# 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
Professor Mizuhito Ogawa
Associate Professor Nao Hirokawa
Professor Tachio Terauchi

February, 2015 (Submitted)

# Abstract

Solving polynomial constraints is raised from many applications of Software Verification such as round-off/over-flow error analysis, automatic termination proving or loop invariant generation. Although in 1948, Tarski proved the decidability of polynomial constraints over real numbers, the current complete method named Quantifier Elimination by Cylindrical Algebraic Decomposition has the complexity of doubly-exponential with respect to the number of variables which remains as an impediment. Interval Constraint Propagation (ICP) which uses the inequalities/equations to contract the interval of variables by removing the unsatisfiable intervals is an efficient methodology because it uses floating point arithmetic. However the number of boxes (combination of intervals of variables) may grow exponentially.

This thesis presents strategies for efficiency improvement and extensions of an SMT solver named **raSAT** for polynomial constraints. **raSAT** which initially focuses on polynomial inequalities over real numbers follows ICP methodology and adds testing to boost satisfiability detection. In this work, in order to deal with exponential exploration of boxes, several heuristic measures, namely *SAT likelyhood*, *sensitivity*, and *the number of unsolved polynomial inequalities*, are proposed. From the experiments on standard SMT-LIB benchmarks, **raSAT** is able to solve large constraints (in terms of the number of variables) which are difficult for other tools. In addition to those heuristics, extensions for handling equations using the Intermediate Value Theorem and handling constraints over integer number are also presented in this thesis. The contributions of this work are as follows:

1. Because the number of boxes (products of intervals) grows exponentially with respect to the number of variables during refinement (interval decomposition), strategies for *selecting one variable* to decomposed and *selecting one box* to explore play a crucial role in efficiency. We introduce the following strategies:
   - **Selecting one box.** The box with more possiblity to satisfy the constraint is selected to explore, which is estimated by several heuristic measures, called *SAT likelyhood*, and *the number of unsolved polynomial inequalities*.
   - **Selecting one variable.** The most influential variable is selected as priority in approximation and refinement process. This is estimated by *sensitivity* which is determined during the approximation process.
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 exhaustive search, it starts searching with a threshold that each interval will be decomposed no smaller than it. If neither satisfiability nor unsatisfiability is detected, a smaller threshold is taken and raSAT restarts.
- **Incremental widening**. Starting with a small intervals, if **raSAT** detects UNSAT, it enlarges input intervals and restarts. This strategy is effective in detecting satisfiability of constraints because small intervals reduce the number of boxes after decomposition.

3. *Satisfiability confirmation* step by an error-bound guaranteed floating point package **iRRAM**[2], to avoid soundess bugs caused by roundoff errors.
4. This work also implemented the idea of using Intermediate Value Theorem to show *the satisfiability of multiple equations* which was suggested in [10].
5. **raSAT** is also extended to *handle constraints over integer numbers* by simple extension in the approximation process.

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

# Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Mizuhito Ogawa for the continuous support of not only my thesis but also my life. He instructed me how to critically think about and evaluate research.

My sincere thanks also goes to Associate Professor Nao Hirokawa for his detailed and constructive comments and also for his important support through this work. I learned a lot from him how to write scientific documents.

I owe my loving thanks to my parents, my older brother, my older sister and my girl friend. Without their encouragement and understanding it would have been impossible for me to finish this work.

# Contents

# Chapter 1

# Introduction

This chapter is going to introduce about polynomial constraints and present the overview of our approach to handle them. 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., TTT2[1], AProVE[2], that leads to polynomial constraints.
- **Loop invariant generation** [2, 19] is reduced to solving polynomial constraints over coefficients of invariant template.

## 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 constraint $x^2 + y^2 < 1 \land xy > 1$ is an example of an unsatisfiable one. While the set of satisfiable points for the first inequality ($x^2 + y^2 < 1$) forms the read circle in Figure 1.1, that for the second forms the blue area. Because these two areas do not intersect, the conjunction of two inequalities is unsatisfiable.*

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

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

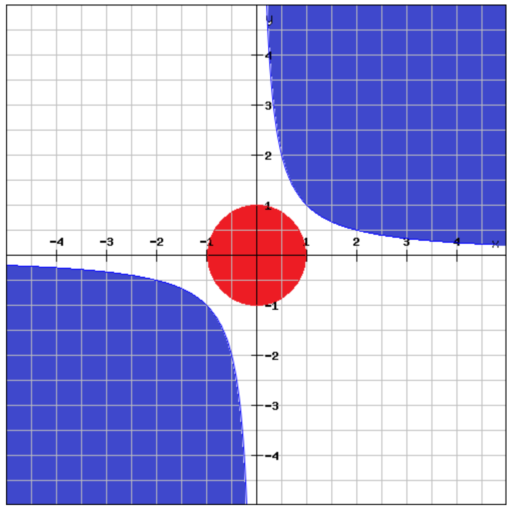


Figure 1.1: Example of UNSAT constraint

## 1.2 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.

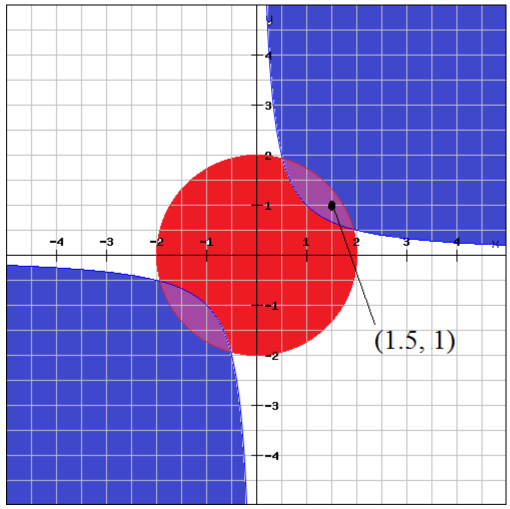



Figure 1.2: Example of SAT constraint

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 and constraints over integer numbers. 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 of real numbers to variables 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 one variable* to decomposed and *selecting one box* to explore play a crucial role in efficiency. We introduce the following strategies:

- **Selecting one box.** The box with more possibility to satisfy the given constraint is selected to explore, which is estimated by several heuristic measures, called *SAT likelihood*, and *the number of unsolved polynomial inequalities*.
- **Selecting one variable.** The most 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 strategy 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 **iRRAM**[3], to avoid soundess bugs caused by roundoff errors.
4. This work also implemented the idea of using Intermediate Value Theorem to show *the satisfiability of multiple equations* which was suggested in [10].
5. We also extend **raSAT** to *handle constraints 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.

## 1.3 Thesis Outline

The rest of this thesis is organized as following.

- Chapter 2 presents the basics about abstract Davis-Putnam-Logemann-Loveland (DPLL) procedure for solving propositional formulas, basics about Satisfiability Module Theories (SMT) and SMT for polynomial constraints over real numbers.
- Chapter 3 introduces proposed approximation schemes and their instances, **raSAT** loop algorithm and its soundness, completeness.
- Variants of Interval Arithmetic which is an instance of over-approximation are presented in Chapter 4.

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

- Chapter 5 illustrates strategies for improving efficiency of **raSAT** loop.
- Experiments to show how efficient are proposed strategies and comparison with other SMT solvers are presented in Chapter 6.
- Chapter 7 proposes extensions for handling equations and constraints over integer numbers.
- Before summarizing the thesis and suggesting future works in Chapter 9, we present related works in Chapter 8.

