Equality handling and efficiency improvement of SMT for non-linear constraints over reals

By VU XUAN TUNG

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Mizuhito Ogawa

March, 2015

Equality handling and efficiency improvement of SMT for non-linear constraints over reals

By VU XUAN TUNG (1310007)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Mizuhito Ogawa

and approved by
Associate Professor Nao Hirokawa
Professor Takuya Katayama
Associate Professor Katsuhiko Gondow

February, 2015 (Submitted)

Abstract

This thesis presents strategies for efficiency improvement and extensions of an SMT solver **raSAT** for polynomial constraints.

raSAT which initially focuses on polynomial inequalities over real numbers follows ICP methodology and adds testing to boost satisfiability detection [10]. In this work, in order to deal with exponential exploration of boxes, several heuristic measures, called *SAT likelyhood*, *sensitivity*, and the number of unsolved atomic polynomial constraints, are proposed. Extensions for handling equations and handling constraints over integer number are also presented.

Contents

Chapter 1

Introduction

1.1 Polynomial Constraint Solving

Polynomial constraint solving over real numbers aims at computing an assignment of real values to variables that satisfies given polynomial inequalities/equations. If such an assignment exists, the constraint is said to be satisfiable (SAT) and the assignment is called SAT instance; otherwise we mention it as unsatisfiable (UNSAT).

Example 1.1.1. *The formula $x^2 + y^2 < 1 \wedge xy > 1$ is an example of an unsatisfiable one. While the set of satisfiable points for the first inequality ($x^2 + y^2 < 1$) is the read circle in Figure 1.1, the set for the second one is the blue area. Because these two areas do not intersect, the conjunction of two equalities is UNSAT.*

Example 1.1.2. *Figure 1.2 illustrates the satisfiability of the constraint: $x^2+y^2 < 4 \wedge xy > 1$. Any point in the purple area is a SAT instance of the constraint, e.g.* $(1.5, 1)$.

Solving polynomial constraints has many application in Software Verification, such as

- **Locating roundoff and overflow errors**, which is our motivation [14, 15]
- **Automatic termination proving**, which reduces termination detection to finding a suitable ordering [12], e.g., $\mathsf{T_TT_2}$[1], AProVE[2].
- **Loop invariant generation**. Farkas's lemma is a popular approach in linear loop invariant generation [3], and is reduced to degree 2 polynomials. Non-linear loop invariant [19] requires more complex polynomials.
- **Hybrid system**. Solving polynomial constraints over real numbers is often used as backend engines [18].
- **Mechanical contrnol design**. PID control is simple but widely used, and designing parameters is reduced to polynomial constraints [1].

[1] `http://cl-informatik.uibk.ac.at/software/ttt2/`
[2] `http://aprove.informatik.rwth-aachen.de`

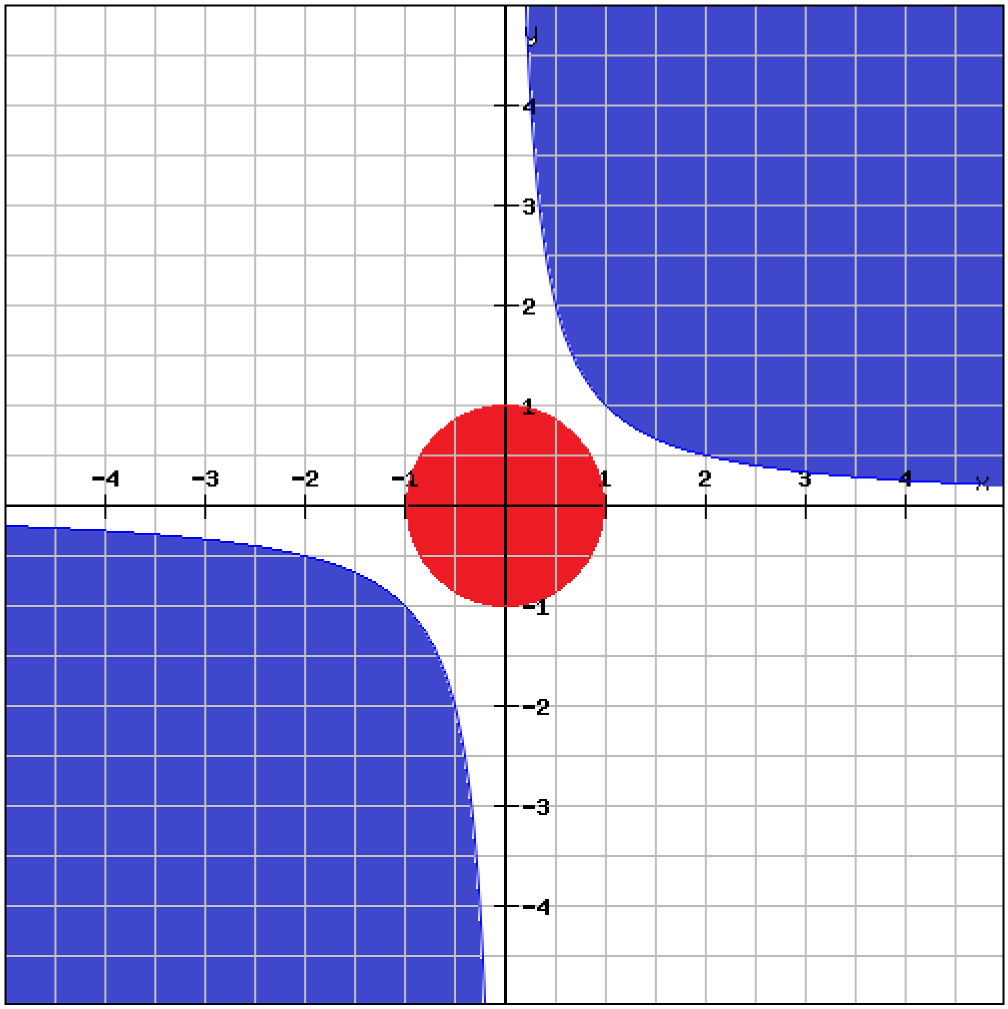


Figure 1.1: Example of UNSAT constraint

1.2 Existing Approaches

Although solving polynomial constraints on real numbers is decidable [21], current methodologies have their own pros and cos. They can be classified into the following categories:

1. **Quantifier elimination by cylindrical algebraic decomposition (QE-CAD)** [2] is a complete technique, and is implemented in Mathematica, Maple/SynRac, Reduce/Redlog, QEPCAD-B, and recently in Z3 4.3 (which is referred as nlsat in [9]). Although QE-CAD is precise and detects beyond SAT instances (e.g., SAT regions), scalability is still challenging, since its complexity is doubly-exponential with respect to the number of variables.

2. **Virtual substitution** eliminates an existential quantifier by substituting the corresponding quantified variable with a very small value ($-\infty$), and either each root (with respect to that variable) of polynomials appearing in the constraint or each

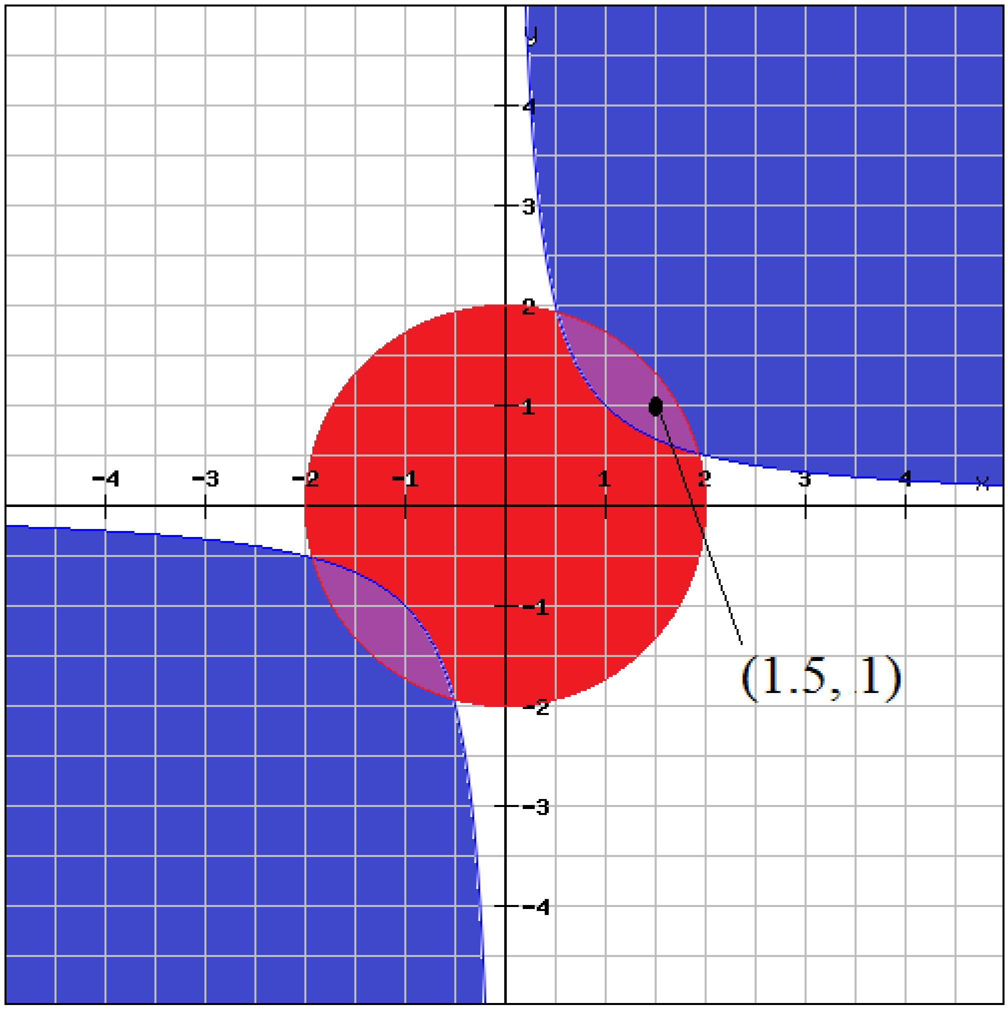


Figure 1.2: Example of SAT constraint

root plus an infinitesimal ϵ. Disjunction of constraints after substitutions is equivalent to the original constraint. Because VS needs the formula for roots of polynomials, its application is restricted to polynomials of degree up to 4. SMT-RAT and Z3 [16] applies VS.

3. **Bit-blasting**. In this category of methodology, numerical variables are represented by a sequence of binary variables. The given constraint is converted into another constraint over the boolean variables. SAT solver is then used to find a satisfiable instance of binary variables which can be used to calculate the values of numerical variables. MiniSmt [22], the winner of QF_NRA in SMT competition 2010, applies it for (ir)-rational numbers. It can show SAT quickly, but due to the bounded bit encoding, it cannot conclude UNSAT. In addition, high degree of polynomial results in large SAT formula which is an obstacle of bit-blasting.

4. **Linearization**. CORD [6] uses COrdinate Rotation DIgital Computer (CORDIC)

for real numbers to linearizes multiplications into a sequence of linear constraints. Each time one multiplication is linearized, a number of new constraints and new variables are introduced. As a consequence, high degree polynomials in the original constraint lead to large number of linear constraints.

5. **Interval Constraint Propagation (ICP)** which are used in SMT solver community, e.g., iSAT3 [5], dReal [7], and RSOLVER [17]. ICP combines over-approximation by interval arithmetics and constraint propagation to prune out the set of unsatisfiable points. When pruning does not work, decomposition (branching) on intervals is applied. ICP which is capable of solving "multiple thousand arithmetic constraints over some thousands of variables" [5] is practically often more efficient than algebraic computation.

1.3 Proposed Approach and Contributions

Our aim is an SMT solver for solving polynomial constraint. We first focus on strict inequalities because of the following reasons.

1. Satisfiable inequalities allow over-approximation. An over-approximation estimates the range of a polynomial f as $range^O(f)$ that covers all the possible values of f, i.e. $range(f) \subseteq range^O(f)$. For an inequality $f > 0$, if $range^O(f)$ stays in the positive side, it can be concluded as SAT. On the other hand, over-approximation cannot prove the satisfiability of SAT equations.

2. Satisfiable inequalities allow under-approximation. An under-approximation computes the range of the polynomial f as $range^U(f)$ such that $range(f) \supseteq range^U(f)$. If $range^U(f)$ is on the positive side, $f > 0$ can be said to be SAT. Due to the continuity of f, finding such an under-approximation for solving $f > 0$ is more feasible than that for $f = 0$.

   – If $f(\bar{x}) > 0$ has a real solution $\bar{x_0}$, there exist rational points near $\bar{x_0}$ which also satisfy the inequality. Solving inequalities over real numbers thus can be reduced to that over rational numbers.
   – The real solution of $f(\bar{x}) = 0$ cannot be approximated to any rational number.

