Appendix: A Complete Algorithm for Optimization Modulo Nonlinear Real Arithmetic

Fuqi Jia[1, 3], Yuhang Dong[2, 3], Rui Han[1, 3], Pei Huang[4], Minghao Liu[5], Feifei Ma[2, 3], Jian Zhang[1, 3] *

[1]State Key Laboratory of Computer Science, ISCAS, Beijing, China
[2]Laboratory of Parallel Software and Computational Science, ISCAS, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
[4] Stanford University, Stanford, USA
[5] University of Oxford, Oxford, UK
{jiafq, maff, zj}@ios.ac.cn

Introduction

In this appendix, we provide supplementary materials that support the main text of this work. The appendix is organized as follows:

- **Theory Preliminaries**: This section defines the essential concepts, operators, and components that form the foundation of our algorithm, ensuring clarity before presenting further details.
- **Proof Details**: We provide complete and rigorous proofs of Theorem 3 and 4, ensuring theoretical soundness.
- **Algorithm Pseudocode**: This section presents the step-by-step pseudocode for the proposed algorithms, offering a clear guide for implementation and understanding.
- **Supplementary Results**: Additional experimental results are included here to further validate our findings, providing extended insights and supporting evidence.

Algorithm Preliminary

Given a set of variables $\boldsymbol{x} = \{x_1, \cdots, x_n\}$, a term t_i is a finite production of powers of variables, that is, $t_i = \prod_{j=1}^{n} x_j^{d_{i,j}}$, where $d_{i,j} \in \mathbb{N}$ is the degree of the variable x_j. The *degree* of a term is $\sum_{j=1}^{n} d_{i,j}$. A polynomial $p \in \mathbb{Q}[\boldsymbol{x}]$ of *general form* is a finite sum of terms, that is, $p = \sum_{i=1}^{m} c_i t_i$, where $c_i \in \mathbb{Q}$ is the coefficient of the term t_i. In addition, the equivalent *recursive form* of polynomial $p \in \mathbb{Q}[x_1, \cdots, x_n]$, where $x_1 \prec x_2 \prec \cdots \prec x_n$ as the current order, $p = a_m x_n^{d_m} + a_{m-1} x_n^{d_{m-1}} + \cdots + a_0$, where $d_1, \cdots, d_m \in \mathbb{N}$, $d_1 < \cdots < d_m$, and the coefficients a_i are polynomials, i.e., $a_i \in \mathbb{Q}[x_1, \cdots, x_{n-1}]$ with $a_m \neq 0$. x_i denotes *main variable*, d_m is *degree*, a_m is the *leading coefficient* of p (denoted as *lcoeff*(p)), and a_0 is the *trailing coefficient* (the coefficient independent of x_n) of p (denoted as *tcoeff*(p)).

Example 1. *Given a set of variables $\{x_1, x_2, x_3\}$ and variable order $x_1 \prec x_2 \prec x_3$,*

$$\begin{aligned} p(x_1, x_2, x_3) &:= x_1 x_2 x_3^2 + x_2 x_3^2 + x_3^2 + x_1 x_3 + x_2 x_3 + 3 \\ &:= ((x_1 + 1)x_2)x_3^2 + (x_1 + x_2)x_3 + 3. \end{aligned}$$

The degrees of the terms $x_1x_2x_3^2, x_2x_3^2, x_3^2, x_1x_3, x_2x_3$ are 4,3,2,2,2, while the degree of the polynomial $p(x_1, x_2, x_3)$ is 2, the leading coefficient is $((x_1 + 1)x_2)$, and the trailing coefficient is 3.

Definition 1 (Resultant)**.** *Let p_1, p_2 be two polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$. Assume that*

$$p_1 = a_m x_n^{d_m} + a_{m-1} x_n^{d_{m-1}} + \cdots + a_0,$$

$$p_2 = b_n x_n^{d_n} + b_{n-1} x_n^{d_{n-1}} + \cdots + b_0.$$

The resultant of p_1 and p_2 with respect to x_n, res(p_1, p_2, x_n), is:

$$\mathit{res}(p_1, p_2, x_n) = \begin{vmatrix} a_m & \cdots & a_0 & & & \\ & a_m & \cdots & a_0 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_m & \cdots & a_0 \\ b_n & \cdots & b_0 & & & \\ & b_n & \cdots & b_0 & & \\ & & \ddots & \ddots & \ddots & \\ & & & b_n & \cdots & b_0 \end{vmatrix}$$

Definition 2 (Discriminant)**.** *Let p be a polynomial in $\mathbb{Q}[x_1, \ldots, x_n]$. Assume that*

$$p = a_m x_n^{d_m} + a_{m-1} x_n^{d_{m-1}} + \cdots + a_0.$$

The discriminant of f with respect to x_n, disc(f, x_n), is:

$$\mathit{disc}(p, x_n) = \frac{(-1)^{\frac{m(m-1)}{2}}}{a_m} \mathit{res}(p, \frac{\partial p}{\partial x_n}, x_n).$$

Proof Details

We begin by introducing the correctness of the CAC algorithm, as guaranteed by Theorem 1 (Ábrahám et al. 2021; Bär et al. 2023).