# Chapter 2

# Preliminaries

## 2.1 Abstract DPLL

In propositional logic, we have a set of *propositional symbols* $P$ and every $p \in P$ is called an *atom*. A *literal* $l$ is either $p$ or $\neg p$ with $p \in P$. The *negation* $\neg l$ of a literal $l$ is $\neg p$ if $l$ is $p$, or $p$ if $l$ is $\neg p$. A disjunction $l_1 \vee \cdots \vee l_n$ of literals is said to be a *clause*. A *Conjuctive Normal Form (CNF) formula* is a conjunction of clauses $C_1 \wedge \cdots \wedge C_n$. If $C = l_1 \vee \cdots \vee l_n$ is a clause, $\neg C$ is used to denote the CNF formula $\neg l_1 \wedge \cdots \wedge \neg l_n$

An (partial) *assignment* $M$ is a set of literals such that $l \in M$ and $\neg l \in M$ for no literal $l$. A literal $l$ is undefined in $M$ if neither $l \in M$ nor $\neg l \in M$. If $l \in M$, $l$ is said to be true in M. On the other hand if $\neg l \in M$, we say that $l$ is false in $M$. A clause is true in $M$ if at least one of its literals is in $M$. An assignment $M$ satisfies a CNF formula $F$ (or $F$ is satisfied by $M$) if all clauses of $F$ is true in $M$ which is denoted as $M \models F$. Given two CNF formula $F$ and $F'$, we write $F \models F'$ if for any assignment $M$, $M \models F$ implies $M \models F'$. The formula $F$ is unsatisfiable if there is no assignment $M$ such that $M \models F$.

Abstract Davis-Putnam-Logemann-Loveland (DPLL) Procedure [16] searches for an assignment that satisfies a CNF formula. Each state of the procedure is either *FailState* or a pair $M \parallel F$ of an assignment $M$ and a CNF formula $F$. For the purpose of the procedure, $M$ is represented as a sequence of literals where each literal is optionally attached an *annotation*, e.g. $l^d$ which basically means that the literal $l$ is selected to be included in the assignment by making a decision ($l$ is called a decision literal). An empty sequence is denoted by $\emptyset$. Each DPLL procedure is modeled by a collection of states and a binary relation $\Longrightarrow$ between states. Basic DPLL procedure is a transition system which contains the following four rules.

1. **UnitPropagate**

$$M \parallel F \wedge (C \vee l) \Longrightarrow Ml \parallel F \wedge (C \vee l) \text{ if } \begin{cases} M \models \neg C \\ l \text{ is undefined in } M. \end{cases}$$

2. **Decide**

$$M \parallel F \Longrightarrow Ml^d \parallel F \text{ if } \begin{cases} l \text{ or } \neg l \text{ occur in a clause of F} \\ l \text{ is undefined in } M. \end{cases}$$

3. **Fail**

$$M \parallel F \wedge C \Longrightarrow FailState \text{ if } \begin{cases} M \models \neg C \\ l^d \in M \text{ for no literal } l. \end{cases}$$

4. **Backjump**

$$Ml^dM' \parallel F \wedge C \Longrightarrow Ml' \parallel F \wedge C \text{ if } \begin{cases} Ml^dM' \models \neg C, \text{ and there is some clause } C' \vee l' \\ \text{such that } F \wedge C \models C' \vee l' \text{ and } M \models \neg C', l' \text{ is} \\ \text{undefined in } M, \text{ and } l' \text{ or } \neg l' \text{ occurs in } F \text{ or in} \\ Ml^dM'. \end{cases}$$

Let $F$ be a given CNF formula. Starting with the state $\emptyset \parallel F$, basic DPLL procedure terminates with either $FailState$ (which is denoted as $\emptyset \parallel F \Longrightarrow^! FailState$) when $F$ is unsatisfiable, or a state $M \parallel F$ (which is denoted as $\emptyset \parallel F \Longrightarrow^! M \parallel F$) where $M$ satisfies $F$. Intuitively, the above four rules can be explained as following.

- **UnitPropagate**: In order to satisfy $F \wedge (C \vee l)$, $C \vee l$ needs to be satisfied. Because all the literals in $C$ is false in current assignment $M$ ($M \models \neg C$), $l$ must be made true when extending $M$.

- **Decide**: This rule is applied when no more **UnitPropagation** can be applied. The annotation $d$ in $l^d$ denotes that if $Ml$ cannot be extended to satisfy $f$, $M\neg l$ needs to be explored further.

- **Fail**: When a clause is false in $M$ (conflict) and $M$ has no literal which is decided by making a decision (there is no more options to explored), the formula $F$ is unsatisfiable.

- **Backjump**: As same as in **Fail** rule, a conflict is detected. However, in **Backjump**, because there exists some decision literal in the assignment, new possible assignments can be explored. The clause $C' \vee l'$ is called the backjump clause.

**Example 2.1.1.** *For the formula* $(\neg l_1 \vee l_2) \wedge (\neg l_3 \vee l_4) \wedge (\neg l_5 \vee \neg l_6) \wedge (l_6 \vee \neg l_5 \vee \neg l_2)$, *the basic DPLL procedure proceeds as following:*

$$\begin{aligned}
\emptyset \parallel (\neg l_1 \vee l_2) \wedge (\neg l_3 \vee l_4) \wedge (\neg l_5 \vee \neg l_6) \wedge (l_6 \vee \neg l_5 \vee \neg l_2) &\Longrightarrow (\textbf{Decide}) \\
l_1^d \parallel (\neg l_1 \vee l_2) \wedge (\neg l_3 \vee l_4) \wedge (\neg l_5 \vee \neg l_6) \wedge (l_6 \vee \neg l_5 \vee \neg l_2) &\Longrightarrow (\textbf{UnitPropagate}) \\
l_1^d l_2 \parallel (\neg l_1 \vee l_2) \wedge (\neg l_3 \vee l_4) \wedge (\neg l_5 \vee \neg l_6) \wedge (l_6 \vee \neg l_5 \vee \neg l_2) &\Longrightarrow (\textbf{Decide}) \\
l_1^d l_2 l_3^d \parallel (\neg l_1 \vee l_2) \wedge (\neg l_3 \vee l_4) \wedge (\neg l_5 \vee \neg l_6) \wedge (l_6 \vee \neg l_5 \vee \neg l_2) &\Longrightarrow (\textbf{UnitPropagate}) \\
l_1^d l_2 l_3^d l_4 \parallel (\neg l_1 \vee l_2) \wedge (\neg l_3 \vee l_4) \wedge (\neg l_5 \vee \neg l_6) \wedge (l_6 \vee \neg l_5 \vee \neg l_2) &\Longrightarrow (\textbf{Decide}) \\
l_1^d l_2 l_3^d l_4 l_5^d \parallel (\neg l_1 \vee l_2) \wedge (\neg l_3 \vee l_4) \wedge (\neg l_5 \vee \neg l_6) \wedge (l_6 \vee \neg l_5 \vee \neg l_2) &\Longrightarrow (\textbf{UnitPropagate}) \\
l_1^d l_2 l_3^d l_4 l_5^d l_6 \parallel (\neg l_1 \vee l_2) \wedge (\neg l_3 \vee l_4) \wedge (\neg l_5 \vee \neg l_6) \wedge (l_6 \vee \neg l_5 \vee \neg l_2) &\Longrightarrow (\textbf{Backjump}) \\
l_1^d l_2 l_3^d l_4 \neg l_5 \parallel (\neg l_1 \vee l_2) \wedge (\neg l_3 \vee l_4) \wedge (\neg l_5 \vee \neg l_6) \wedge (l_6 \vee \neg l_5 \vee \neg l_2) &
\end{aligned}$$

*In the* **Bacjump** *step of this example:* $M$ *is* $l_1^d l_2 l_3^d l_4$, $l^d$ *is* $l_5^d$, $M'$ *is* $l_6$, $F$ *is* $(\neg l_1 \vee l_2) \wedge (\neg l_3 \vee l_4) \wedge (\neg l_5 \vee \neg l_6)$, $C$ *is* $l_6 \vee \neg l_5 \vee \neg l_2$, $C'$ *is* $\neg l_1$, *and* $l'$ *is* $\neg l_5$.

Additionally DPLL implementation can add the backjump clauses into the CNF formula as learnt clause (or lemmas), which is usually referred as *conflict-driven learning*. Lemmas aim at preventing similar conflicts to occur in the future. When the conflicts are not likely to happen, the lemmas can be removed. The following two rules are prepared for DPLL.