For UNSAT constraint (both inequalities and equations) can be solved by over-approximation. Suppose $range^O(f)$ is the result of an over-approximation for a polynomial f.

1. If $range^O(f)$ resides on the negative side, $f > 0$ is UNSAT.

2. If $range^O(f)$ stays on either negative or positive side, $f = 0$ is UNSAT.

Our approach of "iterative approximation refinement" - **raSAT** loop for solving polynomial constraint was proposed and implemented as an SMT solver named raSAT in [10]. This work improves the efficiency of the tool and extend it to handle equations. The summary of the proposed method in [10] is:

1. Over-approximation is used for both disproving and proving polynomial inequalities. In addition, under-approximation is used for boosting SAT detection. When both of these methods cannot conclude the satisfiability, the input formula is refined so that the result of approximation become more precise.

2. Interval Arithmetic (IA) and Testing are instantiated as an over-approximation and an under-approximation respectively. While IA defines the computations over the intervals, e.g. $[1,3] +_{IA} [3,6] = [2,9]$, Testing attempts to propose a number of assignments from variables to real numbers and check each assignment against the given constraint to find a SAT instance.

3. In refinement phase, intervals of variables are decomposed into smaller ones. For example, $x \in [0,10]$ becomes $x \in [0,4] \vee x \in [4,10]$.

4. Khanh and Ogawa [10] also proposed a method for detecting satisfiability of equations using the Intermediate Value Theorem.

The contributions of this work are as follows:

1. Although the method of using IA is robust for large degrees of polynomial, the number of boxes (products of intervals) grows exponentially with respect to the number of variables during refinement (interval decomposition). As a result, strategies for selecting variables to decomposed and boxes to explore play a crucial role in efficiency. We introduce the following strategies:

   – A box with more possiblity to be SAT is selected to explore, which is estimated by several heuristic measures, called *SAT likelyhood*, and the number of unsolved atomic polynomial constraints.

   – A more influential variable is selected for multiple test cases and decomposition. This is estimated by *sensitivity* which is determined during the computation of IA.

2. Two schemes of incremental search are proposed for enhancing solving process:

   – **Incremental deepening**. raSAT follows the depth-first-search manner. In order to escape local optimums, it starts searching with a threshold that each interval will be decomposed no smaller than it. If neither SAT nor UNSAT is detected, a smaller threshold is taken and raSAT restarts.

   – **Incremental widening**. Starting with a small interval, if raSAT detects UNSAT, input intervals are enlarged and raSAT restarts. For SAT constraint, small (finite) interval allows sensitivity to be computed because Affine Interval [10] requires finite range of variables. As a consequence, our above strategies will take effects on finding SAT instance. For the UNSAT case, combination of small intervals and incremental deepening helps raSAT quickly determines the threshold in which unsatisfiability may be proved by IA.

3. SAT confirmation step by an error-bound guaranteed floating point package **iR-RAM**[3], to avoid soundess bugs caused by roundoff errors.

4. This work also implement the idea of using Intermediate Value Theorem to show the satisfiability of equations which was suggested in [10].

5. We also extend raSAT to handle constraint over integer numbers. For this extension, we only generate the integer values for variables in testing phase. In addition, the threshold used for stopping decomposition is set to 1.

1.4 Thesis Outline

Coming soon...

[3] `http://irram.uni-trier.de`

Chapter 2

Preliminaries

2.1 Abstract DPLL

2.1.1 Syntax

Definition 2.1.1. *The signature of propositional logic has an alphabet consisting of*

1. *proposition symbols:* $p_0, p_1, \cdots,$
2. *logical connective:* $\wedge, \vee, \rightarrow, \leftrightarrow, \neg, \bot$.

Definition 2.1.2. *The set PROP of propositions is the set that satisfies the following properties*

1. $\forall i \in \mathbb{N}\ p_i \in PROP$,
2. $\bot \in PROP$,
3. $\varphi, \psi \in PROP \implies \varphi \circ \psi \in PROP$ *where* $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$,
4. $\varphi \in PROP \implies \neg\varphi \in PROP$.

2.1.2 Semantics

Definition 2.1.3. *A model* M *is a map* $M : \{p_i | i \in \mathbb{N}\} \mapsto \{0, 1\}$

Definition 2.1.4. *Given a model* M *and a proposition* $\varphi \in PROP$*, the valuation of* φ *which is denoted by* φ^M *is defined recursively as:*

1. *If* $\varphi = p_i$ *for some* i*, then* $\varphi^M = M(p_i)$.
2. *If* $\varphi = \bot$*, then* $\varphi^M = 0$.
3. *If* $\varphi = \varphi_1 \wedge \varphi_2$*, then* $\varphi^M = min(\varphi_1^M, \varphi_2^M)$.
4. *If* $\varphi = \varphi_1 \vee \varphi_2$*, then* $\varphi^M = max(\varphi_1^M, \varphi_2^M)$.

5. *If* $\varphi = \varphi_1 \to \varphi_2$*, then* $\varphi^M = max(1 - \varphi_1^M, \varphi_2^M)$.

6. *If* $\varphi = \varphi_1 \leftrightarrow \varphi_2$*, then* $\varphi^M = \begin{cases} 1 & \text{if } \varphi_1^M = \varphi_2^M \\ 0 & \text{otherwise} \end{cases}$

7. *If* $\varphi = \neg\varphi'$*, then* $\varphi^M = 1 - \varphi'^M$.

2.2 Satisfiability Modulo Theories - SMT

2.2.1 Syntax

Definition 2.2.1. *A signature* Σ *is a 4-tuple* (S, P, F, V, α) *containing a set* S *of sorts, a set* P *of predicate symbols, a set* F *of function symbols, a set of variables, and a map* α *which associates symbols to their sorts.*

– $\forall p \in P; \alpha(p)$ *is a* n*-tuple argument sorts of* p.

– $\forall f \in F; \alpha(f)$ *is a* n*-tuple of argument and returned sorts of* f.

– $\forall v \in V;\ \alpha(v)$ *represents the sort of variable* v.

Followings are the definitions of terms and formulas.

Definition 2.2.2. *The set* $TERM$ *of terms is the set that satisfies the properties*

1. $v \in V \Rightarrow v \in TERM$

2. $t_1, \cdots, t_n \in TERM;\ f \in F \Rightarrow f(t_1, \cdots, t_n) \in TERM$

Definition 2.2.3. *The set* $FORM$ *of formulas is the set that satisfies the properties:*

1. $\perp \in FORM$

2. $t_1, \cdots, t_n \in TERM;\ p \in P \Rightarrow p(t_1, \cdots, t_n) \in FORM$

3. $\varphi, \psi \in FORM \Rightarrow (\varphi \circ \psi) \in FORM$ *where* $\circ \in \{\wedge, \vee, \to, \leftrightarrow\}$

4. $\varphi \in FORM \Rightarrow \neg\varphi \in FORM$

5. $\varphi \in FORM; v \in V \Rightarrow (\forall v)\varphi, (\exists v)\varphi \in FORM$

2.2.2 Semantics

Definition 2.2.4. *Let Σ is a signature. A model M of Σ is a pair (U, I) in which U is the universe and I is the interpretation of symbols.*

- $\forall s \in S$; $I(s) \subseteq U$ *specifies the possible values of sort* s.
- $\forall f \in F$; $I(f) = \{(t_1, \cdots, t_n) | t_1 \in I(s_1), \cdots, t_n \in I(s_{n-1})\} \mapsto I(s_n)$ *with* $, \alpha(f) = (s_1, \cdots, s_n)$
- $\forall p \in P$; $I(p) = \{(t_1, \cdots, t_n) | t_1 \in I(s_1), \cdots, t_n \in I(s_n)\} \mapsto \{0, 1\}$ *with* $\alpha(p) = (s_1, \cdots, s_n)$
- $\forall v \in V$; $I(v) \in I(\alpha(v))$

The interpretation of one predicate symbol is allowed to be not total and the symbol $\mathring{u}$ (unknown) is used to indicate the result of undefined operations. For further convenience, we also define the following relations: $\mathring{u} < 0, 1$ and $\mathring{u} > 0, 1$ and the following arithmetic $1 - \mathring{u} = \mathring{u}$ which are useful when we evaluate the values of logical connectives, i.e. $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, or $\neg$.

Definition 2.2.5. *Let Σ is a signature. A Σ-theory T is a (infinite) set of Σ-models.*

A theory T' is a subset of theory T iff $T' \subseteq T$.

Definition 2.2.6. *Let $\Sigma = (S, P, F, \alpha)$, t, φ and $M = (U, I)$ are a signature, a $\Sigma-$term, a $\Sigma-$formula and a $\Sigma-$model respectively. The valuations of t against M which is denoted by t^M is defined recursively as:*

1. *If $t = v \in V$, then $t^M = I(v)$.*
2. *If $t = f(t_1, \cdots, t_n)$, then* $t^M = \begin{cases} I(f)(t_1^M, \cdots, t_n^M) & \text{if } (t_1^M, \cdots, t_n^M) \in Dom(I(f)) \\ \mathring{u} & \text{otherwise} \end{cases}$
   for $f \in F$ and $t_1, \cdots, t_n \in TERM$

Similarly, the valuation φ^M of φ is defined as:

1. *If $\varphi = p(t_1, \cdots, t_n)$, then* $\varphi^M = \begin{cases} I(p)(t_1^M, \cdots, t_n^M) & \text{if } (t_1^M, \cdots, t_n^M) \in Dom(I(p)) \\ \mathring{u} & \text{otherwise} \end{cases}$
   for $p \in P$ and $t_1, \cdots, t_n \in TERM$
2. *If $\varphi = \perp, \neg\varphi', \varphi_1 \circ \varphi_2$ for $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ and $\varphi', \varphi_1, \varphi_2 \in FORM$, then φ^M is defined similarly as in Definition 2.1.4.*
3. *If $\varphi = \forall v \varphi'$, then $\varphi^M = \min_{v_i \in I(\alpha(v))} \varphi'^{(U, I < v \mapsto v_i >)}$ for $v \in V$, $\varphi' \in FORM$ and $I < v \mapsto v_i >$ denotes the map after updating the map from v to v_i into I.*
4. *If $\varphi = \exists v \varphi'$, then $\varphi^M = \max_{v_i \in I(\alpha(v))} \varphi'^{(U, I < v \mapsto v_i >)}$ for $v \in V$ and $\varphi' \in FORM$.*

We say that M satisfies φ which is denoted by $\models_M \varphi$ iff $\varphi^M = 1$. If $\varphi^M = 0$, $\not\models_M \varphi$ is used to denote that M does not satisfy φ.

Lemma 2.2.1. *Given any Σ-model M and Σ-formula φ, we have $\models_M \varphi \iff \not\models_M \neg\varphi$*

Proof. $\models_M \varphi \iff \varphi^M = 1 \iff 1 - \varphi^M = 0 \iff (\neg\varphi)^M = 0 \iff \not\models^M \neg\varphi$ □

Definition 2.2.7. *Let T be a Σ-theory. A Σ-formula φ is:*

- *satisfiable in T or T-SAT iff $\exists M \in T; \models_M \varphi$*
- *valid in T or T-VALID iff $\forall M \in T; \models_M \varphi$*
- *unsatisfiable in T or T-UNSAT iff $\forall M \in T; \not\models_M \varphi$*
- *unknown in T or T-UNKNOWN iff $\forall M \in T; \varphi^M = \mathring{u}$*

Lemma 2.2.2. *If T be a Σ-theory, then φ is T-VALID $\iff$ $\neg\varphi$ is T-UNSAT*

Proof. φ is T-VALID $\iff$ $\forall M \in T;\ \models_M \varphi \iff \forall M \in T;\ \not\models_M \neg\varphi$ (Lemma 2.2.1) $\iff$ $\neg\varphi$ is T-UNSAT. □

Lemma 2.2.3. *If $T' \subseteq T$, then φ is T'-SAT $\implies$ φ is T-SAT.*

Proof. φ is T'-SAT $\implies \exists M \in T';\ \models_M \varphi \implies \exists M \in T;\ \models_M \varphi$ (because $T' \subseteq T$) $\implies$ φ is T-SAT. □

2.3 Polynomial Constraints over Real Numbers

2.3.1 Syntax

We instantiate the signature $\Sigma^p = (S^p, P^p, F^p, V^p, \alpha^p)$ in Section 2.2.1 for polynomial constraints as following:

1. $S^p = \{Real\}$
2. $P^p = \{\succ, \prec, \succeq, \preceq, \approx, \not\approx\}$
3. $F^p = \{\oplus, \ominus, \otimes, \mathbf{1}\}$
4. $\forall p \in P^p$; $\alpha^p(p) = (Real, Real)$
5. $\forall f \in F^p \setminus \{\mathbf{1}\}$; $\alpha^p(f) = (Real, Real, Real)$ and $\alpha^p(\mathbf{1}) = Real$
6. $\forall v \in V;\ \alpha^P(v) = Real$

A polynomial and a polynomial constraint are a Σ^p-term and a Σ^p-formula respectively. However, for simplicity, we currently focus on small portion of Σ^p-formulas

2.3.2 Semantics

A model $M^p_{\mathbb{R}} = (\mathbb{R}, I^p_{\mathbb{R}})$ over real numbers contains the set of reals number $\mathbb{R}$ and a map I that satisfies the following properties.

