds, to confirm that a detected SAT instance gives really SAT.

Currently, **raSAT** applies incremental search, namely *incremental widening* and *deepening*; and SAT-directed strategies, particularly *SAT likelihood* and *sensitivity* (Section 3), but does not prepare UNSAT-directed strategies, e.g., UNSAT core.

raSAT has participated the SMT Competition 2015 on two categories of main tracks, *QF_NRA* and *QF_NIA*. **raSAT** is originally developed for *Q*F_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] http://irram.uni-trier.de

As the overall rating (Main Track), **raSAT** is 8^{th} among 19 SMT solvers[4]. 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. We mainly discuss on solving polynomial inequalities, 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 arithmetics on $\mathbb{R}$ is extended to those on $\mathbb{R}^\infty$ as in [11]. 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 inequality (API). When $x_1, \cdots, x_n$ are clear from the context, we write ψ as $\psi(x_1, \cdots, x_n)$ and similarly for $p_j(x_1, \cdots, x_n)$ with $j = 1, \cdots, m$. We denote the set of variables appearing in p_j by $var(p_j)$.

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).

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

2.1 ICP Overview

Algorithm 1 describes the basic ICP for solving polynomial inequalities 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 IA. Although the number of boxes increases exponentially, ICP always detects SAT of the inequalities ψ if $I_1, \cdots, I_n$ are bounded. However, ICP may miss to detect UNSAT in cases of *kissing* and *convergence* (Figure 1).

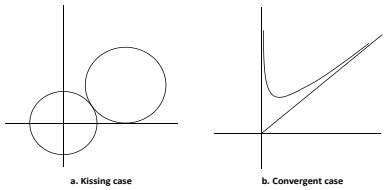


Fig. 1: Limitations for detection of UNSAT

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

2.2 raSAT Loop

ICP is extended to **raSAT** loop [10] which is displayed in Algorithm 2. Its implementation **raSAT** adapts various IAs including Affine Interval [3, 12, 10] and Classical Interval (CI) [11]. Although precision is incomparable, Affine Interval partially preserves the dependency among values, which is 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 [10].

Example 1. Let $g = x^3 - 2xy$, $x = [0, 2] = 1 + \epsilon_1$ and $y = [1, 3] = 2 + \epsilon_2$. While AF_2 [10] estimates the range of g as $-3 - \epsilon_1 - 2\epsilon_2 + 3\epsilon_+ + 3\epsilon_\pm$ (thus $[-9, 6]$), CAI [10] does as $[-4, -\frac{11}{4}] + [-\frac{1}{4}, 0]\epsilon_1 - 2\epsilon_2 + \mathbf{3}|\epsilon_1| + [-2, 2]\epsilon_\pm$ (thus $[-8, 4.5]$).

3 SAT directed Strategies of raSAT

Performance of ICP is affected by the number of variables because the initial box $I_1 \times \cdots \times I_n$ will be decomposed into exponentially many boxes. The detection of UNSAT requires exhaustive search on all boxes, and thus finding a small

[4] `http://smtcomp.sourceforge.net/2015/results-competition-main.shtml`

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

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 $0 < \delta_0 < \delta_1 < \cdots < \delta_k = \infty$, *incremental widening* starts with $B_0 = [-\delta_0, \delta_0] \times \cdots \times [-\delta_0, \delta_0]$, and if ψ stays UNSAT, then enlarge the box to $B_1 = [-\delta_1, \delta_1] \times \cdots \times [-\delta_1, \delta_1]$. This continues until either SAT, timeout, UNSAT of $B_k = [-\delta_k, \delta_k] \times \cdots \times [-\delta_k, \delta_k]$. In **raSAT**, AF_2 is used for B_i if $i \neq k$, and CI is used otherwise.

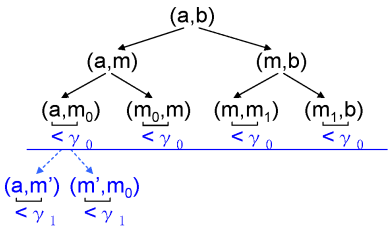


Fig. 2: Incremental Deepening

Incremental Deepening. To combine depth-first-search and breadth-first search among decomposed boxes, **raSAT** applied *incremental deepening*. Let $\gamma_0 > \gamma_1 > \cdots > 0$. 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 such API using *sensitivity*.