Theorem 1. *Let ψ be a conjunction of polynomial atoms with $x_1, \cdots, x_n, S = \{s'_1, \cdots, s'_m\} \subseteq \mathbb{R}, P^1, \cdots, P^m \subseteq \mathbb{Q}[x_1, \cdots, x_i]$ and $s \in \mathbb{R}^{i-1}$ for $1 < i \leq n$. If $\{s\} \times \mathbb{R} \subseteq \bigcup_{j=1}^{m} \mathcal{C}(P^j, (s, s'_j))$ and for $1 \leq j \leq m, \mathcal{C}(P^j, (s, s'_j))$ is unsatisfiable for F, then $\mathcal{C}(proj_{cov}(P^1, \cdots, P^m, s, S), s)$ is unsatisfiable for ψ.*

*Corresponding author.

In practice, CAC excludes cells that cannot extend to a full assignment, guaranteeing finite-step termination by either exhausting the entire space or finding a full assignment.

Theorem 2. *Given an OMT branch formula $\psi \wedge t = x_t$, P denotes the set of polynomials in $\psi \wedge t = x_t$. If $\psi \wedge t = x_t$ is satisfiable, that is, there exists a complete assignment $s = (s_t, s_1, \cdots, s_n)$ that satisfies $\psi \wedge t = x_t$, then $\forall \gamma_o \in \mathcal{C}(proj_{dec}^n(P), s)_{x_t}, \psi \wedge t = \gamma_o$ is satisfiable.*

In the OCAC algorithm, once a full assignment is found, a satisfiable interval for x_t can be constructed based on Theorem 2. Within this interval, every value leads to a full assignment, indicating that the optimum lies on the boundary.

Theorem 3 (Termination of OCAC)**.** *Given an OMT branch formula $\psi \wedge t = x_t$, OCAC terminates.*

Proof. Give an *OMT branch formula* $\psi \wedge t = x_t$ with order $(x_t, x_1, \cdots, x_n)$. Let $P = \mathcal{P}(\psi \wedge t = x_t)$ contain all the polynomials in $\psi \wedge t = x_t, R = \{r_1, r_2, \cdots, r_k\}$ is the real root set of $proj_{dec}^n(P)$, and $\mathbb{L} = \{I_1, I_2, \cdots, I_{2k+1}\}$ with $I_1, I_2, \cdots, I_{2k+1}$ denoted as $(-\infty, r_1), [r_1, r_1], \cdots, (r_k, +\infty)$, respectively, and $\bigcup_{i=1}^{2k+1} I_i = \mathbb{R}$. Assume $x_t \mapsto s_t$ and $s_t \in I_i$.

If s_t can be extended to a full assignment $(s_t, s_1, \cdots s_n)$ that satisfies $\psi \wedge t = x_t$, then Solve_Internal($\psi \wedge t = x_t \wedge x_t = s_t$) terminates and returns a characterization interval I that $x_t \mapsto \gamma_o, \gamma_o \in I$ leads to a full assignment of $\psi \wedge t = x_t$ and we have $I_i = I$. OCAC prunes I_i.

If s_t cannot be extended to a satisfiable full assignment for $\psi \wedge t = x_t$, the operations of the algorithm replicate those of the original CAC algorithm. Its termination is guaranteed by the CAC algorithm. Solve_Internal($\psi \wedge t = x_t \wedge x_t = s_t$) returns a characterization interval I where $x_t \mapsto \gamma_o, \gamma_o \in I$ leads to violating some atoms of $\psi \wedge t = x_t$ and we have $I_i \subseteq I$. OCAC prunes I.

Therefore, each time Solve_Internal($\psi \wedge t = x_t \wedge x_t = s_t$) samples over x_t, it prunes the characterization interval and eliminates at least one element from $\mathbb{L}$. Given that $|\mathbb{L}| = 2k+1$, Solve_Internal($\psi \wedge t = x_t \wedge x_t = s_t$) concludes after at most $2k+1$ samplings of x_t, ensuring the termination of OCAC for the *OMT branch formula*. ☐

Theorem 4 (Correctness of OCAC)**.** *Given an OMT branch formula $\psi \wedge t = x_t$, if $\psi \wedge t = x_t$ is unsatisfiable, OCAC returns UNSAT; otherwise, OCAC can find the optimum.*

Proof. By Theorem 3, OCAC must be terminated with a set of intervals that satisfies $\mathbb{R} \subseteq \bigcup_{i=1}^{k_1} I_{unsat,i} \cup \bigcup_{j=1}^{k_2} I_{sat,j}$, where $I_{sat,j}$ is the satisfiable interval such that $\forall \gamma_o \in I_{sat,j}$, $F \wedge t = \gamma_o$ is satisfiable, i.e., $\exists s \in \mathbb{R}^n$, (γ_o, s) satisfies $F \wedge t = \gamma_o$; $I_{unsat,i}$ is the unsatisfiable interval such that $\forall \gamma_o \in I_{unsat,i}$, $F \wedge t = \gamma_o$ is unsatisfiable, i.e., $\forall s \in \mathbb{R}^n$, (γ_o, s) does not satisfy $F \wedge t = \gamma_o$. Obviously, from the definitions, $I_{unsat,i} \cap I_{sat,j} = \emptyset, 1 \leq i \leq k_1, 1 \leq j \leq k_2$. So, we can find the optimum on the lower bound of the leftmost satisfiable interval. If there is no satisfiable interval, $\psi \wedge t = x_t$ is unsatisfiable. ☐

Algorithm 1: OCAC

Input: $\psi \wedge t = x_t$: The *OMT branch formula*, with n variables.
Output: g, v, l: A flag that $\psi \wedge t = x_t$ is satisfiable; The optimum value; The cutting lemma.