1. $I^p_{\mathbb{R}}(Real) = \mathbb{R}$.

2. $\forall p \in P^p;\ I_{\mathbb{R}}(p) = \mathbb{R} \times \mathbb{R} \mapsto \{1, 0\}$ such that $I^p_{\mathbb{R}}(p)(r_1, r_2) = \begin{cases} 1 & \text{if } r1\ p_{\mathbb{R}}\ r2 \\ 0 & \text{otherwise} \end{cases}$ where $(\succ_{\mathbb{R}}, \prec_{\mathbb{R}}, \succeq_{\mathbb{R}}, \preceq_{\mathbb{R}}, \approx_{\mathbb{R}}, \not\approx_{\mathbb{R}}) = (>, <, \geq, \leq, =, \neq)$.

3. $\forall f \in F^p \setminus \{\mathbf{1}\};\ I^p_{\mathbb{R}}(f) = \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ such that $I^p_{\mathbb{R}}(f)(r_1, r_2) = r_1\ f_{\mathbb{R}}\ r_2$ where $(\oplus_{\mathbb{R}}, \ominus_{\mathbb{R}}, \otimes_{\mathbb{R}}) = (+, -, *)$.

4. $I^p_{\mathbb{R}}(\mathbf{1}) = 1$

5. $\forall v \in V;\ I^p_{\mathbb{R}}(v) \in \mathbb{R}$.

The theory of real numbers is $T^p_{\mathbb{R}} = \{M^p_{\mathbb{R}} | M^p_{\mathbb{R}}$ is a model of real numbers $\}$. By this instantiation, each model differs to another by the mapping from variables to real numbers. As a result, an assignment of real numbers to variables can be used to represent a model $M^p_{\mathbb{R}}$; i.e. $\{v \mapsto r \in \mathbb{R} | v \in V\}$ represents a model. Given a map $\theta = \{v \mapsto r \in \mathbb{R} | v \in V\}$, $\theta^p_{\mathbb{R}}$ denotes the model represented by θ.

Representing (sub-)theory of real numbers as a constraint of intervals

The signature of the first order logic is instantiated as $\Sigma^I = (S^I, P^I, F^I, \alpha^I)$ for interval constraints:

1. $S^I = \{Real, Interval\}$

2. $P^I = \{\in\}$

3. $F^I = \{c | c \text{ is a constant}\}$

4. $\alpha^I(\in) = (Real, Interval)$ and $\forall c \in F^I;\ \alpha^I(c) = Interval$

5. $\forall c \in F^I;\ \alpha^I(c) = Interval$.

6. $\forall v \in V;\ \alpha^I(v) = Real$.

We call Σ^I-formula is an interval constraint. A model $M^I_{\mathbb{R}} = (\mathbb{I} \cup \mathbb{R}, I^I)$ over intervals contains the set of real numbers and real intervals $\mathbb{I} \cup \mathbb{R}$ and a map I^I that satisfies the following properties.

1. $I^I(Real) = \mathbb{R}$ and $I^I(Interval) = \mathbb{I}$. $\mathbb{I}$ is the set of all real intervals which is defined later in Definition 3.2.2.

2. $I^I(\in) = \mathbb{R} \times \mathbb{I} \mapsto \{1, 0\}$ such that $I^p_{\mathbb{R}}(p)(r, \langle a, b \rangle) = \begin{cases} 1 & \text{if } a \leq r \leq b \\ 0 & \text{otherwise} \end{cases}$.

3. $\forall c \in F^I$; $I^I(c) \in \mathbb{I}$.

4. $\forall v \in V$; $I^I(v) \in \mathbb{R}$.

The theory of real intervals is $T^I = \{M^I | M^I$ is a model of real intervals $\}$. By this instantiation, each model differs to another by the mapping from variables to real numbers. As a result, an assignment of real numbers to variables can be used to represent a model M^I; i.e. $\{v \mapsto r \in \mathbb{R} | v \in V\}$ represents a model. Given a map $\theta = \{v \mapsto r \in \mathbb{R} | v \in V\}$, we denote θ^I as a model of real intervals. If Π is an interval constraint, the notation $\Pi^p_{\mathbb{R}} = \{\theta^p_{\mathbb{R}} | \theta = \{v \mapsto r \in \mathbb{R} | v \in V\}$ and $\models_{\theta^I} \Pi\}$ represents the (sub-)theory of real numbers that each of its model (after converted into the model of intervals) satisfies the constraint Π.

Chapter 3

Over-Approximation and Under-Approximation

3.1 Approximation Theory

Definition 3.1.1. *Let T, T' be Σ-theories and φ be any Σ-formula.*

- *T' is an* over-approximation theory *(of T) iff T'-UNSAT of φ implies T-UNSAT of φ.*
- *T' is an* under-approximation theory *(of T) iff T'-SAT of φ implies T-SAT of φ.*

Theorem 3.1.1. *If T_O be an over-approximation theory of T, then for any Σ-formula φ: φ is T_O-VALID $\implies$ φ is T-VALID.*

Proof. φ is T_O-VALID $\implies$ $\neg\varphi$ is T_O-UNSAT (Lemma 2.2.2) $\implies$ $\neg\varphi$ is T-UNSAT (Definition 3.1.1) $\implies$ φ is T-VALID (Lemma 2.2.2) $\square$

A typical ICP applies $O.T$ only as an interval arithmetic. Later in Section **??**, we will instantiate interval arithmetic as $O.T$. Adding to $O.T$-valid, **raSAT** introduce testing as $U.T$ to accelerate SAT detection.

3.2 Interval Arithmetic as an Over-Approximation Theory

3.2.1 Real Intervals

We adopt the definition of real intervals from [8]:

Definition 3.2.1. *[8] Let a and b be reals such that $a \leq b$.*

$$\begin{array}{rcl}\langle a,b\rangle & \stackrel{def}{=} & \{x\in\mathbb{R}|a\le x\le b\}\\ \langle -\infty,b\rangle & \stackrel{def}{=} & \{x\in\mathbb{R}|x\le b\}\\ \langle a,+\infty\rangle & \stackrel{def}{=} & \{x\in\mathbb{R}|a\le x\}\\ \langle -\infty,+\infty\rangle & \stackrel{def}{=} & \mathbb{R}\end{array}$$

All intervals in this definition can be summarized by $\langle a,b\rangle$ where $a,b\in\mathbb{R}\cup\{-\infty,+\infty\}$ and $a\le b$ with the assumption that $\forall c\in\mathbb{R} - \infty < c < \infty$. Furthermore,Hickey et al. [8] also defined arithmetic operations for $\mathbb{R}\cup\{-\infty,+\infty\}$ which is summarized in Table .

$x+y$	$-\infty$	NR	0	PR	$+\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$\perp$
NR		NR	NR	$\mathbb{R}$	$+\infty$
0			0	PR	$+\infty$
PR				PR	$+\infty$
$+\infty$					$+\infty$

$x-y$	$-\infty$	NR	0	PR	$+\infty$
$-\infty$	$\perp$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
NR	$-\infty$	$\mathbb{R}$	PR	PR	$+\infty$
0	$-\infty$	NR	0	PR	$+\infty$
PR	$-\infty$	NR	NR	$\mathbb{R}$	$+\infty$
$+\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$\perp$

$x\times y$	$-\infty$	NR	0	PR	$+\infty$
$-\infty$	$+\infty$	$+\infty$	$\perp$	$-\infty$	$-\infty$
NR		PR	0	NR	$-\infty$
0			0	0	$\perp$
PR				PR	$+\infty$
$+\infty$					$+\infty$

Definition 3.2.2. *The set of all real intervals $\mathbb{I}$ is defined as $\mathbb{I}=\{\langle a,b\rangle|a,b\in\mathbb{R}\cup\{-\infty,+\infty\}$ and $a\le b\}$.*

3.2.2 Interval Arithmetic as an Over-Approximation Theory

A model $M^p_{IA}=(U^p_{IA},I^p_{IA})$ over intervals contains a set of all intervals $U^p_{IA}=\{\langle l,h\rangle|l,h\in\mathbb{R}\cup\{-\infty,+\infty\}$ and $l\le h\}$ and a map I^p_{IA} that satisfies the following conditions.

1. $I^p_{IA}(Real)=U^p_{IA}$
2. $\forall p\in P^p$; $I^p_{IA}(p)=U^p_{IA}\times U^p_{IA}\mapsto\{0,1\}$ where $I^p_{IA}(p)(i_1,i_2)=i_1\ p_{IA}\ i_2$. The definition of p_{IA} is as follow:
   - $\langle l_1,h_1\rangle \succ_{IA} \langle l_2,h_2\rangle = \begin{cases}1 & \text{if } l_1>h_2\\ 0 & \text{if } h_1\le l_2\end{cases}$
   - $\langle l_1,h_1\rangle \prec_{IA} \langle l_2,h_2\rangle = \begin{cases}1 & \text{if } h_1<l_2\\ 0 & \text{if } l_1\ge h_2\end{cases}$
   - $i_1\succeq_{IA} i_2 = 1-(i_1\prec_{IA} i_2)$
   - $i_1\preceq_{IA} i_2 = 1-(i_1\succ_{IA} i_2)$
   - $i_1\approx_{IA} i_2 = min(i_1\succeq_{IA} i_2, i_1\preceq_{IA} i_2)$

- $i_1 \not\approx_{IA} i_2 = 1 - (i_1 \approx_{IA} i_2)$

3. $\forall f \in F^p \setminus \{\mathbf{1}\}$; $I^p_{IA}(f) = U^p_{IA} \times U^p_{IA} \mapsto U^p_{IA}$ such that $I^p_{IA}(f)(i_1, i_2) = i_1\ f_{IA}\ i_2$ where f_{IA} satisfies the following properties:

   - $i_1 \oplus_{IA} i_2 \supseteq \{r_1 + r_2 | r_1 \in i_1 \text{ and } r_2 \in i_2\}$.
   - $i_1 \ominus_{IA} i_2 \supseteq \{r_1 - r_2 | r_1 \in i_1 \text{ and } r_2 \in i_2\}$.
   - $i_1 \otimes_{IA} i_2 \supseteq \{r_1 * r_2 | r_1 \in i_1 \text{ and } r_2 \in i_2\}$.

4. $I^p_{IA}(\mathbf{1}) = \langle 1, 1 \rangle$

5. $\forall v \in V$; $I^p_{IA} \in U^p_{IA}$

Theory $T^p_{IA} = \{M^p_{IA} | M^p_{IA} \text{ is a model over intervals}\}$. Each model differs to another by the mapping from variables to intervals. As a consequence, one assignment from variables to intervals can be used to describe an model. An assignment $\{v \mapsto i \in \mathbb{I} | v \in V\}$ and an interval constraint $\bigwedge\limits_{v \in V} v \in i$ are equivalent in terms of the set of assignments from variables to real numbers. So by abusing notation, for a constraint of the form $\Pi = \bigwedge\limits_{v \in V} v \in i$, we denote Π^p_{IA} as a model of interval arithmetics for polynomial constraints. By definition, $\{\Pi^p_{IA}\}$ represents the a sub-theory of T^p_{IA}.

Lemma 3.2.1. *Let $M^p_{IA} = (U^p_{IA}, I^p_{IA})$ be a model over intervals, we have:*

$$\forall i_1, i_2 \in U^p_{IA}; \forall r_1 \in i_2, r_2 \in i_2; \forall p \in P^p (i_1\ p_{IA}\ i_2 = 0 \implies \textit{not}\ (r_1\ p_{\mathbb{R}}\ r_2))$$

Proof. Let $i_1 = \langle l_1, h_1 \rangle$ and $i_2 = \langle l_2, h_2 \rangle$ where $l_1 \leq h_1$ and $l_2 \leq h_2$. We have:

- $r_1 \in i_1 \implies l_1 \leq r_1 \leq h_1$.
- $r_2 \in i_2 \implies l_2 \leq r_2 \leq h_2$.

Suppose that $i_1\ p_{IA}\ i_2 = 0$, we need to show *not* $(r_1\ p_{\mathbb{R}}\ r_2)$ by considering all the possible cases of p:

1. If p is $\succ$, the we have $\langle l_1, h_1 \rangle \succ_{IA} \langle l_2, h_2 \rangle = 0 \implies h_1 \leq l_2 \implies r_1 \leq r_2$ (because $r_1 \leq h_1$ and $l_2 \leq r_2$) $\implies$ *not* $(r_1 > r2) \implies$ *not* $(r_1 \succ_{\mathbb{R}} r_2)$.