In line 3 of Algorithm 2, an IA estimates the range $range(g_j, B)$ of each polynomial g_j in a box B. If an Affine Interval is used, the estimated range of g_j is of the form $[c_1, d_1]\epsilon_1 + \cdots + [c_n, d_n]\epsilon_n$ 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 $I = range(g_j, B)$.
- 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.

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

Test Case Generation Strategy. The sensitivity of variables is further used in test case generation where **raSAT** 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 effects of designed strategies are examined on Zankl and Meti-Tarski families of *QF_NRA*, which are illustrated in Table 1. A brief comparison with ICP-based solvers is also presented in the table where UNSAT of **raSAT** and **dReal** is over $[-\infty, \infty]$ while that of **iSAT3** is over $[-1000, 1000]$ (otherwise **iSAT3** often causes segmentation faults with $[-\infty, \infty]$). In addition, **dReal** concludes either $\delta-$SAT which cannot imply SAT, or UNSAT. 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).

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.

Benchmark	Random				Strategy				iSAT3				dReal			
	SAT		UNSAT		SAT		UNSAT		SAT		UNSAT		δ−SAT		UNSAT	
Zankl	20	243.82 (s)	10	0.38(s)	**36**	1679.76(s)	10	0.39(s)	14	201.08(s)	**15**	8.06 (s)	65	6282.32(s)	0	0(s)
Meti-Tarski	3451	895.14 (s)	1060	233.46 (s)	**3473**	419.25 (s)	1052	821.85 (s)	2916	811.53(s)	**1225**	73.83 (s)	3523	441.35 (s)	1197	55.39 (s)

Table 1: Effectiveness of Heuristics

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.

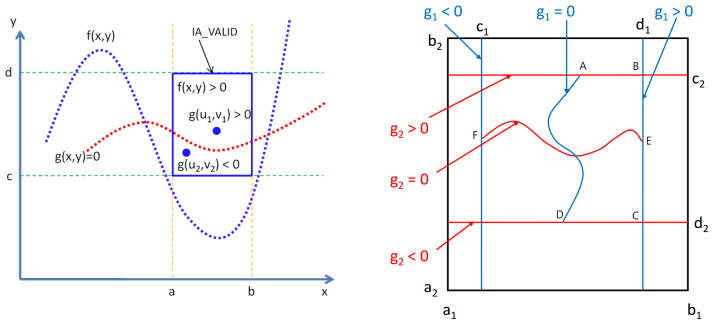
Lemma 1. *For* $\psi = \bigwedge_{j=1}^{m} g_j > 0 \wedge g = 0$. *Suppose a box* B *and let* $[l_g, h_g] = range(g, B)$,

(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* $\boldsymbol{t}, \boldsymbol{t'} \in B$ *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 $\varphi = f(x, y) > 0 \wedge g(x, y) = 0$. 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.



(a) Single equation

(b) Multiple equations

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

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 $V = \{v_1, \cdots, v_n\}$ be the set of variables. For $V' = \{x_{i_1}, \cdots x_{i_k}\} \subseteq V$, we denote $B \downarrow_{V'}$ and $B \uparrow_{V'}$ as $\{(r_1, \cdots, r_n) \in B \mid r_i = l_i \text{ for } i = i_1, ..., i_k\}$ 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 $(g_1, \cdots, g_m)$ on 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 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 *$(V_{m+1}, \cdots, V_{m'})$ of $(g_{m+1}, \cdots, g_{m'})$ on B.*

Then, ψ has a SAT instance in B.

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

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] 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] 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)

[3] Comba, J.L.D., Stolfi, J.: Affine arithmetic and its applications to computer graphics (1993)

[4] 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)

[5] 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)

[6] 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)

[7] 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)

[8] Hickey, T., Ju, Q., Van Emden, M.H.: Interval arithmetic: From principles to implementation. J. ACM 48(5), 1038–1068 (Sep 2001)

[9] 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)

[10] 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)

[11] Moore, R.: Interval analysis. Prentice-Hall series in automatic computation, Prentice-Hall (1966)

[12] 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)

[13] 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)

[14] Ratschan, S.: Efficient solving of quantified inequality constraints over the real numbers. ACM Trans. Comput. Logic 7(4), 723–748 (Oct 2006)

[15] 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)