$\mathbb{I} := \emptyset$, $g := \perp$, $v :=$ None
while $\bigcup_{I \in \mathbb{I}} I \neq \mathbb{R}$ **do**
  $s_t :=$ Sample_Objective_Value($\mathbb{I}$)
  $(T, O) :=$ Solve_Internal($\psi \wedge t = x_t \wedge x_t = s_t$)
  if $T = \top$ **then**
    $g := \top$, $v :=$ Analyze_Cell(O)
    $O := O \cup [s_t, +\infty)$
  end if
  $\mathbb{I} := \mathbb{I} \cup \{O\}$
end while
return $(g, v, \text{Lemma}(v))$

Algorithm Pseudocode

The missing algorithms for Algorithm 1 are: Sample_Objective_Value, Solve_Internal (Algorithm 2), Analyze_Cell, and Lemma. We omit Sample_Objective_Value, Analyze_Cell, and Lemma algorithms, because it is obvious. For most algorithms, we reuse the implementations of CVC5, with minor modifications. Algorithm 3 is a slightly changed version of "exists" algorithm and the original version can be found in (Kremer and Nalbach 2022). The *Real_Roots* algorithm processes a set of polynomials with a partial assignment, converting them into univariate polynomials, and then computes all real roots for each polynomial in this univariate set. It is implemented using the Lazard lifting scheme (Kremer and Brandt 2021) in CVC5. This implementation is essential for the application of the Lazard projection operator, which is complete in CAD (McCallum, Parusinski, and Paunescu 2019).

Before we present the pseudocode of the algorithms, we need to provide the definition of The *CAC Interval* (Ábrahám et al. 2021), which is defined as $I = \{l, u, P_L, P_U, P_i, P_\perp\}$, where

- l is the lower bound;
- u is the upper bound;
- P_L is a polynomial set that defined l, which is the subset of P_i such that $\forall p \in P_L, p(s \times \{l\}) = 0$;
- P_U is a polynomial set that defined u, which is the subset of P_i such that $\forall p \in P_U, p(s \times \{u\}) = 0$;
- P_i is a polynomial set with x_{i+1} as the main variable, serving as the characterization polynomial set that either satisfies or does not satisfy the *OMT branch formula*, depending on the satisfiability flag, g;
- $P_\perp$ is a polynomial set with main variable smaller than x_{i+1} that possess the same properties of P_i.

Algorithm 2: Solve_Internal

Input: The *OMT branch formula* and objective variable's sample value.
Output: ($\top$, O) or ($\bot$, O) where any $s_t \in I$ can or cannot be extended to a complete assignment.

(T, O) := Exists($\psi \wedge t = x_t, (s_t)$)
return (T, O)

Algorithm 3: Exists

Input: ψ: The OMT branch formula, with n variables.
Input: s: Current sampled partial assignment $(s_t, s_1, \ldots, s_{i-1})$ for $x_t, x_1, \cdots, x_{i-1}$.
Output: T: A flag that $F \wedge O$ is satisfiable.
Output: O: The characterization interval.

$\mathbb{I} := \emptyset$
while $\bigcup_{I \in \mathbb{I}} I \neq \mathbb{R}$ **do**
  s_i := Sample_Outside($\mathbb{I}$)
  if $\psi(s \times \{s_i\}) = \bot$ **then**
    (T, O) := ($\bot$, Get_Enclosing_Interval($\psi \wedge t = x_t$, $s \times \{s_i\}$))
  else if $\psi(s \times \{s_i\}) = \top$ **then**
    (T, O) := ($\top$, Get_Enclosing_Interval($\psi \wedge t = x_t$, $s \times \{s_i\}$))
  else if $i < n$ **then**
    (T, O) := Exists($\psi \wedge t = x_t$, $s \times \{s_i\}$)
  end if
  if $T = \top$ **then**
    return $(\top, O)$
  end if
  $\mathbb{I} := \mathbb{I} \cup \{O\}$
end while
O := Characterization_Interval($(s_t, \cdots, s_{i-2})$, s_{i-1}, $\mathbb{I}$, $\bot$)
return $(\bot, O)$

Additional Results

Table 1 shows the results of the comparison of different solvers. We include yicesQS (Bonacina, Graham-Lengrand, and Vauthier 2023) based on yices 2.6 (Dutertre 2014), CVC5 1.2.0 under GPL(Barbosa et al. 2022), and dReal v4.21.06.2 (Gao, Kong, and Clarke 2013). The issues are:

- CVC5 struggles to efficiently solve satisfiable FOL formulas, particularly when the optimum is RAN or RAN + ϵ, but it performs best on unsatisfiable instances.
- YicesQS has issues with getting models, making it impossible to distinguish results based on the type of optimum.
- dReal cannot solve all satisfiable instances under δ -SAT for the following reasons:
  - SMT-LIB format unsupported: For example, when using the `define-fun` command on Boolean variables, it reports: "what(): Variable `assumptions_` is of type BOOLEAN and it should not be used to construct a symbolic expression."
  - Type conversion: For example, "what(): dreal/util/math.cc:39 Fail to convert an `int64_t` value 2300000000000000000 to double."
  - Overflow: For example, "terminate called after throwing an instance of 'std::out_of_range' what(): `stol`."