2. If p is $\prec$, the we have $\langle l_1, h_1 \rangle \prec_{IA} \langle l_2, h_2 \rangle = 0 \implies l_1 \geq h_2 \implies r_1 \geq r_2$ (because $r_1 \geq l_1$ and $r_2 \leq h_2$) $\implies$ *not* $(r_1 < r_2) \implies$ *not* $(r_1 \prec_{\mathbb{R}} r_2)$.

3. If p is $\succeq$, the we have $\langle l_1, h_1 \rangle \succeq_{IA} \langle l_2, h_2 \rangle = 0 \implies 1 - (\langle l_1, h_1 \rangle \prec_{IA} \langle l_2, h_2 \rangle) = 0 \implies \langle l_1, h_1 \rangle \prec_{IA} \langle l_2, h_2 \rangle = 1 \implies h_1 < l_2 \implies r_1 < r_2$ (because $r_1 \leq h_1$ and $r_2 \geq l_2$) $\implies$ *not* $(r_1 \geq r_2) \implies$ *not* $(r_1 \succeq_{\mathbb{R}} r_2)$.

4. If p is $\preceq$, the we have $\langle l_1, h_1 \rangle \preceq_{IA} \langle l_2, h_2 \rangle = 0 \implies 1 - (\langle l_1, h_1 \rangle \succ_{IA} \langle l_2, h_2 \rangle) = 0 \implies \langle l_1, h_1 \rangle \succ_{IA} \langle l_2, h_2 \rangle = 1 \implies l_1 > h_2 \implies r_1 > r_2$ (because $r_1 \geq l_1$ and $r_2 \leq h_2$) $\implies$ *not* $(r_1 \leq r_2) \implies$ *not* $(r_1 \preceq_{\mathbb{R}} r_2)$.

5. If p is $\approx$, the we have $i_1 \approx_{IA} i_2 = 0 \implies min(i_1 \succeq_{IA} i_2, i_1 \preceq_{IA} i_2) = 0 \implies i_1 \succeq_{IA} i_2 = 0$ or $i_1 \preceq_{IA} i_2 = 0 \implies r_1 < r_2$ or $r_1 > r_2$ (as the third and fourth case of this proof) $\implies$ *not* $(r_1 = r_2) \implies$ *not* $(r_1 \approx_{\mathbb{R}} r_2)$.

6. If p is $\not\approx$, the we have $i_1 \not\approx_{IA} i_2 = 0 \implies 1 - (i_1 \approx_{IA} i_2) = 0 \implies min(i_1 \succeq_{IA} i_2, i_1 \preceq_{IA} i_2) = 1 \implies i_1 \succeq_{IA} i_2 = 1$ and $i_1 \preceq_{IA} i_2 = 1 \implies 1-(i_1 \prec_{IA} i_2) = 1$ and $1 - (i_1 \succ_{IA} i_2) = 1 \implies i_1 \prec_{IA} i_2 = 0$ and $i_1 \succ_{IA} i_2 = 0 \implies r_1 \geq r_2$ and $r_1 \leq r_2$ (as the first and second case of this proof) $\implies r_1 = r_2 \implies$ *not* $(r_1 \neq r_2) \implies$ *not* $(r_1 \not\approx_{\mathbb{R}} r_2)$.

□

Lemma 3.2.2. *Let* $\Pi = \bigwedge_{v\in V} v \in i$ *with* $i \in \mathbb{I}$, *we have* $\forall t \in TERM^p\ \forall M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}\ t^{M^p_{\mathbb{R}}} \in t^{\Pi^p_{IA}}$.

Proof. Proof is done by induction on structure of term. Let $t \in TERM^p$ and $M^p_{\mathbb{R}} = (\mathbb{R}, I^p_{\mathbb{R}}) \in \Pi^p_{\mathbb{R}}$

1. If $t = v \in V$, then $t^{M^p_{\mathbb{R}}} = I^p_{\mathbb{R}}(v) \in \Pi(v)$ because $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$. In addition, $t^{\Pi^p_{IA}} = \Pi^p_{\mathbb{R}}$.

2. If $t = \mathbf{1}$, then $t^{M^p_{\mathbb{R}}} = 1 \in \langle 1, 1\rangle = t^{\Pi^p_{IA}}$

3. If $t = t_1\ f\ t_2$ for some $f \in F^p \setminus \{0, 1\}$

   $t^{M^p_{\mathbb{R}}} = t_1^{M^p_{\mathbb{R}}}\ f_{\mathbb{R}}\ t_1^{M^p_{\mathbb{R}}}$

   $t^{\Pi^p_{IA}} = t_1^{\Pi^p_{IA}}\ f_{IA}\ t_2^{\Pi^p_{IA}}$

   By induction hypothesis, we have $t_1^{M^p_{\mathbb{R}}} \in t_1^{\Pi^p_{IA}}$ and $t_1^{M^p_{\mathbb{R}}} \in t_2^{\Pi^p_{IA}}$. In addition, due to the properties of f_{IA}: $t_1^{M^p_{\mathbb{R}}}\ f_{\mathbb{R}}\ t_1^{M^p_{\mathbb{R}}} \in t_1^{\Pi^p_{IA}}\ f_{IA}\ t_2^{\Pi^p_{IA}}$, or $t^{M^p_{\mathbb{R}}} \in t^{\Pi^p_{IA}}$

□

Theorem 3.2.1. *If* $\Pi = \bigwedge_{v\in V} v \in i$ *with* $i \in \mathbb{I}$ *is a map from variables to intervals, then* $\{\Pi^p_{IA}\}$ *is an over-approximation of* $\Pi^p_{\mathbb{R}}$.

Proof. In order to prove Theorem 3.2.1, we need to prove that $\forall\varphi^p(\varphi^p$ is $\{\Pi^p_{IA}\}$-UNSAT $\implies \varphi^p$ is $\Pi^p_{\mathbb{R}}$-UNSAT).

Given an Σ^p-formula φ^p and suppose that φ^p is $\{\Pi^p_{IA}\}$-UNSAT. Suppose φ^p is not $\Pi^p_{\mathbb{R}}$-UNSAT, that means it is either $\Pi^p_{\mathbb{R}}$-SAT or $\Pi^p_{\mathbb{R}}$-VALID. In either case, there exist at least a model $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$ such that $\models_{M^p_{\mathbb{R}}} \varphi^p \iff (\varphi^p)^{M^p_{\mathbb{R}}} = true$.

1. If $\varphi^p = p(t_1, t_2)$ for some $p \in P^p$, then $(\varphi^p)^{M^p_{\mathbb{R}}} = true \implies t_1^{M^p_{\mathbb{R}}}\ p_{\mathbb{R}}\ t_2^{M^p_{\mathbb{R}}}$.

   On the other hand, φ^p is $\{\Pi^p_{IA}\}$-UNSAT $\implies (\varphi^p)^{\Pi^p_{IA}} = 0 \implies t_1^{\Pi^p_{IA}}\ p_{IA}\ t_2^{\Pi^p_{IA}} = 0$. In addition, because $t_1^{M^p_{\mathbb{R}}} \in t_1^{\Pi^p_{IA}}$ and $t_2^{M^p_{\mathbb{R}}} \in t_2^{\Pi^p_{IA}}$ (Lemma 3.2.2), $t_1^{\Pi^p_{IA}}\ p_{IA}\ t_2^{\Pi^p_{IA}} = 0 \implies \neg(t_1^{M^p_{\mathbb{R}}}\ p_{\mathbb{R}}\ t_2^{M^p_{\mathbb{R}}})$ (Lemma 3.2.1). This is a contradiction. As the result, φ^p must be $\Pi^p_{\mathbb{R}}$-UNSAT.

2. If $\varphi^p = \varphi_1^p \wedge \varphi_2^p$, then we have φ^p is $\{\Pi_{IA}^p\}$-UNSAT $\implies \not\models_{\Pi_{IA}^p} (\varphi_1^p \wedge \varphi_2^p) \implies (\varphi_1^p \wedge \varphi_2^p)^{\Pi_{IA}^p} = 0 \implies (\varphi_1^p)^{\Pi_{IA}^p} \wedge (\varphi_2^p)^{\Pi_{IA}^p} = 0 \implies (\varphi_1^p)^{\Pi_{IA}^p} = 0$ and $(\varphi_2^p)^{\Pi_{IA}^p} = 0$. By induction hypothesis, φ_1^p and φ_2^p are $\Pi_{\mathbb{R}}^p$-UNSAT $\implies \forall M_{\mathbb{R}}^p \in \Pi_{\mathbb{R}}^p \not\models_{M_{IA}^p} \varphi_1^p$ and $\forall M_{\mathbb{R}}^p \in \Pi_{\mathbb{R}}^p \not\models_{M_{IA}^p} \varphi_2^p \implies \forall M_{\mathbb{R}}^p \in \Pi_{\mathbb{R}}^p \not\models_{M_{IA}^p} (\varphi_1^p \wedge \varphi_2^p) \implies \varphi_1^p \wedge \varphi_2^p$ is $\Pi_{\mathbb{R}}^p$-UNSAT.

□

3.3 Testing as an Under-Approximation Theory

Definition 3.3.1. *Let $T^p \subseteq T_{\mathbb{R}}^p$ be a sub-theory of real numbers. Any sub-theory T_T^p of T^p, i.e. $T_T^p \subseteq T^p$ is call a theory of testing with respect to T^p.*

Theorem 3.3.1. *If T_T^p is a theory of testing w.r.t T^p, T_R^p is an under-approximation of T^p.*

Proof. Let φ^p is a Σ^p-formula and suppose it is T_T^p-SAT. We need to prove φ^p is T^p-SAT. We have φ^p is T_T^p-SAT $\implies \exists M^p \in T_T^p \models_{M^p} \varphi^p \implies \exists M^p \in T^p \models_{M^p} \varphi^p$ (because $T_T^p \subseteq T^p$) $\implies \varphi^p$ is T^p-SAT. □

Given a theory T^p, we randomly select a number of its models to form a theory of testing T_T^p w.r.t T^p.

3.4 raSAT Loop

In other word, each model can be represented by an assignment of one real number to each variable. State of search procedure contains $(\Pi, \varphi, \bar{\Pi}, \varphi^V, \varphi^U, \varepsilon, \tau)$ where

- Π is the interval constraint.
- φ represents the polynomial constraint.
- $\bar{\Pi} = \bigwedge\limits_{v_i \in V} v_i \in i_i$ with $i_i \in \mathbb{I}$ is the result of DPLL procedure on Π.
- φ^V contains the constraints that are valid under over-approximation.
- φ^U is the set of constraints which are UNKNOWN under over-approximation.
- ε indicates the threshold to stop decomposing intervals.
- τ is a flag to mark whether the threshold of intervals has been reached.

The transition rules are described in Table 3.1. Figure 3.1 illustrates the transition system.

$$\frac{\Pi\models_{SAT}\bot}{(\Pi,\varphi,\emptyset,\emptyset,\emptyset,\varepsilon,\emptyset)\rightarrow UNSAT}\ \text{Π\_UNSAT\_UNSAT}$$

$$\frac{\Pi\models_{SAT}\bot}{(\Pi,\varphi,\emptyset,\emptyset,\emptyset,\varepsilon,\top)\rightarrow UNSAT}\ \text{Π\_UNSAT\_UNKNOWN}$$

$$\frac{\Pi\models_{SAT}\bar{\Pi}\quad \Pi^{c'}=flattern(\bar{\Pi})}{(\Pi,\varphi,\emptyset,\emptyset,\emptyset,\varepsilon,\tau)\rightarrow(\Pi,\varphi,\Pi^{c'},\emptyset,\emptyset,\varepsilon,\tau)}\ \text{Π\_SAT}$$

$$\frac{\bar{\Pi}\neq\emptyset\quad \varphi^V\wedge\varphi^U=\varphi\quad \varphi^V\cup\varphi^U=\emptyset\quad \varphi^V \text{ is } \bar{\Pi}_{IA}\text{-VALID}}{(\Pi,\varphi,\bar{\Pi},\emptyset,\emptyset,\varepsilon,\tau)\rightarrow(\Pi,\varphi,\bar{\Pi},\varphi^V,\varphi^U,\varepsilon,\tau)}\ \text{IA\_SAT}$$

$$\frac{\varphi^V=\varphi}{(\Pi,\varphi,\bar{\Pi},\varphi^V,\varphi^U,\varepsilon,\tau)\rightarrow SAT}\ \text{IA\_VALID}$$

$$\frac{\bar{\Pi}\neq\emptyset\quad \varphi^U\neq\emptyset\quad \varphi^U \text{ is } \bar{\Pi}_T\text{-SAT}}{(\Pi,\varphi,\bar{\Pi},\varphi^V,\varphi^U,\varepsilon,\tau)\rightarrow SAT}\ \text{TEST\_SAT}$$