6. **Learn**

$$M \parallel F \Longrightarrow M \parallel F \wedge C \text{ if } \begin{cases} \text{each atom in C appears in F} \\ F \models C. \end{cases}$$

7. **Forget**

$$M \parallel F \wedge C \Longrightarrow M \parallel F \text{ if } \begin{cases} \text{each atom in C appears in F} \\ F \models C. \end{cases}$$

## 2.2 Satisfiability Modulo Theories - SMT

### 2.2.1 Syntax

**Definition 2.2.1.** *A* signature $\Sigma$ *is a 4-tuple* $(S, P, F, V, \alpha)$ *consisting of a set* $S$ *of* sorts*, a set* $P$ *of* predicate symbols*, a set* $F$ *of* function symbols*, a set* $V$ *of* variables*, and a* sorts map $\alpha$ *which associates symbols to their sorts such that*

- *for all* $p \in P$*,* $\alpha(p)$ *is a* $n$*-tuple argument sorts of* $p$*,*
- *for all* $f \in F$*,* $\alpha(f)$ *is a* $n$*-tuple of argument and returned sorts of* $f$*, and*
- *for all* $v \in V$*,* $\alpha(v)$ *represents the sort of variable* $v$*.*

A $\Sigma$-*term* $t$ over the signature $\Sigma$ is defined as

$$\begin{array}{lcll} t & ::= & v & \text{where } v \in V \\ & | & f(t_1, \cdots, t_n) & \text{where } f \in F \text{ with arity } n \end{array}$$

A $\Sigma$-*formula* $\varphi$ over the signature $\Sigma$ is defined recursively as (we only focus on equantifier-free formulas):

$$\begin{array}{lcll} \varphi & ::= & p(t_1, \cdots, t_n) & \text{where } p \in P \text{ with arity } n \\ & | & \bot \mid \neg\varphi_1 & \\ & | & \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 & \\ & | & \varphi_1 \rightarrow \varphi_2 \mid \varphi_1 \leftrightarrow \varphi_2 & \end{array}$$

### 2.2.2 Semantics

**Definition 2.2.2.** *Let* $\Sigma = (S, P, F, V, \alpha)$ *is a signature. A* $\Sigma$-model $M$ *of* $\Sigma$ *is a pair* $(U, I)$ *in which* $U$ *is the* universe *and* $I$ *is the* interpretation *of symbols such that*

- *for all* $s \in S$*,* $I(s) \subseteq U$ *specifies the possible values of sort* $s$*,*

- *for all $f \in F$, $I(f)$ is a function from $I(s_1) \times \cdots \times I(s_{n-1})$ to $I(s_n)$ with $\alpha(f) = (s_1, \cdots, s_n)$,*
- *for all $p \in P$, $I(p)$ is a function from $I(s_1) \times \cdots \times I(s_n)$ to $\{0, 1\}$ where $\alpha(p) = (s_1, \cdots, s_n)$, and*
- *for all $v \in V$, $I(v) \in I(\alpha(v))$*

*A* $\Sigma$-theory *$T$ is a (infinite) set of $\Sigma$-models. A theory $T'$ is a subset of theory $T$ if and only if $T' \subseteq T$.*

The interpretation of each predicate or function symbol is allowed to be not total, i.e. $I(p)$ or $I(f)$ are not necessarily total. We extend the universe of each model to include the symbol $\mathring{u}$ (unknown) which is prepared 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 formulas containing logical connectives ($\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, or $\neg$).

**Definition 2.2.3.** *Let $\Sigma = (S, P, F, V, \alpha)$ and $M = (U, I)$ are a signature and a $\Sigma-$model respectively. The* valuation *of a $\Sigma$-term $t$ against $M$ which is denoted by $t^M$ is defined recursively as:*

$$
\begin{array}{lll l}
v^M & = & I(v) & \text{where } v \in V, \text{ and} \\
f^M(t_1, \cdots, t_n) & = & \begin{cases} I(f)(t_1^M, \cdots, t_n^M) & \text{if } (t_1^M, \cdots, t_n^M) \in Dom(I(f)) \\ \mathring{u} & \text{otherwise} \end{cases} & \text{where } f \in F.
\end{array}
$$

*Similarly, the valuation $\varphi^M$ of $\varphi$ is defined as:*

$$
\begin{array}{lll l}
p^M(t_1, \cdots, t_n) & = & \begin{cases} I(p)(t_1^M, \cdots, t_n^M) & \text{if } (t_1^M, \cdots, t_n^M) \in Dom(I(p)) \\ \mathring{u} & \text{otherwise} \end{cases} & \text{where } p \in P, \\
\perp^M & = & 0\ , & \\
(\neg\varphi_1)^M & = & 1 - \varphi_1^M\ , & \\
(\varphi_1 \wedge \varphi_2)^M & = & \min(\varphi_1^M, \varphi_2^M), & \\
(\varphi_1 \vee \varphi_2)^M & = & \max(\varphi_1^M, \varphi_2^M), & \\
(\varphi_1 \rightarrow \varphi_2)^M & = & \max(1 - \varphi_1^M, \varphi_2^M), \text{ and} & \\
(\varphi_1 \rightarrow \varphi_2)^M & = & \begin{cases} 1 & \text{if } \varphi_1^M = \varphi_2^M \\ 0 & \text{otherwise} \end{cases}. &
\end{array}
$$

*We say that $M$ satisfies $\varphi$ 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 $\varphi$.*

Given a signature $\Sigma$, a $\Sigma$-theory $T$ and a $\Sigma$-formula $\varphi$, a Satisfiability Modulo Theories(SMT) problem is the task of finding a model $M \in T$ such that $\models_M \varphi$.

**Lemma 2.2.1.** *Given any $\Sigma$-model $M$ and $\Sigma$-formula $\varphi$, we have $\models_M \varphi$ if and only if $\not\models_M \neg\varphi$*

*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 2.2.4.** *Let $T$ be a $\Sigma$-theory. A $\Sigma$-formula $\varphi$ is:*

- satisfiable *in $T$ or $T$-SAT if and only if for all $M \in T$ we have $\models_M \varphi$,*
- valid *in $T$ or $T$-VALID if and only if for all $M \in T$ we have $\models_M \varphi$,*
- unsatisfiable *in $T$ or $T$-UNSAT if and only if for all $M \in T$ we have $\not\models_M \varphi$, and*
- unknown *in $T$ or $T$-UNKNOWN if and only if for all $M \in T$, $\varphi^M = \mathring{u}$.*

**Lemma 2.2.2.** *If $T$ be a $\Sigma$-theory, then $\varphi$ is $T$-VALID if and only if $\neg\varphi$ is $T$-UNSAT*

*Proof.* $\varphi$ 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 $\varphi$ is $T'$-SAT implies that $\varphi$ is $T$-SAT.*

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

## 2.3 Polynomial Constraints over Real Numbers

### 2.3.1 Syntax

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

- $S^p = \{Real\}$
- $P^p = \{\succ, \prec, \succeq, \preceq, \approx, \not\approx\}$
- $F^p = \{\oplus, \ominus, \otimes, \mathbf{1}\}$
- for all $p \in P^p$, $\alpha^p(p) = (Real, Real)$
- for all $f \in F^p \setminus \{\mathbf{1}\}$, $\alpha^p(f) = (Real, Real, Real)$ and $\alpha^p(\mathbf{1}) = Real$
- for all $v \in V$, $\alpha^p(v) = Real$

A polynomial and a polynomial constraint are a $\Sigma^p$-term (we referred as letters $f$ or $g$) and a $\Sigma^p$-formula respectively.