Figure 1 - 5 are the detailed comparsion on different optimum types. We omit comparison with FOL(YicesQS) and dReal in the following figures.

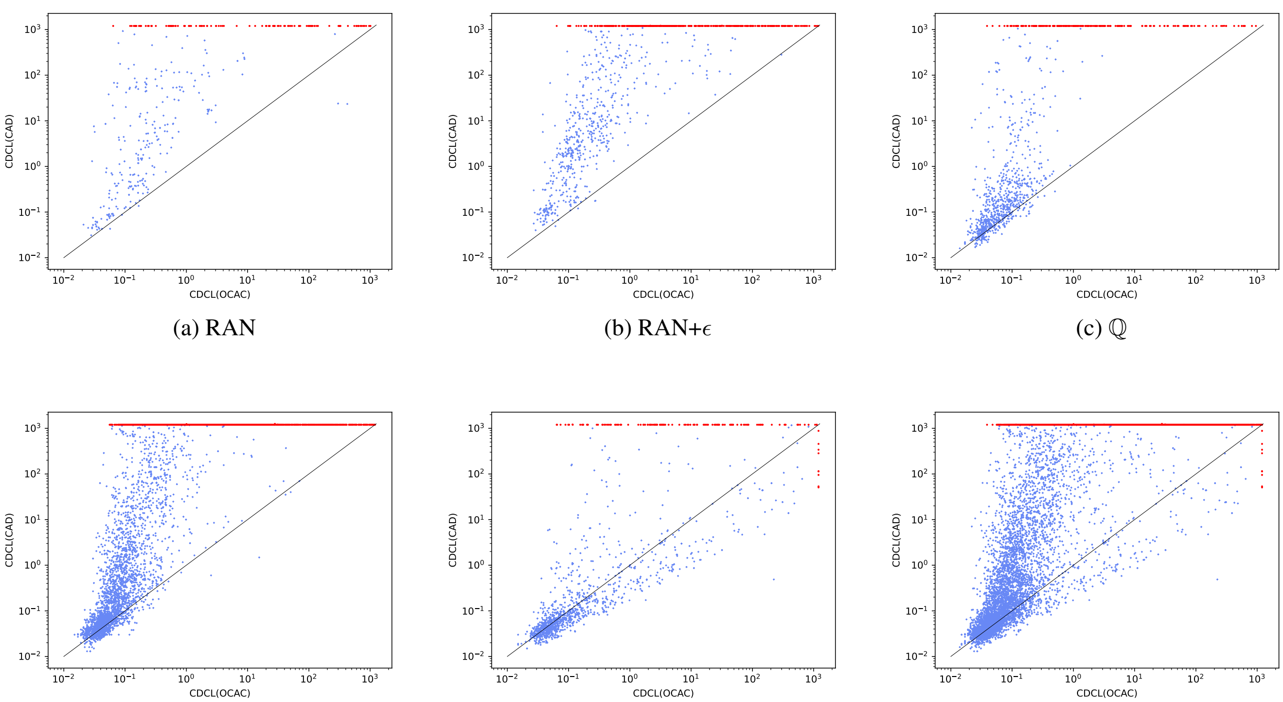
Algorithm 4: Get_Enclosing_Interval

Input: ψ: The OMT branch formula, with n variables.
Input: s: Current sampled partial assignment $(s_t, s_1, \ldots, s_{i-1})$ for $x_t, x_1, \cdots, x_{i-1}$ and $s_i \in \mathbb{R}$ such that $\psi[s \times \{s_i\}] \in \{False, True\}$.
Output: O: The characterization interval around s_i over s.

Denote by P the set of polynomials in ψ
Replace P by its irreducible factors
$P_\bot := \{p \in P | p \in \mathbb{Q}[x_1, \cdots, x_{i-1}]\}$
$P_i := P - P_\bot$
$Z := \{-\infty\} \cup Real\_Roots(P_i, s) \cup \{+\infty\}$
$l := \max\{z \in Z | z \leq s_i\}$
$u := \min\{z \in Z | z \geq s_i\}$
$P_L := \{p \in P_i | p((s \times \{l\})) = 0\}$
$P_U := \{p \in P_i | p((s \times \{u\})) = 0\}$
Define new interval O with $l, u, P_L, P_U, P_i, P_\bot$
return O

Table 1: Performance in terms of number of solved instances.