$$\frac{\varphi^U \text{ is } \bar{\Pi}_T\text{-UNSAT}\quad \bar{\Pi}=\bigwedge_{v_i\in V} v_i\in\langle l_i,h_i\rangle\quad \forall i(h_i-l_i<\varepsilon)}{(\Pi,\varphi,\bar{\Pi},\varphi^V,\varphi^U,\varepsilon,\tau)\rightarrow(\Pi\wedge\neg\bar{\Pi},\varphi,\emptyset,\emptyset,\emptyset,\varepsilon,\top)}\ \text{THRESHOLD}$$

$$\frac{\varphi^U \text{ is } \bar{\Pi}_T\text{-UNSAT}\quad \bar{\Pi}=\bigwedge_{v_i\in V} v_i\in\langle l_i,h_i\rangle\quad \exists j(h_j-l_j>\varepsilon)\quad l_j<d\in\mathbb{R}<h_j\quad I_j=v_j\in\langle l_j,h_j\rangle\quad I_{j1}=x_j\in\langle l_j,d\rangle\quad I_{j2}=x_j\in\langle d,h_j\rangle}{(\Pi,\varphi,\bar{\Pi},\varphi^V,\varphi^U,\varepsilon,\tau)\rightarrow(\Pi\wedge(\neg I_j\vee I_{j1}\vee I_{j2})\wedge(I_j\vee\neg I_{j1})\wedge(I_j\vee\neg I_{j2})\wedge(\neg I_{j1}\vee\neg I_{j2}),\varphi,\emptyset,\emptyset,\emptyset,\varepsilon,\tau)}$$

$$\frac{\{\varphi_i|\varphi_i\wedge\varphi_i'=\varphi;\ \varphi_i \text{ is } \bar{\Pi}_{IA}\text{-UNSAT}\}\quad var(\varphi_i)=\{x_{ij}|j=1,\cdots,m_i\}\quad (I_{ij}=x_{ij}\in[l_{ij},h_{ij}])\in\bar{\Pi}}{(\Pi,\varphi,\bar{\Pi},\emptyset,\emptyset,\varepsilon,\tau)\rightarrow(\Pi\wedge\bigwedge_i\bigvee_{j=1}^{m_i}\neg I_{ij},\varphi,\emptyset,\emptyset,\emptyset,\varepsilon,\tau)}\ \text{IA\_UNSAT}$$

Table 3.1: Transition rules

3.5 Soundness - Completeness

3.5.1 Soundness

Theorem 3.5.1. *Let $(\Pi, \varphi, \bar{\Pi}, \varphi^V, \varphi^U, \varepsilon, \tau)$ be any state of our system, then the following properties are invariants:*

1. $\bar{\Pi}_{\mathbb{R}} \subseteq T^p_{\mathbb{R}}$
2. φ^V *is* $\bar{\Pi}_R$*-VALID.*
3. $\varphi^U = \emptyset \vee (\varphi = \varphi^U \wedge \varphi^V)$
4. φ *is* $\Pi_{\mathbb{R}}$*-UNSAT* $\implies$ φ *is* $T^p_{\mathbb{R}}$*-UNSAT*

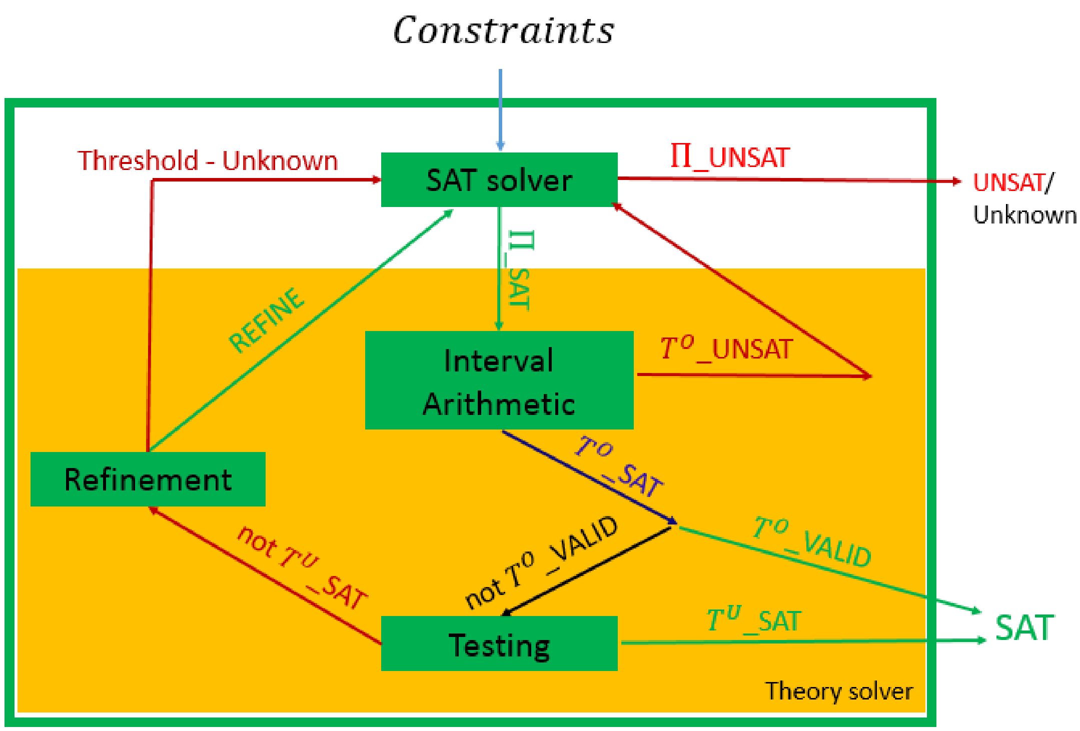


Figure 3.1: **raSAT** design

Proof. 1. Easy from the definition.

2. Easy to see.

3. Easy from the transitions.

4. The proof is done inductively:

   – Initial state: $(\bigwedge_{v\in V} v \in (-\infty,+\infty), \varphi, \emptyset, \emptyset, \emptyset)$ and by definition, $(\bigwedge_{v\in V} v \in (-\infty,+\infty))_{\mathbb{R}} = T_R^p$. Then, the invariant trivially holds.

   – Each transition is considered:

     – For rules Π_SAT, IA_SAT the interval constraint Π does not changed, so if the properties holds for the former state, it also does for the later one.

     – For REFINE transition:
     Denote $\Pi' = \Pi \wedge (\neg I_j \vee I_{j1} \vee I_{j2}) \wedge (I_j \vee \neg I_{j1}) \wedge (I_j \vee \neg I_{j2}) \wedge (\neg I_{j1} \vee \neg I_{j2})$. We will prove that φ is $\Pi'_{\mathbb{R}}$-UNSAT $\implies$ φ is $\Pi_{\mathbb{R}}$-UNSAT. First suppose that φ is $\Pi'_{\mathbb{R}}$-UNSAT. Let

     – for IA_UNSAT transition,

$\square$

Theorem 3.5.2. *Let φ be the polynomial constraint to be solved. Starting with the state $(\Pi, \varphi, \emptyset, \emptyset, \emptyset)$, if our transitional system terminates and output:*

- *SAT then φ is $T^p_{\mathbb{R}}$-SAT.*
- *UNSAT then φ is $T^p_{\mathbb{R}}$-UNSAT.*

Proof. 1. If the system output SAT, there are two possibles transition to SAT:

- In the case of IA_VALID, we have φ^V is $\bar{\Pi}_R$-VALID (invariant 2) $\implies$ φ^V is $\bar{\Pi}_{\mathbb{R}}$-SAT $\implies$ φ is $\bar{\Pi}_{\mathbb{R}}$-SAT (because $\varphi^V = \varphi$ is the condition of this transition). In addition, following invariant 1, we have $\bar{\Pi}_{\mathbb{R}} \subseteq T^p_{\mathbb{R}}$, then φ is $T^p_{\mathbb{R}}$-SAT (Lemma 2.2.3).
- In the case of TEST_SAT, φ^U is $\bar{\Pi}_{\mathbb{R}}$-SAT $\implies$ $\exists M \in \bar{\Pi}_{\mathbb{R}} \quad \models_M \varphi^V$. Let $M^c_{\mathbb{R}} \in \bar{\Pi}_{\mathbb{R}}$ such that $\models_{M^c_{\mathbb{R}}} \varphi^V$ which implies $(\varphi^V)^{M^c_{\mathbb{R}}} = 1$. In addition, because φ^V is $\bar{\Pi}_{\mathbb{R}}$-VALID (invariant 2) and $M^c_{\mathbb{R}} \in \bar{\Pi}_{\mathbb{R}}$, we have $\models_{M^c_{\mathbb{R}}} \varphi^U$ or $(\varphi^U)^{M^c_{\mathbb{R}}} = 1$. Consider the evaluation of φ under the model $M^c_{\mathbb{R}}$: $(\varphi)^{M^c_{\mathbb{R}}} = (\varphi^U \wedge \varphi^V)^{M^c_{\mathbb{R}}}$ (invariant 3) $= min((\varphi^U)^{M^c_{\mathbb{R}}}, (\varphi^V)^{M^c_{\mathbb{R}}}) = min(1, 1) = 1 \implies \models_{M^c_{\mathbb{R}}} \varphi \implies \varphi$ is $\bar{\Pi}_{\mathbb{R}}$-SAT $\implies$ φ is $T^p_{\mathbb{R}}$-SAT (because of invariant 1 and Lemma 3.2.1)

2. If the system output UNSAT, there is only one transition of rule Π_UNSAT. Because $\Pi \models_{SAT} \perp$, $\Pi_{\mathbb{R}}$ is empty (Lemma ?). As a result, φ is $\Pi_{\mathbb{R}}$-UNSAT which implies that φ is $T^p_{\mathbb{R}}$-UNSAT (invariant 4).

□

3.5.2 Completeness

Definition 3.5.1. *Let* $\Pi = \bigwedge_{v \in V} v \in i$ *with* $i \in \mathbb{I}$, *and* $\varphi = \bigwedge_{i=1}^{n} f_i > 0$. *An over-approximation* T^p_O *of* $T^p_{\mathbb{R}}$ *is complete if for each open set* $O \subset \{(r_1, r_2, \cdots) | \models_{\theta^{\mathbb{I}}} \Pi; \theta = \{v_i \mapsto r_i \in \mathbb{R} | v_i \in V\}\}$, *we have* $\forall c \in O;\ \forall \delta > 0;\ \exists \gamma > 0;\ (\langle c - \gamma, c + \gamma \rangle \in O$ *and* $\models_{\Pi'^p_{\mathbb{R}}} \bigwedge_{i=1}^{n} (f_i(c) - \delta < f_i(x) < f_i(c) + \delta)$ *where* $\Pi' = \bigwedge_{v_i \in V} v_i \in \langle c_i - \gamma, c_i + \gamma \rangle$, *and* $c = (c_1, c_2, \cdots)$

Definition 3.5.2.

Theorem 3.5.3.

Chapter 4

Interval Arithmetic

Interval Arithmetic is defined formally in Section 3.2 of Chapter 3. This chapter is going to present two instances of Interval Arithmetic which are used in raSAT: Classical Interval and Affine Interval. These two kinds differ to each other in the way they represent intervals and interpret function symbols.

4.1 Classical Interval

A model $M^p_{CI} = (U^p_{CI}, I^p_{CI})$ over intervals contains a set of all intervals $U^p_{CI} = U^p_{IA}$ and a map I^p_{CI} that satisfies the following conditions.