**Definition 2.3.1.** *Given a polynomial $f$, the set of its variables which is denoted as $var(f)$ is defined recursively as following:*

1. *$var(v) = \{v\}$ for $v \in V$.*
2. *$var(\mathbf{1}) = \emptyset$.*
3. *$var(f_1 \circ f_2) = var(f_1) \cap var(f_2)$ with $\circ \in \{\oplus, \ominus, \otimes\}$.*

### 2.3.2 Semantics

A model $M^p_{\mathbb{R}} = (\mathbb{R}, I^p_{\mathbb{R}})$ over real numbers for polynomial constraints 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. $\forall p \in P$; $I^p_{\mathbb{R}}(p)$ is a function from $\mathbb{R} \times \mathbb{R}$ to $\{1, 0\}$ 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. $\forall f \in F \setminus \{\mathbf{1}\}$; $I^p_{\mathbb{R}}(f)$ is a function from $\mathbb{R} \times \mathbb{R}$ to $\mathbb{R}$ 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 valuation of polynomials ($\Sigma^p$-terms) and polynomial constraints ($\Sigma^p$-formulas) against a model $M^p_{\mathbb{R}}$ follows Definition 2.2.3.

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, e.g. $\{v \mapsto r \in \mathbb{R} | v \in V\}$, can be used to represent a model. Given a map $\theta = \{v \mapsto r \in \mathbb{R} | v \in V\}$, $\theta^p_{\mathbb{R}}$ denotes the model represented by $\theta$.

Moreover, because all the predicate and function symbols' interpretations are total functions, so a given polynomial constraint $\varphi$ cannot be $T^p_{\mathbb{R}}$-UNKNOWN. In the other words, $\varphi$ can only be $T^p_{\mathbb{R}}$-SAT, $T^p_{\mathbb{R}}$-VALID, or $T^p_{\mathbb{R}}$-UNSAT.

From now on, we focus on $\Sigma^p$-formulas $\varphi$ of the forms:

$$\begin{array}{rcll} \varphi & ::= & p(f_1, f_2) & \text{where } p \in P \\ & | & \varphi_1 \wedge \varphi_2 & \end{array}$$

because this does not lose the generality. Given a general polynomial constraint $\varphi$ that is formed by the syntax in Section 2.2.1:

- If we consider each formula $p(f_1, f_2)$ as an propositional symbols, we can first convert $\varphi$ into an CNF formula and then use DPLL procedure to infer a sequence of literals that satisfies $\varphi$ (in terms of propositional logic).

- The sequence of literals may contains some literals of the form $\neg p(f_1, f_2)$. However, from the semantics of polynomial constraints, we can change:

$$\begin{array}{lcl} \neg(\succ (f_1, f_2)) & \text{to} & \preceq (f_1, f_2) \\ \neg(\prec (f_1, f_2)) & \text{to} & \succeq (f_1, f_2) \\ \neg(\succeq (f_1, f_2)) & \text{to} & \prec (f_1, f_2) \\ \neg(\preceq (f_1, f_2)) & \text{to} & \succ (f_1, f_2) \\ \neg(\approx (f_1, f_2)) & \text{to} & \not\approx (f_1, f_2) \\ \neg(\not\approx (f_1, f_2)) & \text{to} & \approx (f_1, f_2) \end{array}$$

- The remaining task is solving the SMT problem with the constraint is the conjunction of literals in the sequence.

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

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

1. $S^I = \{Real, Interval\}$,
2. $P^I = \{\in\}$,
3. $F^I = \{c \mid c \text{ is a constant}\}$,
4. $\alpha^I(\in) = (Real, Interval)$,
5. for all $c \in F^I$, $\alpha^I(c) = Interval$, and
6. $\forall v \in V;\ \alpha^I(v) = Real$.

We call $\Sigma^I$-formula is an interval constraint. The interval constraints in this thesis is represented by symbol $\Pi$ with possibly subscription. A model $M^I_{\mathbb{R}} = (\mathbb{I} \cup \mathbb{R}, I^I_{\mathbb{R}})$ of real intervals consists of the union of real intervals $\mathbb{I}$ which is defined later in Definition 3.2.2 and real numbers $\mathbb{R}$ and a map $I^I_{\mathbb{R}}$ that satisfies the following properties.

1. $I^I_{\mathbb{R}}(Real) = \mathbb{R}$ and $I^I_{\mathbb{R}}(Interval) = \mathbb{I}$.
2. $I^I_{\mathbb{R}}(\in) = \mathbb{R} \times \mathbb{I} \mapsto \{1, 0\}$ such that

$$I^I_{\mathbb{R}}(\in)(r, \langle a, b \rangle) = \begin{cases} 1 & \text{if } a \leq r \leq b \\ 0 & \text{otherwise} \end{cases}$$

3. For all $c \in F^I$, $I^I_{\mathbb{R}}(c) \in \mathbb{I}$.
4. For all $v \in V$, $I^I_{\mathbb{R}}(v) \in \mathbb{R}$.

The valuation of $\Sigma^{\mathbb{I}}$-terms and $\Sigma^{\mathbb{I}}$-terms follows Definition 2.2.3. The theory of real intervals is $T^I_{\mathbb{R}} = \{M^I_{\mathbb{R}} | M^I_{\mathbb{R}} \text{ is a model of real intervals}\}$. 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, e.g. $\{v \mapsto r \in \mathbb{R} | v \in V\}$, can be used to represent a model. Given an assignment $\theta = \{v \mapsto r \in \mathbb{R} | v \in V\}$, we denote $\theta^I_{\mathbb{R}}$ as a model of real intervals represented by $\theta$. If $\Pi$ is a $\Sigma^{\mathbb{I}}$-formula, the notation $\Pi^p_{\mathbb{R}} = \{\theta^p_{\mathbb{R}} \mid \theta = \{v \mapsto r \in \mathbb{R} \mid v \in V\} \text{ and } \models_{\theta^I_{\mathbb{R}}} \Pi\}$ represents the (sub-)theory of real numbers that each of its model contain the assignment from real numbers to variables that (intuitively) satisfies $\Pi$.

# Chapter 3

# Over-Approximation and Under-Approximation

## 3.1 Approximation Theory

**Definition 3.1.1.** *Let $T, T'$ be $\Sigma$-theories and $\varphi$ be any $\Sigma$-formula.*

- *$T'$ is an* over-approximation theory *of $T$ iff $T'$-UNSAT of $\varphi$ implies $T$-UNSAT of $\varphi$.*
- *$T'$ is an* under-approximation theory *of $T$ iff $T'$-SAT of $\varphi$ implies $T$-SAT of $\varphi$.*

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

*Proof.* $\varphi$ 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) $\implies$ $\varphi$ is T-VALID (Lemma 2.2.2) □

## 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 $a \leq b$.*

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

| $x+y$ | $-\infty$ | 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 | $\mathbb{R}$ | $+\infty$ |
| $+\infty$ | $-\infty$ | $-\infty$ | $-\infty$ | $-\infty$ | $\perp$ |

| $x*y$ | $-\infty$ | NR | 0 | PR | $+\infty$ |
|---|---|---|---|---|---|
| $-\infty$ | $+\infty$ | $+\infty$ | $\perp$ | $-\infty$ | $-\infty$ |
| $NR$ | | PR | 0 | NR | $-\infty$ |
| 0 | | | 0 | 0 | $\perp$ |
| $PR$ | | | | PR | $+\infty$ |
| $+\infty$ | | | | | $+\infty$ |

Table 3.1: Arithmetics Operations for $\mathbb{R} \cup \{-\infty, +\infty\}$

The *intervals* in this definition can be summarized by $\langle a, b\rangle$ where $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$ and $a \leq b$ 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 3.1.

**Definition 3.2.2.** *The set of all* real intervals $\mathbb{I}$ *is defined as:*

$$\mathbb{I} = \{\langle a, b\rangle \mid a, b \in \mathbb{R} \cup \{-\infty, +\infty\} \text{ and } a \leq b\}$$

.

### 3.2.2 Interval Arithmetic as an Over-Approximation Theory

A model $M^p_{IA} = (\mathbb{I}, I^p_{IA})$ over intervals for polynomial constraints consists a set of all real intervals $\mathbb{I}$ and a map $I^p_{IA}$ that satisfies the following conditions.