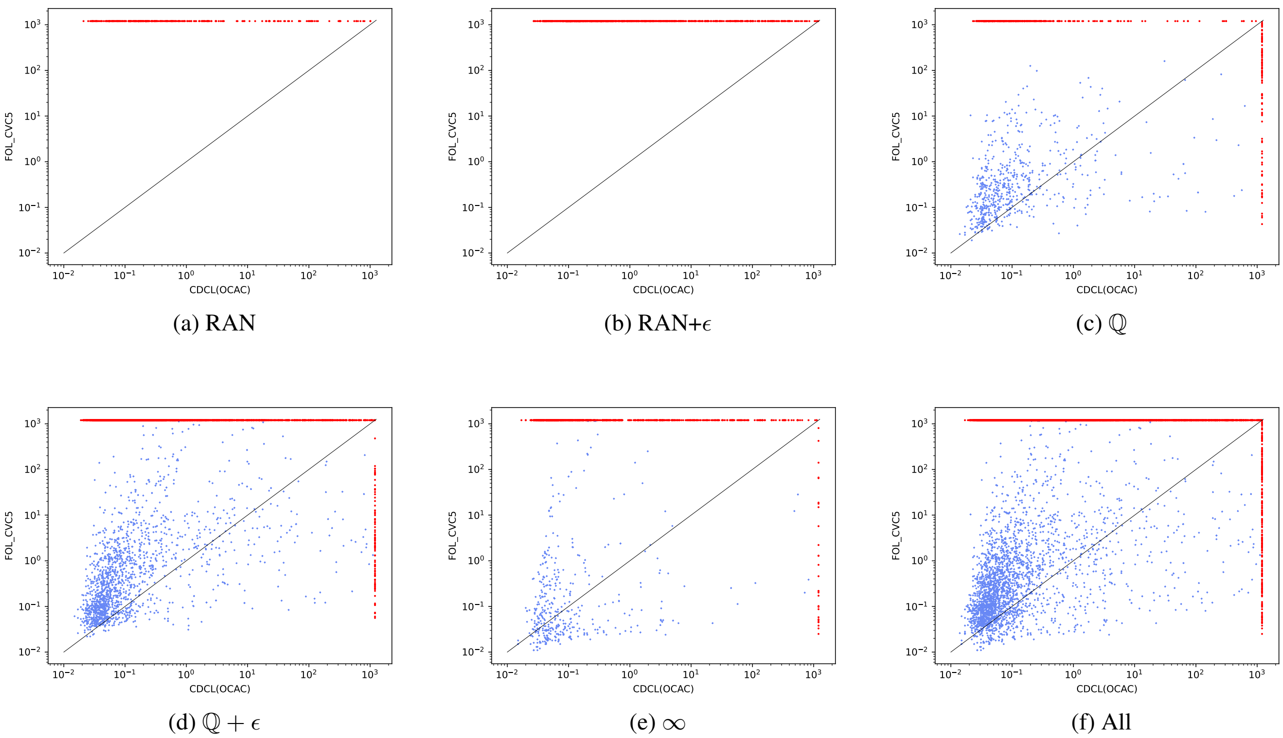
	#(RAN)	#(RAN $+\epsilon$)	#($\mathbb{Q}$)	#($\mathbb{Q}+\epsilon$)	#(∞)	#SAT	#UNSAT
CDCL(CAD)	246	551	802	2990	1129	5718	4568
FOL(CVC5)	0	0	722	1641	423	2786	**5103**
FOL(YicesQS)	0	0	0	0	0	6457	4933
FOL(Z3)	304	610	**1101**	3545	1165	6725	4392
OptiMathSAT(Bin)	0	0	943	1870	353	3166	5040
OptiMathSAT(Lin)	0	0	928	1819	336	3083	5040
dReal	0	0	0	0	0	0	1470
CDCL(OCAC) (Ours)	**369**	**981**	1084	**4248**	**1250**	**7932**	5019



(a) RAN (b) RAN+ϵ (c) $\mathbb{Q}$

(d) $\mathbb{Q}+\epsilon$ (e) ∞ (f) All

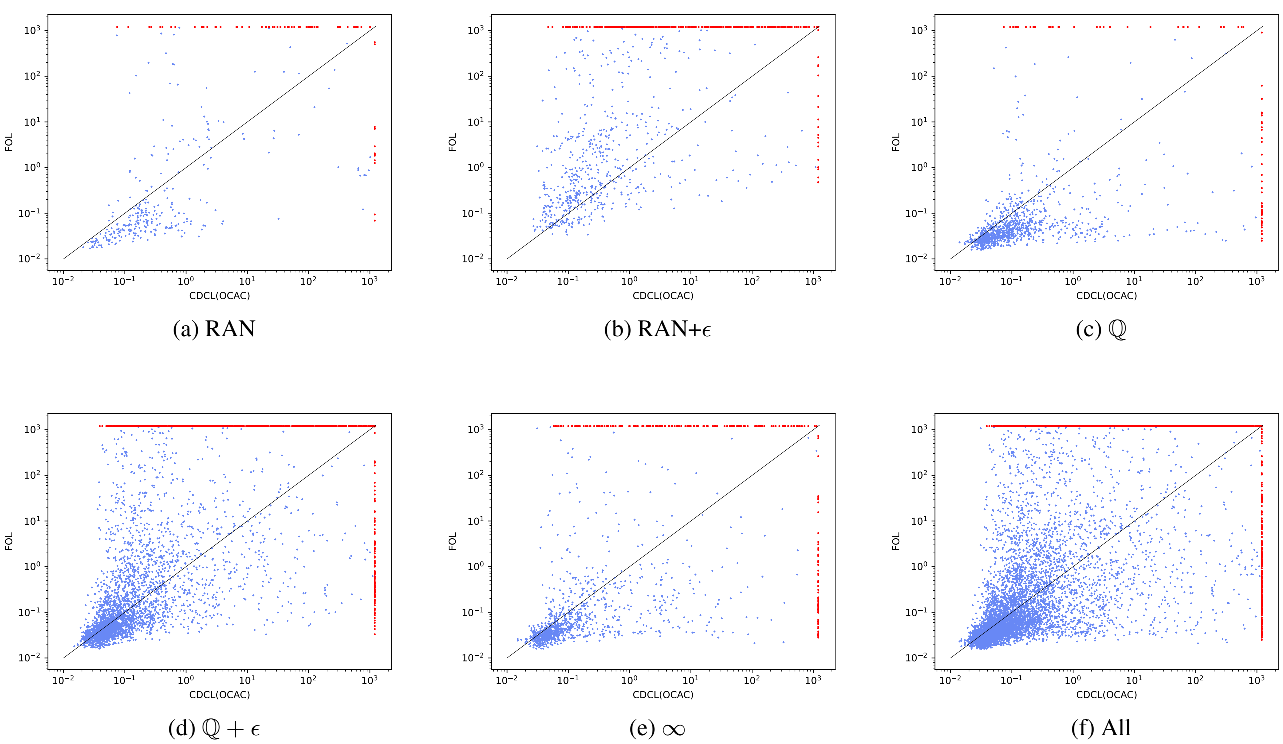
Figure 1: Detailed comparison of CDCL(OCAC) with CDCL(CAD).