1. $I^p_{CI}(Real) = I^p_{IA}(Real)$
2. $\forall p \in P^p$; $I^p_{CI}(p) = I^p_{IA}(p)$
3. $\forall f \in F^p \setminus \{\mathbf{1}\}$; $I^p_{CI}(f) = U^p_{CI} \times U^p_{CI} \mapsto U^p_{CI}$ such that $I^p_{CI}(f)(i_1, i_2) = i_1\ f_{CI}\ i_2$ where the definition of f_{CI} is:
   - $\langle l_1, h_1\rangle \oplus_{CI} \langle l_2, h_2\rangle = \langle l_1 + l_2, h_1 + h_2\rangle$.
   - $\langle l_1, h_1\rangle \ominus_{CI} \langle l_2, h_2\rangle = \langle l_1 - h_2, h_1 - l_2\rangle$.
   - Operation $i_1 \otimes_{CI} i_2$ is defined using case analysis on the types of i_1 and i_2. First, the intervals are classified into the following:
     - $P = \{\langle a, b\rangle | a \geq 0 \wedge b > 0\}$
     - $N = \{\langle a, b\rangle | b \leq 0 \wedge a < 0\}$
     - $M = \{\langle a, b\rangle | a < 0 < b\}$
     - $Z = \{\langle a, b\rangle\}$ The definition of $\otimes_{CI}$ is given in Table

Class of $\langle l_1, h_1\rangle$	Class of $\langle l_2, h_2\rangle$	$\langle l_1, h_1\rangle \otimes_{CI} \langle l_2, h_2\rangle$
P	P	$\langle l_1 \times l_2, h_1 \times h_2\rangle$
P	M	$\langle h_1 \times l_2, h_1 \times h_2\rangle$
P	N	$\langle h_1 \times l_2, l_1 \times h_2\rangle$
M	P	$\langle l_1 \times h_2, h_1 \times h_2\rangle$
M	M	$\langle \min(l_1 \times h_2, h_1 \times l_2), max(l_1 \times l_2, h_1 \times h_2)\rangle$
M	N	$\langle h_1 \times l_2, l_1 \times l_2\rangle$
N	P	$\langle l_1 \times h_2, h_1 \times l_2\rangle$
N	M	$\langle l_1 \times h_2, l_1 \times l_2\rangle$
N	N	$\langle h_1 \times h_2, l_1 \times l_2\rangle$
Z	P, N, M, Z	$\langle 0, 0\rangle$
P, N, M	Z	$\langle 0, 0\rangle$

4. $I^p_{CI}(\mathbf{1}) = \langle 1, 1\rangle$

5. $\forall v \in V;\ I^p_{CI} \in U^p_{CI}$

Theory $T^p_{CI} = \{M^p_{CI} | M^p_{CI}$ is a model over intervals$\}$. Each model differs to another by the mapping from variables to intervals. As a consequence, one assignment from variables to intervals can be used to describe an model. We denote Π^p_{CI} as the model represented by $\Pi = \{x \in \langle l, h\rangle | v \in V\}$.

Theorem 4.1.1. *CI is an IA.*

Proof. Easy. □

4.2 Affine Interval

Affine Interval use the formula $a_0 + \sum_{i=1}^{n} a_i\epsilon_i$ to represent the interval $\langle a_0 - \sum_{i=1}^{n} |a_i|, a_0 + \sum_{i=1}^{n} |a_i|\rangle$ with $a_i \in \mathbb{R}$ for $i = 0, 1, \cdots$. For example, the affine interval form of $(x \in)\langle 2, 4\rangle$ and $(y \in)\langle 0, 2\rangle$ is $3 + \epsilon_1$ and $1 + \epsilon_2$ respectively, thus:

$$\begin{aligned} x^2 - x \times y &= (3 + \epsilon_1)^2 - (3 + \epsilon_1) \times (1 + \epsilon_2) \\ &= 9 + 6\epsilon_1 + \epsilon_1^2 - (3 + 3\epsilon_2 + \epsilon_1 + \epsilon_1\epsilon_2) \\ &= 6 + 5\epsilon_1 - 3\epsilon_2 + \epsilon_1^2 + \epsilon_1\epsilon_2 \end{aligned}$$

Types of affine interval vary by choices of estimating multiplications ϵ_1^2 and $\epsilon_1\epsilon_2$:

1. AA [4, 20] replaces $\epsilon_1\epsilon_2$ by a fresh noise symbol.

2. AF1 and AF2 [13] prepares a fixed noise symbol for any $\epsilon_1\epsilon_2$.

3. EAI [14] replaces $\epsilon_1\epsilon_2$ by $\langle -1, 1\rangle\epsilon_1$ or $\langle -1, 1\rangle\epsilon_2$.

4. AF2 [13] replaces ϵ_1^2 by the fixed noise symbols ϵ_+ or ϵ_-.

A model $M^p_{AF2} = (U^p_{AF2}, I^p_{AF2})$ over intervals contains a set of all intervals $U^p_{AF2} = \{a_0 + \sum_{i=1}^{n} a_i\epsilon_i + a_{n+1}\epsilon_+ + a_{n+2}\epsilon_- + a_{n+3}\epsilon_\pm | \forall i \in \{0, 1, \cdots, n+3\}; a_i \in \mathbb{R}\}$ and a map I^p_{AF2} that satisfies the following conditions.

1. $I^p_{AF2}(Real) = U^p_{AF2}$
2. $\forall p \in P^p$; $I^p_{AF2}(p) = U^p_{AF2} \times U^p_{AF2} \mapsto \{true, false\}$ such that $I^p_{AF2}(p)(a_0 + \sum_{i=1}^{n} a_i\epsilon_i + a_{n+1}\epsilon_+ + a_{n+2}\epsilon_- + a_{n+3}\epsilon_\pm, b_0 + \sum_{i=1}^{n} b_i\epsilon_i + b_{n+1}\epsilon_+ + b_{n+2}\epsilon_- + b_{n+3}\epsilon_\pm) = I^p_{AI}(p)(\langle a_0 - \sum_{i=1}^{n} |a_i| - a_{n+2} - a_{n+3}, a_0 + \sum_{i=1}^{n} |a_i| + a_{n+1} + a_{n+3}\rangle, \langle b_0 - \sum_{i=1}^{n} |b_i| - b_{n+2} - b_{n+3}, b_0 + \sum_{i=1}^{n} |b_i| + b_{n+1} + b_{n+3}\rangle)$
3. $\forall f \in F^p \backslash \{\mathbf{1}\}$; $I^p_{AF2}(f) = U^p_{AF2} \times U^p_{AF2} \mapsto U^p_{AF2}$ such that $I^p_{AF2}(f)(i_1, i_2) = i_1 \; f_{AF2} \; i_2$ where the definition of f_{AF2} is as following. Let $i_1 = a_0 + \sum_{i=1}^{n} a_i\epsilon_i + a_{n+1}\epsilon_+ + a_{n+2}\epsilon_- + a_{n+3}\epsilon_\pm$ and $i_2 = b_0 + \sum_{i=1}^{n} b_i\epsilon_i + b_{n+1}\epsilon_+ + b_{n+2}\epsilon_- + b_{n+3}\epsilon_\pm$, then:

   – $i_1 \oplus_{AF2} i_2 = a_0 + b_0 + \sum_{i=1}^{n}(a_i + b_i)\epsilon_i + (a_{n+1} + b_{n+1})\epsilon_+ + (a_{n+2} + b_{n+2})\epsilon_- + (a_{n+3} + b_{n+3})\epsilon_\pm$.

   – $i_1 \ominus_{AF2} i_2 = a_0 - b_0 + \sum_{i=1}^{n}(a_i - b_i)\epsilon_i + (a_{n+1} + b_{n+1})\epsilon_+ + (a_{n+2} + b_{n+2})\epsilon_- + (a_{n+3} + b_{n+3})\epsilon_\pm$.

   – $i_1 \otimes_{AF2} i_2 = a_0 b_0 + \sum_{i=1}^{n}(a_0 b_i + a_i b_0)\epsilon_i + K_1\epsilon_+ + K_2\epsilon_- + K_3\epsilon_\pm$, where:

   $$K_1 = \sum_{i=1, a_i b_i > 0}^{n+3} a_i b_i + \begin{cases} a_0 b_{n+1} + a_{n+1} b_0 & \text{if } a_0 \geq 0 \text{ and } b_0 \geq 0 \\ a_0 b_{n+1} - a_{n+2} b_0 & \text{if } a_0 \geq 0 \text{ and } b_0 < 0 \\ -a_0 b_{n+2} + a_{n+1} b_0 & \text{if } a_0 < 0 \text{ and } b_0 \geq 0 \\ -a_0 b_{n+2} - a_{n+2} b_0 & \text{if } a_0 < 0 \text{ and } b_0 < 0 \end{cases}$$

   $$K_2 = \sum_{i=1, a_i b_i < 0}^{n+3} a_i b_i + \begin{cases} a_0 b_{n+2} + a_{n+2} b_0 & \text{if } a_0 \geq 0 \text{ and } b_0 \geq 0 \\ a_0 b_{n+2} - a_{n+1} b_0 & \text{if } a_0 \geq 0 \text{ and } b_0 < 0 \\ -a_0 b_{n+1} + a_{n+2} b_0 & \text{if } a_0 < 0 \text{ and } b_0 \geq 0 \\ -a_0 b_{n+1} - a_{n+1} b_0 & \text{if } a_0 < 0 \text{ and } b_0 < 0 \end{cases}$$

   $$K_3 = \sum_{i=1}^{n+3} \sum_{j=1, j \neq i}^{n+3} |a_i b_j| + |a_0| b_{n+3} + a_{n+3} |b_0|$$

4. $I^p_{AF2}(\mathbf{1}) = 1$
5. $\forall v \in V$; $I^p_{AF2} \in U^p_{AF2}$

Theory $T^p_{AF2} = \{M^p_{AF2} | M^p_{AF2}$ is a model over intervals$\}$. Each model differs to another by the mapping from variables to intervals. As a consequence, one assignment from variables to intervals can be used to describe an model. We denote Π^p_{CI} as the model represented by $\Pi = \{x \in \langle l, h\rangle | v \in V\}$.

Theorem 4.2.1. *AI is an IA.*

Proof. Easy. $\square$

Chapter 5

Design Strategies

We implemented a number of strategies for improving efficiency of raSAT: incremental search and refinement heuristics.

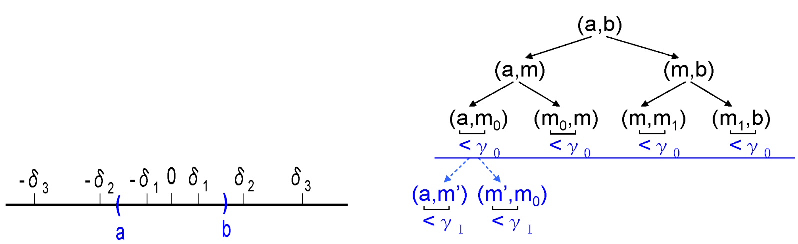
5.1 Incremental search

raSAT applies three incremental strategies, (1) *incremental windening*, (2) *incremental deepening* and (3) *incremental testing*. Let $F = \exists x_1 \in I_1 \cdots x_n \in I_n. \bigwedge_{j=1}^{m} f_j > 0$ for $I_i = (a_i, b_i)$.

5.1.1 Incremental Windening

Given $0 < \delta_0 < \delta_1 < \cdots$, *incremental windening* starts with $F_0 = \exists x_1 \in I_1 \cap (-\delta_0, \delta_0) \cdots x_n \in I_n \cap (-\delta_0, \delta_0). \bigwedge_{j=1}^{m} f_j > 0$, and if it finishes with UNSAT, it runs with $F_1 = \exists x_1 \in I_1 \cap (-\delta_1, \delta_1) \cdots x_n \in I_n \cap (-\delta_1, \delta_1). \bigwedge_{j=1}^{m} f_j > 0$, and so on (Fig. 5.1 (a)).

Note that if $\delta_i < \infty$, **raSAT** applies an Affine interval; otherwise, it uses CI. Experiments in Section **??** are performed with $\delta_0 = 10$ and $\delta_1 = \infty$.



(a) Incremenal widening (b) Incremental Deepening

Figure 5.1: Incremental Widening and Deepening

5.1.2 Incremental Deepening

Starting with $F = \exists x_1 \in I_1 \cdots x_n \in I_n. \bigwedge_{j=1}^{m} f_j > 0$, $I_1 \times \cdots \times I_n$ is decomposed into many boxes, and F becomes the disjunction of existential formulae corrsponding to these boxes. **raSAT** searches these boxes in depth-first manner. Figure **??** illustrates the search procedure of raSAT. First, the initially box I is examined (by IA and testing) and suppose raSAT detects neither SAT or UNSAT here. As a consequence, I is decomposed into 2 boxes I_{11} and I_{12} such that $\bar{x} \in I \leftrightarrow \bar{x} \in I_{11} \vee \bar{x} \in I_{12}$. raSAT will choose one of the two boxes to explore, for example I_{11}. If IA and testing cannot detect satisfiability of the given constraint in the box I_{11}, decomposition of this box takes place: $\bar{x} \in I_{11} \leftrightarrow \bar{x} \in I_{21} \vee \bar{x} \in I_{22}$. Either I_{21} or I_{22} is selected next. The process continues until SAT/UNSAT is detected which may leads to exhaustive local search. To avoid it, **raSAT** applies a threshold γ, such that no more decomposition will be applied when a box becomes smaller than γ. If neither SAT nor UNSAT is detected, **raSAT** restarts with a smaller threshold.

Let $\gamma_0 > \gamma_1 > \cdots > 0$, and **raSAT** incrementally deepens its search with these thresholds, i.e., starting with δ_0, and if it fails, restart with δ_1, and so on (Fig 5.1 (b)).

5.1.3 Incremental Testing

One obstacle in testing is the exponentially large number of test instances. If 2 values are generated for each of n variables, 2^n test cases (combinations of generated values) will present.