1. $I^p_{IA}(Real) = \mathbb{I}$
2. For all $p \in P^p$, $I^p_{IA}(p)$ is a function from $\mathbb{I} \times \mathbb{I}$ to $\{0, 1\}$ where

$$I^p_{IA}(p)(i_1, i_2) = i_1 \; p_{IA} \; i_2$$

The definition of $p_{IA}$ is as follow:

$$
\begin{array}{rcl}
\langle l_1, h_1 \rangle \succ_{IA} \langle l_2, h_2 \rangle & = & \begin{cases} 1 & \text{if } l_1 > h_2 \\ 0 & \text{if } h_1 \leq l_2 \end{cases} \\
\langle l_1, h_1 \rangle \prec_{IA} \langle l_2, h_2 \rangle & = & \begin{cases} 1 & \text{if } h_1 < l_2 \\ 0 & \text{if } l_1 \geq h_2 \end{cases} \\
i_1 \succeq_{IA} i_2 & = & 1 - (i_1 \prec_{IA} i_2) \\
i_1 \preceq_{IA} i_2 & = & 1 - (i_1 \succ_{IA} i_2) \\
i_1 \approx_{IA} i_2 & = & min(i_1 \succeq_{IA} i_2, i_1 \preceq_{IA} i_2) \\
i_1 \not\approx_{IA} i_2 & = & 1 - (i_1 \approx_{IA} i_2)
\end{array}
$$

3. For all $f \in F^p \setminus \{\mathbf{1}\}$, $I^p_{IA}(f)$ is a function from $\mathbb{I} \times \mathbb{I} \mapsto \mathbb{I}$ 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. For all $v \in V$; $I^p_{IA} \in U^p_{IA}$

Theory $T^p_{IA} = \{M^p_{IA} | M^p_{IA}$ 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. In addition, an assignment $\{v \mapsto i \in \mathbb{I} | v \in V\}$ and an interval constraint $\bigwedge_{v \in V} v \in i$ are equivalent in terms of the set of assignments from variables to real numbers. So by abusing notation, for a constraint of the form $\Pi = \bigwedge_{v \in V} v \in i$, we denote $\Pi^p_{IA}$ as a model of interval arithmetics for polynomial constraints. By definition, $\{\Pi^p_{IA}\}$ represents a sub-theory of $T^p_{IA}$.

**Lemma 3.2.1.** *Let $M^p_{IA} = (\mathbb{I}, I^p_{IA})$ be a model over intervals, $i_1, i_2 \in \mathbb{I}$, $r_1 \in i_2, r_2 \in i_2$, $p \in P^p$, we have $i_1 \; p_{IA} \; i_2 = 0$ implies not $(r_1 \; p_{\mathbb{R}} \; r_2)$*

**Example 3.2.1.** *We have $\langle 1, 3 \rangle \succ_{IA} \langle 5, 8 \rangle = 0$ by the definition of $\prec_{IA}$. Take $2 \in \langle 1, 3 \rangle$ and $6 \in \langle 5, 8 \rangle$. Following Lemma 3.2.1, we have not $(2 \prec_{\mathbb{R}} 6)$ holds or not $(2 < 6)$ holds (because $\prec_{\mathbb{R}} = <$) which is obviously true.*

*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 $l_1 \leq r_1 \leq h_1$, and
- $r_2 \in i_2$ 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 $\succ$, 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 $\prec$, 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 $\succeq$, 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 $\preceq$, 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 $\approx$, 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 $\not\approx$, 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* $i \in \mathbb{I}$, $g$ *is a polynomial ($\Sigma^p$-term). For every model over real numbers* $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$, *we have* $g^{M^p_{\mathbb{R}}} \in g^{\Pi^p_{IA}}$.

The intention of this lemma is that for given a box of variables' intervals and a polynomial, interval arithmetic will, essentially, output an interval that contains all the possible values of the polynomial with respect to any point inside the box.

*Proof.* Let $M^p_{\mathbb{R}} = (\mathbb{R}, I^p_{\mathbb{R}}) \in \Pi^p_{\mathbb{R}}$. As mentioned in Section 3.2.2, $\Pi$ can also be referred as a map from variables to intervals, i.e. $\{v \mapsto i \mid v \in V\}$. Proof is done by induction on structure of polynomial $f$.

1. **Base case**

   - If $g = v \in V$, we have

     $$\begin{array}{rcl} v^{M^p_{\mathbb{R}}} & = & I^p_{\mathbb{R}}(v) \in \Pi(v) \text{ because } M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}, \text{ and} \\ v^{\Pi^p_{IA}} & = & \Pi(v) \end{array}$$

     Then, $v^{M^p_{\mathbb{R}}} \in v^{\Pi^p_{IA}}$.

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

2. **Induction case:** $g = f(g_1, g_2)$ for some $f \in F^p \setminus \{\mathbf{1}\}$.

   We have

$$
\begin{array}{rcl}
f^{M^p_{\mathbb{R}}}(g_1, g_2) & = & g_1^{M^p_{\mathbb{R}}} \; f_{\mathbb{R}} \; g_2^{M^p_{\mathbb{R}}} \\
f^{\Pi^p_{IA}}(g_1, g_2) & = & g_1^{\Pi^p_{IA}} \; f_{IA} \; g_2^{\Pi^p_{IA}}
\end{array}
$$

   By induction hypothesis, we have $g_1^{M^p_{\mathbb{R}}} \in g_1^{\Pi^p_{IA}}$ and $g_2^{M^p_{\mathbb{R}}} \in g_2^{\Pi^p_{IA}}$ which due to the properties of $f_{IA}$ implies $g_1^{M^p_{\mathbb{R}}} \; f_{\mathbb{R}} \; g_1^{M^p_{\mathbb{R}}} \in g_2^{\Pi^p_{IA}} \; f_{IA} \; g_2^{\Pi^p_{IA}}$, or $g^{M^p_{\mathbb{R}}} \in g^{\Pi^p_{IA}}$

□

**Theorem 3.2.1.** *Let* $\Pi = \bigwedge_{v \in V} v \in i$ *with* $i \in \mathbb{I}$, *then* $\{\Pi^p_{IA}\}$ *is an over-approximation of* $\Pi^p_{\mathbb{R}}$.

*Proof.* Given an polynomial constraint $\varphi$ and suppose that $\varphi$ is $\{\Pi^p_{IA}\}$-UNSAT. We will prove that $\varphi$ is $\Pi^p_{\mathbb{R}}$-UNSAT by induction on structure of $\varphi$.

1. **Base case:** $\varphi^p = p(g_1, g_2)$ for some $p \in P^p$.

   We prove the lemma for the base case by contradiction. Suppose $\varphi$ 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 $\models_{M^p_{\mathbb{R}}} \varphi \iff \varphi^{M^p_{\mathbb{R}}} = 1$. We have

$$
\varphi^{M^p_{\mathbb{R}}} = 1 \implies g_1^{M^p_{\mathbb{R}}} \; p_{\mathbb{R}} \; g_2^{M^p_{\mathbb{R}}} \tag{3.1}
$$

   On the other hand,

$$
\varphi \text{ is } \{\Pi^p_{IA}\}\text{-UNSAT} \implies \varphi^{\Pi^p_{IA}} = 0 \implies g_1^{\Pi^p_{IA}} \; p_{IA} \; g_2^{\Pi^p_{IA}} = 0
$$

   In addition, because $g_1^{M^p_{\mathbb{R}}} \in g_1^{\Pi^p_{IA}}$ and $g_2^{M^p_{\mathbb{R}}} \in g_2^{\Pi^p_{IA}}$ (Lemma 3.2.2), we have

$$
g_1^{\Pi^p_{IA}} \; p_{IA} \; g_2^{\Pi^p_{IA}} = 0 \implies \text{not } (g_1^{M^p_{\mathbb{R}}} \; p_{\mathbb{R}} \; g_2^{M^p_{\mathbb{R}}})(\text{Lemma } 3.2.1) \tag{3.2}
$$

   Contradiction is raised between (3.1) and (3.2). As the result, $\varphi$ must be $\Pi^p_{\mathbb{R}}$-UNSAT.