(a) RAN (b) RAN+ϵ (c) $\mathbb{Q}$

(d) $\mathbb{Q} + \epsilon$ (e) ∞ (f) All

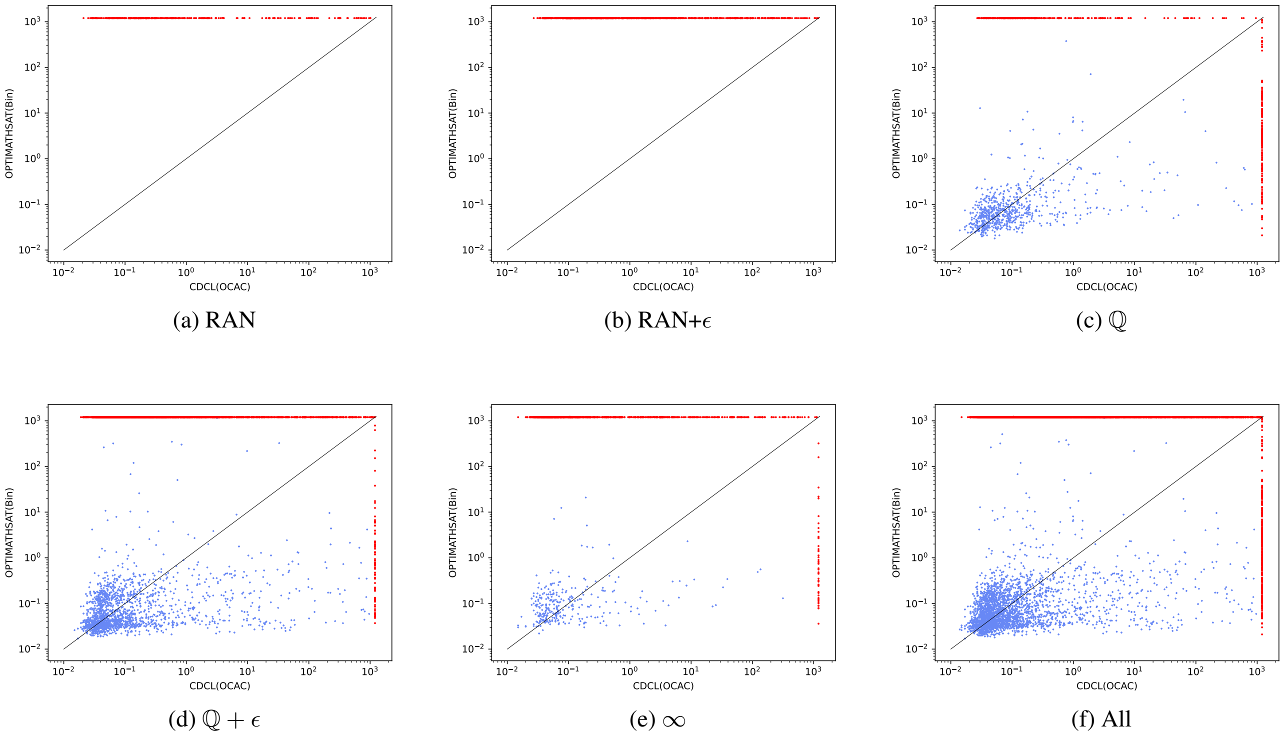
Figure 2: Detailed comparison of CDCL(OCAC) with FOL(CVC5).