Example 5.1.1. *Suppose $\{x, y\}$ is the set of variables which appears in the input constraint and let $\{2, 9\}$ and $\{5, 8\}$ are generated values for x and y respectively. In total 4 test cases arise: $(x, y) = (2, 5), (2, 8), (9, 5), (9, 8)$.*

In order to tackle the problem, the following strategies are proposed:

1. Restrict the number of test cases to 2^{10} by choosing most 10 influential variables which are decided by the following procedures for generating multiple (2) test values.
   - Select API by SAT-likelihood
   - Select 1 variable of each selected API.
2. Incrementally generate test values for variables to prune test cases that do not satisfy an API. This was proposed by Khanh and Ogawa in [11]:
   - Dynamically sort the IA-SAT APIs.
   - Generate the test values for variables of selected APIs.

Example 5.1.2. *Let $x^2 > 4$ and $x * y > 0$ are two IA-VALID APIs to be tested and somehow they are sorted in that order, i.e. $x^2 > 4$ is selected before $x * y > 0$. Suppose $\{1, 3\}$ are generated as test values for x which are is enough to test the first selected API, i.e. $x^2 > 4$. As a result of testing, $x = 1$ is excluded from the* *satisfiable test cases whilst $x = 3$ is not. Next, when $x * y > 0$ is considered, y needs to be generated 2 values, e.g. $\{-3, 4\}$ and two test cases $(x, y) = (3, -3), (3, 4)$ come out to be checked. In this example, $(x, y) = (1, -3)$ and $(x, y) = (1, 4)$ are early pruned by only testing $x = 1$ against $x^2 > 4$.*

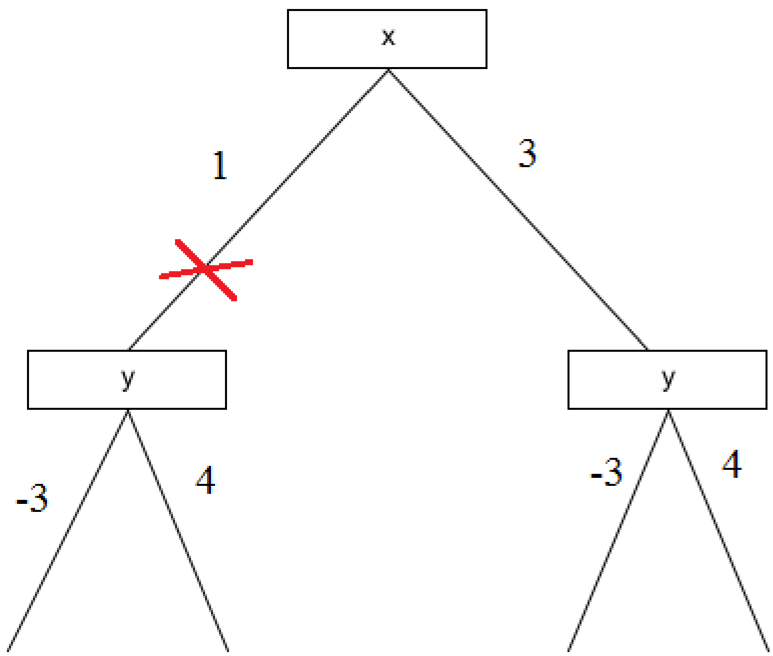


Figure 5.2: Incremental Testing Example

5.2 SAT-directed Heuristics Measure

With several hundred variables, we observe that an SMT solver works when either SAT, or UNSAT with small UNSAT core. For the latter, we need an efficient heuristics to find an UNSAT core, which is left as future work. For the former, the keys are how to choose variables to decompose, and how to choose a box to explore. **raSAT** chooses such a variable in two steps; first it selects a *test-UNSAT API*, and then chooses a variable that appears in the API. We design SAT-directed heuristic measures based on the interval arithemtic $(O.T)$.

Let $F = \exists x_1 \in I_1 \cdots x_n \in I_n. \bigwedge_{j=1}^{m} f_j > 0$ becomes $\vee\ (\exists x_1 \in I'_1 \cdots x_n \in I'_n. \bigwedge_{j=1}^{m} f_j > 0)$ after box decomposition. For $\exists x_1 \in I'_1 \cdots x_n \in I'_n. \bigwedge_{j=1}^{m} f_j > 0$, if some $f_j > 0$ is UNSAT, the box $I'_1 \times \cdots \times I'_n$ is UNSAT. If every $f_j > 0$ is SAT, F is SAT. Thus, if the box $I'_1 \times \cdots \times I'_n$ needs to be explore, it must contain a test-UNSAT API (thus IA-SAT).

We denote the estimated range of f_j for $x_1 \in I'_1 \cdots x_n \in I'_n$ with IA $(O.T)$ by $range(f_j, I'_1 \times \cdots \times I'_n)$. If an IA is an affine interval, it is in the form $[c_1, d_1]\epsilon_1 + \cdots + [c_n, d_n]\epsilon_n$, and by instanciating ϵ_i with $[-1, 1]$, the resulting classical interval coincides with $range(f_j, I'_1 \times \cdots \times I'_n)$. We define

- *Sensitivity* of a variable x_i in a test-UNSAT API $f_j > 0$ is $max(|c_i|, |d_i|)$.
- *SAT-likelyhood* of an API $f_j > 0$ is $|I \cap (0, \infty)|/|I|$, and
- *SAT-likelyhood* of a box $I'_1 \times \cdots \times I'_n$ is the least SAT-likelyhood of test-UNSAT APIs.

Example 5.2.1. *In Example* **??***,*

- *sensitivity is estimated* 1 *for* x *and* 2 *for* y *by* AF_2*, and* $3\frac{1}{4}$ *for* x *and* 2 *for* y*.*
- *SAT-likelyhood of* f *is estimated* $0.4 = \frac{6}{9-(-6)}$ *by* AF_2 *and* $0.36 = \frac{4.5}{4.5-(-8)}$ *by* CAI*.*

SAT-likelyhood intends to estimate APIs how likely to be SAT. For choosing variables, **raSAT** first choose a test-UNSAT API by SAT-likelyhood. There are two choices, either *the largest* or *the least*. *Sensitivity* of a variable intends to estimate how a variable is influencial to the value of an API. From a selected API by SAT-likelyhood, **raSAT** selects a variable with the largest sensitivity. This selection of variables are used for (1) *multiple test instances generation*, and (2) *decomposition*. For test generation, we will select multiple variables by repeating the selection.

For choosing a box to explore, **raSAT** chooses more likely to be SAT. There are two choice, (1) a box with the largest SAT-likelyhood, and (2) a box with the largest number of SAT (either IA-valid or test-SAT) APIs.

5.3 Other

UNSAT core, test case generation, interval decomposition

Chapter 6

Experiments

This chapter is going to present the experiments results which reflect how effective our designed strategies are. In addition, comparison between raSAT, Z3 and iSAT3 will be also shown. The experiments were done on a system with Intel Xeon E5-2680v2 2.80GHz and 4 GB of RAM. In the experiments, we exclude the problems which contain equalities because currently raSAT focuses on inequalities only.

6.1 Experiments on Strategy Combinations

We perform experiments only on Zankl, and Meti-Tarski families.

Our combinations of strategies mentioned in Section **??** are,

Selecting a test-UNSAT API	Selecting a box (to explore):	Selcting a variable:
(1) Least SAT-likelyhood.	(3) Largest number of SAT APIs.	(8) Largest sensitibity.
(2) Largest SAT-likelyhood.	(4) Least number of SAT APIs.	
	(5) Largest SAT-likelyhood.	
	(6) Least SAT-likelyhood.	
(10) Random.	(7) Random.	(9) Random.

Table 6.1 shows the experimental results of above mentioned combination. The timeout is set to 500s, and each time is the total of successful cases (either SAT or UNSAT).

Note that (10)-(7)-(9) means all random selection. Generally speaking, the combination of (5) and (8) show the best results, though the choice of (1),(2), and (10) shows different behavior on benchmarks. We tentatively prefer (1) or (10), but it needs to be investigated further.

Other than heuristics mentioned in Section **??**, there are lots of heuristic choices. For instance,

– how to generate test instances (in $U.T$),

– how to decompose an interval,

and so on.

Benchmark	(1)-(5)-(8)		(1)-(5)-(9)		(1)-(6)-(8)		(1)-(6)-(9)		(10)-(5)-(8)		(10)-(6)-(8)	
Matrix-1 (SAT)	20	132.72 (s)	21	21.48	19	526.76	18	562.19	21	462.57	19	155.77
Matrix-1 (UNSAT)	2	0.00	**3**	0.00	**3**	0.00	**3**	0	**3**	0.00	**3**	0.00
Matrix-2,3,4,5 (SAT)	**11**	632.37	1	4.83	0	0.00	1	22.50	9	943.08	1	30.48
Matrix-2,3,4,5 (UNSAT)	8	0.37	8	0.39	8	0.37	8	0.38	8	0.38	8	0.38

Benchmark	(2)-(5)-(8)		(2)-(5)-(9)		(2)-(6)-(8)		(2)-(6)-(9)		(2)-(7)-(8)		(10)-(7)-(9)	
Matrix-1 (SAT)	**22**	163.47 (s)	19	736.17	20	324.97	18	1068.40	21	799.79	21	933.39
Matrix-1 (UNSAT)	2	0	2	0.00	2	0.00	2	0.00	2	0.00	2	0.00
Matrix-2,3,4,5 (SAT)	5	202.37	1	350.84	1	138.86	0	0.00	0	0.00	0	0.00
Matrix-2,3,4,5 (UNSAT)	8	0.43	8	0.37	8	0.40	8	0.38	8	0.37	8	0.38

Benchmark	(1)-(3)-(8)		(1)-(4)-(8)		(2)-(3)-(8)		(2)-(4)-(8)		(10)-(3)-(8)		(10)-(4)-(8)	
Matrix-1 (SAT)	20	738.26 (s)	21	1537.9	18	479.60	21	867.99	20	588.78	19	196.21
Matrix-1 (UNSAT)	2	0.00	2	0.00	2	0.00	2	0.00	2	0.00	2	0.00
Matrix-2,3,4,5 (SAT)	0	0.00	2	289.17	1	467.12	1	328.03	1	195.18	2	354.94
Matrix-2,3,4,5 (UNSAT)	8	0.36	8	0.36	8	0.34	8	0.37	8	0.37	8	0.39

Benchmark	(1)-(5)-(8)		(1)-(5)-(9)		(10)-(5)-(8)		(10)-(7)-(9)	
Meti-Tarski (SAT, 3528)	3322	369.60 (s)	3303	425.37	**3325**	653.87	3322	642.04
Meti-Tarski (UNSAT, 1573)	1052	383.40	1064	1141.67	**1100**	842.73	1076	829.43

Table 6.1: Combnations of **raSAT** strategies on NRA/Zankl,Meti-Tarski benchmark

Experiments in Table 6.1 are performed with randome generation (k-random tick) for the former and the blanced decomposition (dividing at the exact middle) for the latter. Further investigation is left for future.

6.2 Comparison with other SMT Solvers

We compare **raSAT** with other SMT solvers on NRA benchmarks, Zankl and Meti-Tarski. The timeouts for Zankl and Meti-tarski are $500s$ and $60s$, respectively. For **iSAT3**, ranges of all variables are uniformly set to be in the range $[-1000, 1000]$ (otherwise, it often causes segmentation fault). Thus, UNSAT detection of **iSAT3** means UNSAT in the range $[-1000, 1000]$, while that of **raSAT** and **Z3 4.3** means UNSAT over $[-\infty, \infty]$.

Among these SMT solvers, **Z3 4.3** shows the best performance. However, if we closely observe, there are certain tendency. **Z3 4.3** is very quick for small constraints, i.e., with short APIs (up to 5) and a small number of variables (up to 10). **raSAT** shows comparable performance on SAT detection with longer APIs (larger than 5) and a larger number of variables (more than 10), and sometimes outforms for SAT detection on vary long constraints (APIs longer than 40 and/or more than 20 variables). Such examples appear in Zankl/matrix-3-all-*, matrix-4-all-*, and matrix-5-all-* (total 74 problems), and **raSAT** solely solves

- *matrix-3-all-2* (47 variables, 87 APIs, and max length of an API is 27),
- *matrix-3-all-5* (81 variables, 142 APIs, and max length of an API is 20),
- *matrix-4-all-3* (139 variables, 244 APIs, and max length of an API is 73), and

Benchmark	raSAT				Z3 4.3				iSAT3			
	SAT		UNSAT		SAT		UNSAT		SAT		UNSAT	
Zankl/matrix-1 (53)	20	132.72 (s)	2	0.00	41	2.17	12	0.00	11	4.68	3	0.00
Zankl/matrix-2,3,4,5 (98)	11	632.37	8	0.37	13	1031.68	11	0.57	3	196.40	12	8.06
Meti-Tarski (3528/1573)	3322	369.60	1052	383.40	3528	51.22	1568	78.56	2916	811.53	1225	73.83

Table 6.2: Comparison among SMT solvers

- *matrix-5-all-01* (132 variables, 276 APIs, and max length of an API is 47).