2. **Induction case:** $\varphi = \varphi_1 \wedge \varphi_2$.

   We have $\varphi$ is $\{\Pi^p_{IA}\}$-UNSAT $\implies \not\models_{\Pi^p_{IA}} (\varphi_1 \wedge \varphi_2) \implies (\varphi_1 \wedge \varphi_2)^{\Pi^p_{IA}} = 0 \implies \max(\varphi_1^{\Pi^p_{IA}}, \varphi_2^{\Pi^p_{IA}}) = 0 \implies \varphi_1^{\Pi^p_{IA}} = 0$ and $\varphi_2^{\Pi^p_{IA}} = 0$.

   Thus, by induction hypothesis, $\varphi_1$ and $\varphi_2$ are $\Pi^p_{\mathbb{R}}$-UNSAT $\implies$ for all $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$, $\not\models_{M^p_{IA}} \varphi_1$ and for all $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$, $\not\models_{M^p_{IA}} \varphi_2 \implies$ for all $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$, $\not\models_{M^p_{IA}} (\varphi_1 \wedge \varphi_2) \implies \varphi_1 \wedge \varphi_2$ is $\Pi^p_{\mathbb{R}}$-UNSAT.

□

## 3.3 Testing as an Under-Approximation Theory

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

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

*Proof.* Let $\varphi$ be a polynomial constraint and suppose it is $T_T$-SAT. We need to prove $\varphi$ is $T$-SAT.

We have $\varphi$ is $T_T$-SAT $\implies$ there exists $M \in T_T$ such that $\models_M \varphi$ $\implies$ there exists $M \in T$ such that $\models_M \varphi$ (because $T_T \subseteq T$) $\implies$ $\varphi$ is $T$-SAT. $\square$

Given the interval constraint $\Pi = \bigwedge_{v_i \in V} v \in \langle l_i, h_i \rangle$, we have $\Pi^p_{\mathbb{R}}$ is a sub-theory of $T^p_{\mathbb{R}}$. We randomly select a number of models from $\Pi^p_{\mathbb{R}}$ (by randomly picking values for variables) to form the testing theory $(\Pi^p_{\mathbb{R}})_T$ of $\Pi^p_{\mathbb{R}}$.

**Example 3.3.1.** *Let $\Pi = x \in \langle 1, 5 \rangle \wedge y \in \langle -5, 10 \rangle$ be an interval constraint. If we pick two values for $x$ (e.g. $\{0, 2\}$) and one value for $y$ (e.g. $\{-3\}$), we will have two assignments from real numbers to variables:*

$$\theta_1 = \{x \mapsto 0, y \mapsto -3\},\ \textit{and}$$
$$\theta_1 = \{x \mapsto 2, y \mapsto -3\}$$

*Thus, the testing theory $(\Pi^p_{\mathbb{R}})_T$ of $\Pi^p_{\mathbb{R}}$ is*

$$(\Pi^p_{\mathbb{R}})_T = \{(\theta_1)^p_{\mathbb{R}}, (\theta_2)^p_{\mathbb{R}}\}$$

## 3.4 raSAT Loop

Our algorithm **raSAT** loop is described using a transition system. Each state of the search procedure is represented by $(\Pi, \varphi, \mathring{\Pi}, \varphi^V, \varphi^U, \varepsilon, \tau)$ where

- $\Pi$ is a CNF interval constraint.
- $\varphi$ represents the polynomial constraint.
- $\mathring{\Pi} = \bigwedge_{v_i \in V} v_i \in \langle l_i, h_i \rangle$ with $\langle l_i, h_i \rangle \in \mathbb{I}$. We use $\emptyset$ to denote the empty conjunction.
- $\varphi^V$ consists of conjunction of inequalities that are VALID under over-approximation. We use $\emptyset$ to denote the empty conjunction.
- $\varphi^U$ is the set of inequalities which are UNKNOWN under over-approximation. We use $\emptyset$ to denote the empty conjunction.
- $\varepsilon$ indicates the threshold to stop decomposing intervals.

- $\tau$ is a flag to mark whether the threshold of intervals has been reached. It can be one of two values $\perp$ and $\top$. In the initial state, $\tau$ is always $\perp$.

The transition rules are described in Table 3.2. Figure 3.1 illustrates the transition system. Given one box (product of variables' intervals), e.g. $(x, y) \in \langle 1, 5\rangle \times \langle -3, 8\rangle$, IA and Testing attempt to show the satisfiability/unsatisfiability of the constraint. If neither does, the interval of some variable is decomposed in to smaller intervals, e.g. $\langle 1, 5\rangle$ of x is decomposed into $\langle 1, 2\rangle$ and $\langle 2, 5\rangle$; creating two boxes, e.g. $\langle 1, 2\rangle \times \langle -3, 8\rangle$ and $\langle 2, 5\rangle \times \langle -3, 8\rangle$. Each of these box will be examined in next iterations. We use a SAT solver which implements DPLL procedure to handle the combinations(boxes) of variables intervals by considering each interval, e.g $x \in \langle 1, 5\rangle$, as a propositional atom. The boxes is represented by interval constraint $\Pi$ and $\mathring{\Pi}$ represents the output of DPLL procedure on $\Pi$. We use threshold $\epsilon$ to prevent some interval to be decomposed deeply. When a box has the size smaller that $\epsilon$, it is pruned by being removed from the considering boxes represented by $\Pi$. The transition rules can be understood as following.

- **Π_UNSAT**: The DPLL procedure fails to find an assignment that satisfy $\Pi$ (in terms of propositional logic), then there are no more boxes that possibly make the constraint satisfiable. Because no intervals were pruned by threshold ($\tau = \perp$), the given constraint is unsatisfiable with respect to the initial box.

- **Π_UNKNOWN**: The DPLL procedure fails and some intervals were pruned by threshold ($\tau = \top$), we can conclude neither SAT nor UNSAT.

- **Π_SAT**: DPLL procedure outputs one assignment representing a box which is stored in $\mathring{\Pi}$. We call this box is *the current box*. The box is sent to Interval Arithmetic modules to check against the constraint.

- **IA_UNSAT**: If IA can prove that the constraint is unsatisfiable in the current box, the box will be removed from the considering boxes simply by adding $\neg\mathring{\Pi}$ into $\Pi$.

- **IA_SAT**: If IA does not disprove the constraint with respect to the current box, the inequalities in the given constraint is divided into two set:
  - $\varphi^V$ consisting of inequalities that are proved to be VALID in the current box by IA and
  - $\varphi^U$ (possibly empty) consisting of remaining inequalities.

- **IA_VALID**: If IA showed that all the inequalities are VALID in the current box ($\varphi^U = \emptyset$), we can conclude the satisfiability of the constraint.