(a) RAN (b) RAN+ϵ (c) $\mathbb{Q}$

(d) $\mathbb{Q} + \epsilon$ (e) ∞ (f) All

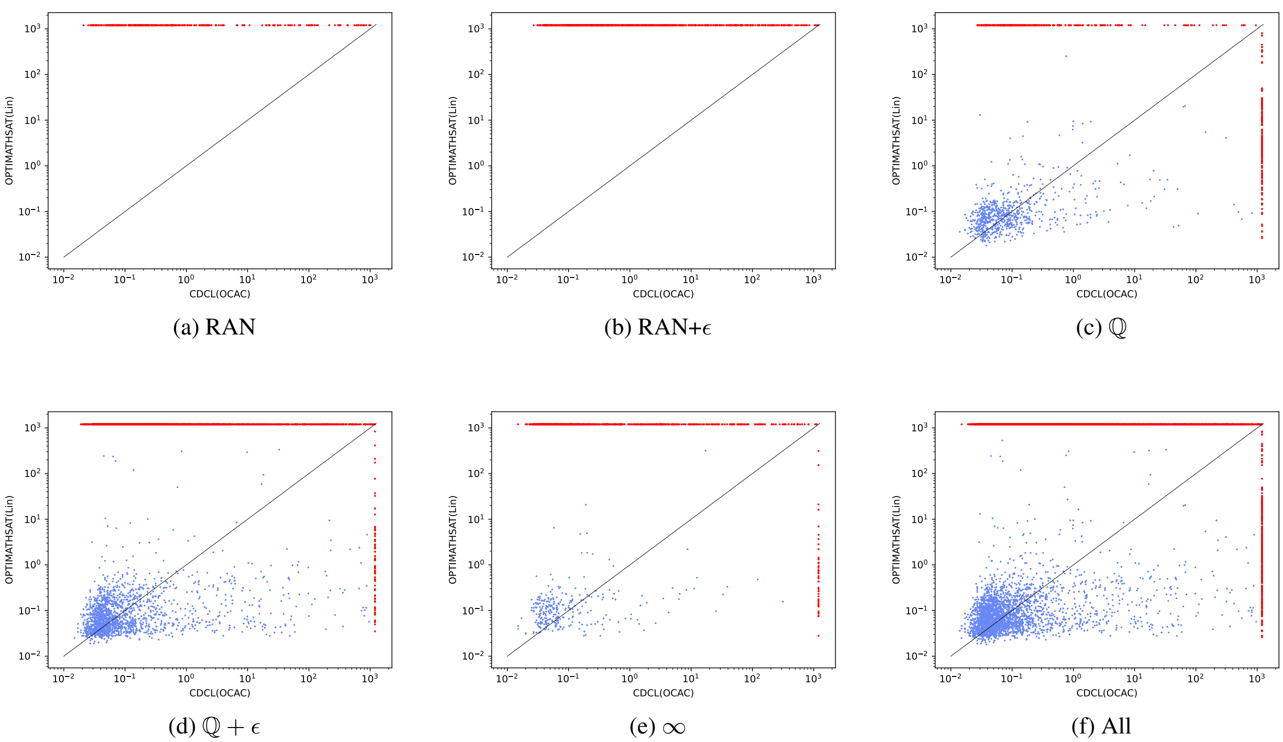
Figure 3: Detailed comparison of CDCL(OCAC) with FOL(Z3).



(a) RAN (b) RAN+ϵ (c) $\mathbb{Q}$

(d) $\mathbb{Q} + \epsilon$ (e) ∞ (f) All

Figure 4: Detailed comparison of CDCL(OCAC) with OptiMathSAT(Bin).



(a) RAN (b) RAN+ϵ (c) $\mathbb{Q}$

(d) $\mathbb{Q} + \epsilon$ (e) ∞ (f) All

Figure 5: Detailed comparison of CDCL(OCAC) with OptiMathSAT(Lin).

Algorithm 5: Characterization_Interval

Input: s: Current sampled partial assignment $(s_t, s_1, \ldots, s_{i-1})$ for $x_t, x_1, \cdots, x_{i-1}$.
Input: s_i: Current sampled value s_i for x_i.
Input: $\mathbb{I}$: A set of *CAC intervals* from $(s_t, s_1, \ldots, s_{i+1})$.
Input: f: A flag that the interval of x_t has a satisfiable sub-interval.
Global Input: Computed: A flag indicating whether CAD_Construction has been computed for the satisfiable assignment.
Output: O: A *CAC Interval*.

if f **then**
  if Computed **then**
    if $x_i = x_t$ **then**
      LOAD(Z)
    else
      Define new interval O with $l = -\infty, u = +\infty, P_L = \emptyset, P_U = \emptyset, P_i = \emptyset, P_\perp = \emptyset$
      return O
    end if
  else
    T := CAD_Construct_Characterization($\mathbb{I}$)
    if $x_i = x_t$ **then**
      $Z := \{-\infty\} \cup Real\_Roots(T, s) \cup \{+\infty\}$
      STORE(Z), Computed := $\top$
    else
      $P_\perp := \{p \in T | p \in \mathbb{Q}[x_1, \cdots, x_{i-1}]\}$.
      $P_i := T - P_\perp$
      Define new interval O with $l = -\infty, u = +\infty, P_L = \emptyset, P_U = \emptyset, P_i, P_\perp$
      return O
    end if
  end if
  $l := \max\{z \in Z | z \leq s_i\}$
  $u := \min\{z \in Z | z \geq s_i\}$
  Define new interval O with $l, u, P_L = \emptyset, P_U = \emptyset, P_i = T, P_\perp = \emptyset$
  return O
else
  T := CAC_Construct_Characterization($(s_t, s_1, \cdots, s_i)$, $\mathbb{I}$)
  $P_\perp := \{p \in T | p \in \mathbb{Q}[x_1, \cdots, x_{i-1}]\}$
  $P_i := T - P_\perp$
  $Z := \{-\infty\} \cup Real\_Roots(P_i, s) \cup \{+\infty\}$
  $l := \max\{z \in Z | z \leq s_i\}$
  $u := \min\{z \in Z | z \geq s_i\}$
  $P_L := \{p \in P_i | p((s \times \{l\})) = 0\}$
  $P_U := \{p \in P_i | p((s \times \{u\})) = 0\}$
  Define new interval O with $l, u, P_L, P_U, P_i, P_\perp$
  return O
end if

Algorithm 6: CAD_Construct_Characterization

Input: $\mathbb{I}$: A set of *CAC intervals* from $(s_t, s_1, \ldots, s_{i+1})$.
Output: T: A polynomial set provides information to characterize the region around s_t that is satisfiable.