Note that, for Zankl, when UNSAT is detected, it is detected very quickly. This is because SMT solvers detects UNSAT only when they find small UNSAT cores, without tracing all APIs. However, for SAT detection with ;arge problems, SMT solvers need to trace all problems. Thus, it takes much longer time.

6.3 Experiments on Equality Handling

6.4 Polynomial Constraints over Integers

raSAT loop is easily modified to NIA (nonlinear arithmetic over integers) from NRA, by setting $\gamma_0 = 1$ in incremental deepening in Section 5.1 and restrcting testdata generation on intergers. We also compare **raSAT** (combination (1)-(5)-(8)) with **Z3 4.3** on NIA/AProVE benchmark. **AProVE** contains 6850 inequalities among 8829. Some has several hundred variables, but each API has few variables (mostly just 2 variables).

The results are,

- **raSAT** detects 6764 SAT in 1230.54s, and 0 UNSAT.

- **Z3 4.3** detects 6784 SAT in 103.70s, and 36 UNSAT in 36.08s.

where the timeout is $60s$. **raSAT** does not successfully detect UNSAT, since UNSAT problems have quite large coefficients which lead exhaustive search on quite large area.

Chapter 7

Equality handling

7.1 SAT on Equality by Intermediate Value Theorem

For solving polynomial constraints with single equality ($g = 0$), we apply *Intermediate Value Theorem*. That is, if existing 2 test cases such that $g > 0$ and $g < 0$, then $g = 0$ is SAT somewhere in between, as in Fig. **??**.

Lemma 7.1.1. *For* $F = \exists x_1 \in (a_1, b_1) \wedge \cdots \wedge x_n \in (a_n, b_n). \bigwedge_j^m f_j > 0 \ \wedge \ g = 0$, *F is SAT, if there is a box* $(l_1, h_1) \times \cdots \times (l_n, h_n)$ *with* $(l_i, h_i) \subseteq (a_i, b_i)$ *such that*

(i) $\bigwedge_j^m f_j > 0$ *is IA-VALID in the box, and*

(ii) there are two instances $\vec{t}, \vec{t'}$ *in the box with* $g(\vec{t}) > 0$ *and* $g(\vec{t'}) < 0$.

raSAT first tries to find an IA-VALID box for $\bigwedge_j^m f_j > 0$ by refinements. If such a box is found, it tries to find 2 instances for $g > 0$ and $g < 0$ by testing. Intermediate Value Theorem guarantees the existence of an SAT instance in between. Note that this method works for single equality and does not find an exact SAT instance. If multiple equalities do not share variables each other, we can apply Intermediate Value Theorem repeatedly to decide SAT. In Zankl benchmarks in SMT-lib, there are 15 gen-**.smt2 that contain equality (among 166 problems), and each of them satisty this condition.

In Table 7.1 we show preliminary experiment for 15 problems that contain polynomial equalities in Zankl family. **raSAT** works well for these SAT problems and it can detect all SAT problems (11 among 15). At the current implementation, raSAT reports *unknown* for UNSAT problems. The first 4 columns indicate *name of problems*, *the number of variables*, *the number of polynomial equalities* and *the number of inequalities* in each problem, respectively. The last 2 columns show comparison results of **Z3 4.3** and **raSAT**.

We also apply the same idea for multiple equalities $\bigwedge_i g_i = 0$ such that $Var(g_k) \cap Var(g_{k'}) = \emptyset$ where $Var(g_k)$ is denoted for the set of variables in the polynomial g_k. In

Problem Name	No. Variables	No. Equalities	No. Inequalities	**Z3 4.3** (15/15)		**raSAT** (11/15)	
				Result	Time(s)	Result	Time(s)
gen-03	1	1	0	SAT	0.01	SAT	0.015
gen-04	1	1	0	SAT	0.01	SAT	0.015
gen-05	2	2	0	SAT	0.01	SAT	0.046
gen-06	2	2	1	SAT	0.01	SAT	0.062
gen-07	2	2	0	SAT	0.01	SAT	0.062
gen-08	2	2	1	SAT	0.01	SAT	0.062
gen-09	2	2	1	SAT	0.03	SAT	0.062
gen-10	1	1	0	SAT	0.02	SAT	0.031
gen-13	1	1	0	UNSAT	0.05	unknown	0.015
gen-14	1	1	0	UNSAT	0.01	unknown	0.015
gen-15	2	3	0	UNSAT	0.01	unknown	0.015
gen-16	2	2	1	SAT	0.01	SAT	0.062
gen-17	2	3	0	UNSAT	0.01	unknown	0.031
gen-18	2	2	1	SAT	0.01	SAT	0.078
gen-19	2	2	1	SAT	0.05	SAT	0.046

Table 7.1: Experimental results for 15 equality problems of Zankl family

the next section we will present idea for solving general cases of multiple equalities.

7.2 Equality Handling using Grobner Basis

Coming soon...

Chapter 8

UNSAT core

Chapter 9

Conclusion

This paper presented **raSAT** loop, which mutually refines over and under–approximation theories. For polynomial inequlaity, we adopted interval arithemtic and testing for over and under-approximation theories, respectively. **raSAT** loop is implemented as an SMT **raSAT**. The result of Experiments on QF_NRA in SMT-lib is encouraging, and **raSAT** shows comparable and sometimes outperforming to existing SMTs, e.g., Z3 4.3, HySAT, and dReal. For instance, ****** which has ** variables and degree ** was solved by **raSAT**, whereas none of above mentioned SMTS can.

9.1 Observation and Discussion

From experimental results in Section **??** and **??**, we observe the followings.

- The degree of polynomials will not affect much.
- The number of variables are matters, but also for Z3 4.3. The experimental results do not show exponential growth, and we expect the strategy of selection of an API in which related intervals are decomposed seems effective in practice. By observing Zankl examples, we think the maximum number of variables of each API seems a dominant factor.
- Effects of the number of APIs are not clear at the moment. In simple benchmarks, **raSAT** is faster than Z3 4.3, however we admit that we have set small degree $n = 6$ for each API.

For instance, *matrix-2-all-5,8,11,12* in Zankl contain a long monomial (e.g., 60) with the max degree 6, and relatively many variables (e.g., 14), which cannot be solved by Z3 4.3, but **raSAT** does. As a general feeling, if an API contains more than 30 $\sim$ 40 variables, **raSAT** seems saturating. We expect that, adding to a strategy to select an API (Section **??**), we need a strategy to select variables in the focus. We expect this can be designed with sensitivity (Example **??**) and would work in practice. Note that sensitivity can be used only with noise symbols in Affine intervals. Thus, iSAT and RSOLVER cannot use this strategy, though they are based on IA, too.

9.2 Future Work

UNSAT core

Exact confirmation. Currently, **raSAT** uses iRRAM to verify SAT instances only. We are planning to implement confirmation phase to UNSAT cases.

Further strategy refinement. Test data generation, blanced box decomposition

Bibliography

[1] Hirokazu Anai. Algebraic methods for solving real polynomial constraints and their applications in biology. In *In Algebraic Biology Computer Algebra in Biology*, pages 139–147, 2005.

[2] GeorgeE. Collins. Quantifier elimination by cylindrical algebraic decomposition twenty years of progress. In BobF. Caviness and JeremyR. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Symbolic Computation, pages 8–23. Springer Vienna, 1998. ISBN 978-3-211-82794-9. doi: 10.1007/978-3-7091-9459-1_2. URL `http://dx.doi.org/10.1007/978-3-7091-9459-1_2`.

[3] MichaelA. Coln, Sriram Sankaranarayanan, and HennyB. Sipma. Linear invariant generation using non-linear constraint solving. In Jr. Hunt, WarrenA. and Fabio Somenzi, editors, *Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 420–432. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40524-5. doi: 10.1007/978-3-540-45069-6_39. URL `http://dx.doi.org/10.1007/978-3-540-45069-6_39`.

[4] Joo Luiz Dihl Comba and Jorge Stolfi. Affine arithmetic and its applications to computer graphics, 1993.

[5] Martin Frnzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. 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] M.K. Ganai and F. Ivancic. Efficient decision procedure for non-linear arithmetic constraints using cordic. In *Formal Methods in Computer-Aided Design, 2009. FMCAD 2009*, pages 61–68, Nov 2009. doi: 10.1109/FMCAD.2009.5351140.

[7] Sicun Gao, Soonho Kong, and EdmundM. Clarke. dreal: An smt solver for nonlinear theories over the reals. In MariaPaola Bonacina, editor, *Automated Deduction CADE-24*, volume 7898 of *Lecture Notes in Computer Science*, pages 208–214. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38573-5. doi: 10.1007/978-3-642-38574-2_14. URL `http://dx.doi.org/10.1007/978-3-642-38574-2_14`.

[8] T. Hickey, Q. Ju, and M. H. Van Emden. Interval arithmetic: From principles to implementation. *J. ACM*, 48(5):1038–1068, September 2001. ISSN 0004-5411. doi: 10.1145/502102.502106. URL `http://doi.acm.org/10.1145/502102.502106`.

[9] Dejan Jovanovi and Leonardo de Moura. Solving non-linear arithmetic. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning*, volume 7364 of *Lecture Notes in Computer Science*, pages 339–354. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-31364-6. doi: 10.1007/978-3-642-31365-3_27. URL `http://dx.doi.org/10.1007/978-3-642-31365-3_27`.

[10] To Van Khanh and Mizuhito Ogawa. {SMT} for polynomial constraints on real numbers. *Electronic Notes in Theoretical Computer Science*, 289(0):27 – 40, 2012. ISSN 1571-0661. doi: http://dx.doi.org/10.1016/j.entcs.2012.11.004. URL `http://www.sciencedirect.com/science/article/pii/S1571066112000746`. Third Workshop on Tools for Automatic Program Analysis (TAPAS' 2012).

[11] To Van Khanh and Mizuhito Ogawa. rasat: Smt for polynomial inequality. Technical report, School of Information Science, Japan Advanced Institute of Science and Technology, 05 2013.

[12] Salvador Lucas and Rafael Navarro-Marset. Comparing csp and sat solvers for polynomial constraints in termination provers. *Electron. Notes Theor. Comput. Sci.*, 206:75–90, April 2008. ISSN 1571-0661. doi: 10.1016/j.entcs.2008.03.076. URL `http://dx.doi.org/10.1016/j.entcs.2008.03.076`.

[13] Frdric Messine. Extensions of affine arithmetic: Application to unconstrained global optimization.

[14] Do Thi Bich Ngoc and Mizuhito Ogawa. Overflow and roundoff error analysis via model checking. In *Proceedings of the 2009 Seventh IEEE International Conference on Software Engineering and Formal Methods*, SEFM '09, pages 105–114, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3870-9. doi: 10.1109/SEFM.2009.32. URL `http://dx.doi.org/10.1109/SEFM.2009.32`.

[15] Do Thi Bich Ngoc and Mizuhito Ogawa. Checking roundoff errors using counterexample-guided narrowing. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 301–304, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0116-9. doi: 10.1145/1858996.1859056. URL `http://doi.acm.org/10.1145/1858996.1859056`.

[16] Anh-Dung Phan, Nikolaj Bjørner, and David Monniaux. Anatomy of alternating quantifier satisfiability (work in progress). In *10th International Workshop on Satisfiability Modulo Theories (SMT)*, 2012.

[17] Stefan Ratschan. Efficient solving of quantified inequality constraints over the real numbers. *ACM Trans. Comput. Logic*, 7(4):723–748, October 2006. ISSN

1529-3785. doi: 10.1145/1183278.1183282. URL http://doi.acm.org/10.1145/1183278.1183282.

[18] Sriram Sankaranarayanan, Henny Sipma, and Zohar Manna. Constructing invariants for hybrid systems. In *in Hybrid Systems: Computation and Control, LNCS 2993*, pages 539–554. Springer-Verlag, 2004.

[19] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Non-linear loop invariant generation using gröbner bases. *SIGPLAN Not.*, 39(1):318–329, January 2004. ISSN 0362-1340. doi: 10.1145/982962.964028. URL http://doi.acm.org/10.1145/982962.964028.

[20] Jorge Stolfi and Luiz Henrique De Figueiredo. Self-validated numerical methods and applications, 1997.

[21] Alfred Tarski. A decision method for elementary algebra and geometry. In BobF. Caviness and JeremyR. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts and Monographs in Symbolic Computation, pages 24–84. Springer Vienna, 1998. ISBN 978-3-211-82794-9. doi: 10.1007/978-3-7091-9459-1_3. URL http://dx.doi.org/10.1007/978-3-7091-9459-1_3.

[22] Harald Zankl and Aart Middeldorp. Satisfiability of non-linear (ir)rational arithmetic. In *Proceedings of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, LPAR'10, pages 481–500, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-17510-4, 978-3-642-17510-7. URL http://dl.acm.org/citation.cfm?id=1939141.1939168.