- **TEST_SAT**: Inequalities $\varphi^U \neq \emptyset$ that can not be verified by IA are passed to the Testing module. If some instances in the current box of variables are found that make $\varphi^U$ satisfiable, we can conclude that $\varphi$ is satisfiable because $\varphi^V$ is satisfiable for all instances in the current box (proved by IA).

$$\frac{\emptyset \| \Pi \Longrightarrow^{!} FailState}{(\Pi,\varphi,\emptyset,\emptyset,\varepsilon,\bot)\rightarrow UNSAT}\ \textsc{Π\_unsat}$$

$$\frac{\emptyset \| \Pi \Longrightarrow^{!} FailState}{(\Pi,\varphi,\emptyset,\emptyset,\varepsilon,\top)\rightarrow UNKNOWN}\ \textsc{Π\_unknown}$$

$$\frac{\emptyset \| \Pi \Longrightarrow^{!} \mathring{\Pi}}{(\Pi,\varphi,\emptyset,\emptyset,\varepsilon,\tau)\rightarrow(\Pi,\varphi,\mathring{\Pi},\emptyset,\emptyset,\varepsilon,\tau)}\ \textsc{Π\_sat}$$

$$\frac{\mathring{\Pi}\neq\emptyset \quad \varphi^V\wedge\varphi^U=\varphi \quad \varphi^V \text{ is } \{\mathring{\Pi}^p_{IA}\}\text{-VALID}}{(\Pi,\varphi,\mathring{\Pi},\emptyset,\emptyset,\varepsilon,\tau)\rightarrow(\Pi,\varphi,\mathring{\Pi},\varphi^V,\varphi^U,\varepsilon,\tau)}\ \textsc{ia\_sat}$$

$$\frac{\varphi^V=\varphi}{(\Pi,\varphi,\mathring{\Pi},\varphi^V,\varphi^U,\varepsilon,\tau)\rightarrow SAT}\ \textsc{ia\_valid}$$

$$\frac{\mathring{\Pi}\neq\emptyset \quad \varphi^U\neq\emptyset \quad \varphi^U \text{ is } (\mathring{\Pi}^p_{\mathbb{R}})_T\text{-SAT}}{(\Pi,\varphi,\mathring{\Pi},\varphi^V,\varphi^U,\varepsilon,\tau)\rightarrow SAT}\ \textsc{test\_sat}$$

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

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

$$\frac{\mathring{\Pi}\neq\emptyset \quad \varphi \text{ is } \{\mathring{\Pi}^p_{IA}\}\text{-UNSAT}}{(\Pi,\varphi,\mathring{\Pi},\emptyset,\emptyset,\varepsilon,\tau)\rightarrow(\Pi\wedge\neg\mathring{\Pi},\varphi,\emptyset,\emptyset,\emptyset,\varepsilon,\tau)}\ \textsc{ia\_unsat}$$

Table 3.2: Transition rules

- **THRESHOLD**: Neither IA nor Testing concludes the constraint and the current box has the size smaller than threshold $\epsilon$, the box will be also removed from the considering boxes and $\tau$ is set to $\top$ to mark this pruning.
- **REFINE**: Neither IA nor Testing concludes the constraint and the some interval of the current box has the size larger than threshold $\epsilon$, decomposition is implemented on that interval.

**Theorem 3.4.1.** *Starting with state* $(\Pi, \varphi, \emptyset, \emptyset, \varepsilon, \bot)$*, if* $\Pi = \bigwedge_{v_i\in V} v_i \in \langle l_i, h_i\rangle$ *and* $\langle l_1, h_1\rangle \times \langle l_2, h_2\rangle \times \cdots$ *is bounded, raSATloop terminates.*

*Proof.* In the worst case, all the interval will be decomposed into smallest boxes with size of $\varepsilon$ whose number are bounded to $\frac{h_1-l_1}{\varepsilon} * \frac{h_2-l_2}{\varepsilon} * \cdots$ (the number of variables in one

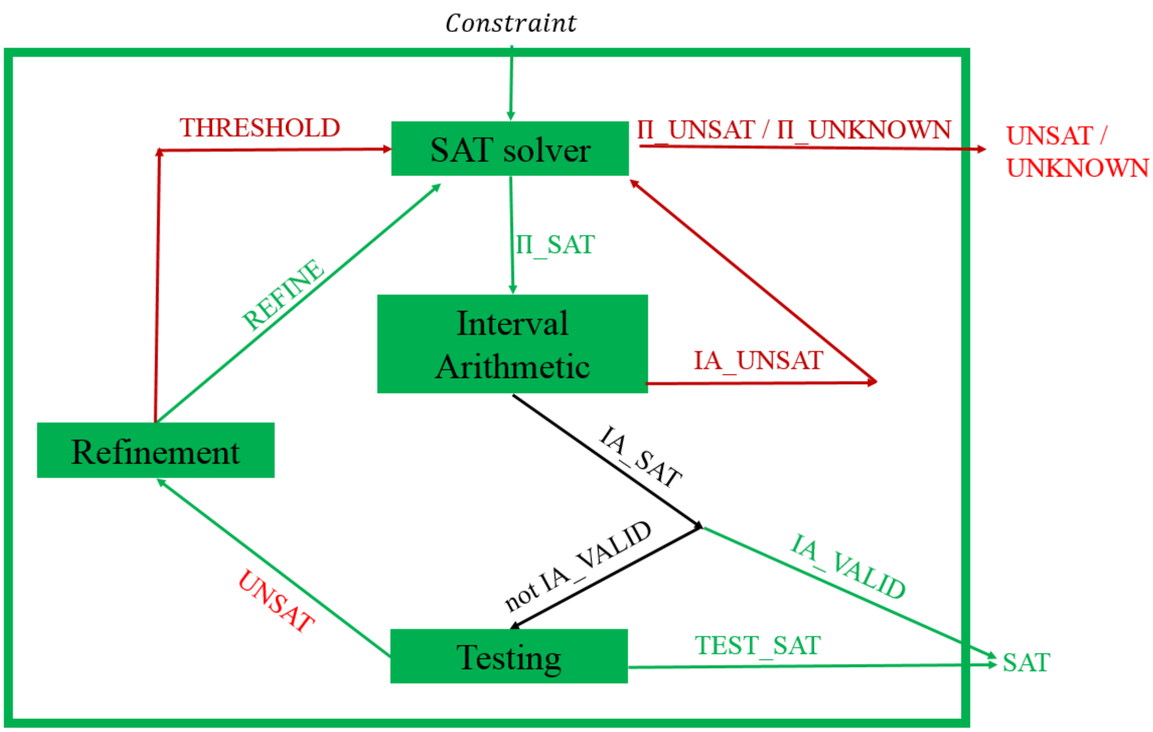



Figure 3.1: **raSAT** design

polynomial constraint is also bounded). As a result, raSATloop terminates after checking all of these boxes. □

**Example 3.4.1.** *In Theorem 3.4.1, if* $\Pi = x \in \langle 1, 5\rangle \wedge y \in \langle -3, 8\rangle$ *and* $\epsilon = 0.1$*, then decomposition will create maximally*

$$\frac{5-1}{0.1} * \frac{8-(-3)}{0.1} = 150$$

*boxes.*

## 3.5 Soundness - Completeness

### 3.5.1 Soundness

**Theorem 3.5.1.** *Let* $(\Pi_0, \varphi_0, \mathring{\Pi}_0, \varphi_0^V, \varphi_0^U, \varepsilon, \perp)$ *be the starting state and* $(\Pi, \varphi, \mathring{\Pi}, \varphi^V, \varphi^U, \varepsilon, \tau)$ *be any state of our system, then the following properties are invariants:*

1. $\mathring{\Pi}^p_{\mathbb{R}} \subseteq T^p_{\mathbb{R}}$
2. $\varphi^V$ *is* $\mathring{\Pi}^p_R$*-VALID.*

*3.* $\varphi^U = \emptyset \vee (\varphi = \varphi^U \wedge \varphi^V)$

*4.* $\varphi$ *is* $\Pi^p_{\mathbb{R}}$*-UNSAT and* $\tau = \bot$ *implies that* $\varphi$ *is* $(\Pi_0)^p_{\mathbb{R}}$*-UNSAT*

*Proof.* 1. Easy from the definition.

2. Easy from the transitions and the fact that $\{\mathring{\Pi}^p_{IA}\}$ is an over-approximation of $\mathring{\Pi}^p_{\mathbb{R}}$ (Theorem 3.2.1).