$T := \emptyset$
Assert size($\mathbb{I}$) = 1
Define $l, u, P_{i+1}, P_\perp$ from the only *CAC Interval* I of $\mathbb{I}$
Assert $l = -\infty$ and $u = +\infty$
$T := T \cup P_\perp$
$T := T \cup \{lcoeff(p) | p \in P_{i+1}\}$
$T := T \cup \{tloeff(p) | p \in P_{i+1}\}$
$T := T \cup \{disc(p, x_{i+1}) | p \in P_{i+1}\}$
$T := T \cup \{res(p, q, x_{i+1}) | p, q \in P_{i+1}, p \neq q\}$
return T

Algorithm 7: CAC_Construct_Characterization

Input: s: Current sampled partial assignment $(s_t, s_1, \ldots, s_i)$ for $x_t, x_1, \cdots, x_i$.
Input: $\mathbb{I}$: A set of *CAC intervals* from $(s_t, s_1, \ldots, s_{i+1})$.
Output: T: A polynomial set that characterizes a region around s that is already unsatisfiable for the same reasons.

Sort the set of *CAC Intervals* and remove the redundant ones
$T := \emptyset$
for $I \in \mathbb{I}$ **do**
  Define $l, u, P_L, P_U, P_{i+1}, P_\perp$ from the *CAC Interval* I.
  $T := T \cup P_\perp$
  $T := T \cup \{lcoeff(p) | p \in P_{i+1}\}$
  $T := T \cup \{tloeff(p) | p \in P_{i+1}\}$
  $T := T \cup \{disc(p, x_{i+1}) | p \in P_{i+1}\}$
  $T := T \cup \{res(p, q, x_{i+1}) | p \in P_L, q \in P_{i+1}, \exists \alpha \leq l, s.t., q((s, \alpha)) = 0\}$
  $T := T \cup \{res(p, q, x_{i+1}) | p \in P_U, q \in P_{i+1}, \exists \alpha \geq u, s.t., q((s, \alpha)) = 0\}$
end for
for all $j \in \{1, \cdots, |\mathbb{I}| - 1\}$ **do**
  Define P_U from j-th *CAC Interval* and P_L from (j+1)-th *CAC Interval*
  $T := T \cup \{res(p, q, x_{i+1}) | p \in P_U, q \in P_L\}$
end for
return T

References

Ábrahám, E.; Davenport, J. H.; England, M.; and Kremer, G. 2021. Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings. *J. Log. Algebraic Methods Program.*, 119: 100633.

Bär, P.; Nalbach, J.; Ábrahám, E.; and Brown, C. W. 2023. Exploiting Strict Constraints in the Cylindrical Algebraic Covering. In Graham-Lengrand, S.; and Preiner, M., eds., *Proceedings of the 21st International Workshop on Satisfiability Modulo Theories (SMT 2023) co-located with the 29th International Conference on Automated Deduction (CADE 2023), Rome, Italy, July, 5-6, 2023*, volume 3429 of *CEUR Workshop Proceedings*, 33–45. CEUR-WS.org.

Barbosa, H.; Barrett, C.; Brain, M.; Kremer, G.; Lachnitt, H.; Mann, M.; Mohamed, A.; Mohamed, M.; Niemetz, A.; Nötzli, A.; et al. 2022. cvc5: A versatile and industrial-strength SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 415–442. Springer.

Bonacina, M. P.; Graham-Lengrand, S.; and Vauthier, C. 2023. QSMA: A New Algorithm for Quantified Satisfiability Modulo Theory and Assignment. In Pientka, B.; and Tinelli, C., eds., *Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings*, volume 14132 of *Lecture Notes in Computer Science*, 78–95. Springer.

Dutertre, B. 2014. Yices 2.2. In Biere, A.; and Bloem, R., eds., *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, 737–744. Springer.

Gao, S.; Kong, S.; and Clarke, E. M. 2013. dReal: An SMT Solver for Nonlinear Theories over the Reals. In Bonacina, M. P., ed., *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings*, volume 7898 of *Lecture Notes in Computer Science*, 208–214. Springer.

Kremer, G.; and Brandt, J. 2021. Implementing arithmetic over algebraic numbers A tutorial for Lazard's lifting scheme in CAD. In Schneider, C.; Marin, M.; Negru, V.; and Zaharie, D., eds., *23rd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2021, Timisoara, Romania, December 7-10, 2021*, 4–10. IEEE.

Kremer, G.; and Nalbach, J. 2022. Cylindrical Algebraic Coverings for Quantifiers (short paper). In Uncu, A. K.; and Barbosa, H., eds., *Proceedings of the 7th SC-Square Workshop co-located with the Federated Logic Conference, SC-Square@FLoC 2022, as a part of the 11th International Joint Conference on Automated Reasoning, IJCAR 2022, Haifa, Israel, August 12, 2022*, volume 3458 of *CEUR Workshop Proceedings*, 1–9. CEUR-WS.org.

McCallum, S.; Parusinski, A.; and Paunescu, L. 2019. Validity proof of Lazard's method for CAD construction. *J. Symb. Comput.*, 92: 52–69.