3. Easy from the transitions.

4. The proof is done inductively on transitions of the system:

- **Initial state:** It is obvious because $\Pi = \Pi_0$.
- **Transitions**:
  - **Π_SAT** and **IA_SAT**: The interval constraint $\Pi$ does not changed, so if the properties holds for the former state, it also does for the later one.
  - **REFINE**:
    Denote $\Pi' = \Pi \wedge \Pi''$ where:

$$\Pi'' = (\neg I_j \vee I_{j1} \vee I_{j2}) \wedge (I_j \vee \neg I_{j1}) \wedge (I_j \vee \neg I_{j2}) \wedge (\neg I_{j1} \vee \neg I_{j2})$$

    We will prove $(\Pi')^p_{\mathbb{R}} = \Pi^p_{\mathbb{R}}$ by showing $(\Pi')^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$ and $(\Pi')^p_{\mathbb{R}} \ni \Pi^p_{\mathbb{R}}$.
    $\boldsymbol{(\Pi')^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}}$: Let $M^p_{\mathbb{R}} = (\mathbb{R}, I^p_{\mathbb{R}})$ be any model in $(\Pi')^p_{\mathbb{R}}$ and $\theta = \{v \mapsto I^p_{\mathbb{R}}(v) \mid v \in V\}$. By definition, we have $\theta^p_{\mathbb{R}} = M^p_{\mathbb{R}}$. Because $M^p_{\mathbb{R}} \in (\Pi')^p_{\mathbb{R}}$, it is the case that $\models_{\theta^I_{\mathbb{R}}} \Pi'$, which implies

$$\begin{aligned}(\Pi')^{\theta^I_{\mathbb{R}}} = 1 &\implies (\Pi \wedge \Pi'')^{\theta^I_{\mathbb{R}}} = 1 \implies \min((\Pi)^{\theta^I_{\mathbb{R}}}, (\Pi'')^{\theta^I_{\mathbb{R}}}) = 1 \\ &\implies (\Pi)^{\theta^I_{\mathbb{R}}} \text{ and } (\Pi'')^{\theta^I_{\mathbb{R}}} \implies \models_{\theta^I_{\mathbb{R}}} \Pi \implies \theta^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}\end{aligned}$$

    or $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$. As the result, $(\Pi')^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$.
    $\boldsymbol{\Pi^p_{\mathbb{R}} \in (\Pi')^p_{\mathbb{R}}}$: Let $M^p_{\mathbb{R}} = (\mathbb{R}, I^p_{\mathbb{R}})$ be any model in $\Pi^p_{\mathbb{R}}$ and $\theta = \{v \mapsto I^p_{\mathbb{R}}(v) \mid v \in V\}$. By definition, we have $\theta^p_{\mathbb{R}} = M^p_{\mathbb{R}}$. Because $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$, it is the case that $\models_{\theta^I_{\mathbb{R}}} \Pi$. There are two possible cases: $\models_{\theta^I_{\mathbb{R}}} I_j$ or $\not\models_{\theta^I_{\mathbb{R}}} I_j$. In the first case, by the construction of $I_{j1}$ and $I_{j2}$ we can imply that $\models_{\theta^I_{\mathbb{R}}} I_{j1}$ or $\models_{\theta^I_{\mathbb{R}}} I_{j2}$. These imply $\models_{\theta^I_{\mathbb{R}}} \Pi''$ and thus $\models_{\theta^I_{\mathbb{R}}} \Pi'$ or $M^p_{\mathbb{R}} \in (\Pi')^p_{\mathbb{R}}$. In the second case, i.e. $\not\models_{\theta^I_{\mathbb{R}}} I_j$, again by the construction of $I_{j1}$ and $I_{j2}$, $\not\models_{\theta^I_{\mathbb{R}}} I_j$ implies $\not\models_{\theta^I_{\mathbb{R}}} I_{j1}$ and $\not\models_{\theta^I_{\mathbb{R}}} I_{j2}$. These imply that $\not\models_{\theta^I_{\mathbb{R}}} \Pi''$ (by some simple calculation we can prove that $(\Pi'')^{\theta^p_{\mathbb{R}}} = 1$). As a result, $\models_{\theta^I_{\mathbb{R}}} \Pi'$ or $M^p_{\mathbb{R}} \in (\Pi')^p_{\mathbb{R}}$.
    In either case, we have $M^p_{\mathbb{R}} \in (\Pi')^p_{\mathbb{R}}$ for any $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$. Then, $\Pi^p_{\mathbb{R}} \in (\Pi')^p_{\mathbb{R}}$.
  - **IA_UNSAT**: suppose $\varphi$ is $(\Pi \wedge \neg\mathring{\Pi})^p_{\mathbb{R}}$-UNSAT. We need to prove that $\varphi$ is $\Pi^p_{\mathbb{R}}$-UNSAT. Let $M^p_{\mathbb{R}} = (\mathbb{R}, I^p_{\mathbb{R}})$ be any model in $\Pi^p_{\mathbb{R}}$ and $\theta = \{v \mapsto I^p_{\mathbb{R}}(v) \mid v \in V\}$. The later is the assignment from real numbers to variables that are included in the former and by definition $M^p_{\mathbb{R}} = \theta^p_{\mathbb{R}}$.

Because $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$, be definition of $\Pi^p_{\mathbb{R}}$ we have $\models_{\theta^I_{\mathbb{R}}} \Pi$. There are two possible cases: either $\models_{\theta^I_{\mathbb{R}}} \mathring{\Pi}$ or $\not\models_{\theta^I_{\mathbb{R}}} \mathring{\Pi}$.
If $\models_{\theta^I_{\mathbb{R}}} \mathring{\Pi}$, by definition $\theta^p_{\mathbb{R}} \in \mathring{\Pi}^p_{\mathbb{R}}$. In addition because $\varphi$ is $\{\mathring{\Pi}^p_I A\}$-UNSAT, $\varphi$ is $\mathring{\Pi}^p_{\mathbb{R}}$-UNSAT (derived from Theorem 3.2.1). As a result, $\not\models_{\theta^p_R} \varphi$ or $\not\models_{M^p_R} \varphi$.
If $\not\models_{\theta^I_{\mathbb{R}}} \mathring{\Pi}$, then by Lemma 2.2.1 we have $\models_{\theta^I_{\mathbb{R}}} \neg\mathring{\Pi}$ which implies $\models_{\theta^I_{\mathbb{R}}} \Pi \wedge \neg\mathring{\Pi}$ (because $\models_{\theta^I_{\mathbb{R}}} \Pi$). By definition this implies $\theta^p_{\mathbb{R}} \in (\Pi \wedge \neg\mathring{\Pi})^p_{\mathbb{R}}$ or $M^p_{\mathbb{R}} \in (\Pi \wedge \neg\mathring{\Pi})^p_{\mathbb{R}}$. In addition because $\varphi$ is $(\Pi \wedge \neg\mathring{\Pi})^p_{\mathbb{R}}$-UNSAT, we can imply $\not\models_{M^p_{\mathbb{R}}} \varphi$
In any cases, we can imply that $\not\models_{M^p_{\mathbb{R}}} \varphi$ for any $M^p_{\mathbb{R}} \in \Pi^p_{\mathbb{R}}$. In other words, $\varphi$ is $\Pi^p_{\mathbb{R}}$-UNSAT.

□

**Theorem 3.5.2.** *Let $\varphi$ be the polynomial constraint to be solved. Starting with the state $(\Pi = \bigwedge_{v_i \in V} v_i \in \langle -\infty, +\infty \rangle, \varphi, \emptyset, \emptyset, \emptyset, \epsilon, \bot)$, if our transitional system terminates and output:*

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

*Proof.* Because the starting state is $\Pi = \bigwedge_{v_i \in V} v_i \in \langle -\infty, +\infty \rangle$, by definition we have $\Pi^p_{\mathbb{R}} = T^p_{\mathbb{R}}$.

- If the system output SAT, there are two possibles transition to SAT:
  - In the case of IA_VALID, we have $\varphi^V$ is $\mathring{\Pi}^p_R$-VALID (invariant 2) $\implies$ $\varphi^V$ is $\mathring{\Pi}^p_{\mathbb{R}}$-SAT $\implies$ $\varphi$ is $\mathring{\Pi}^p_{\mathbb{R}}$-SAT (because $\varphi^V = \varphi$ is the condition of this transition). In addition, following invariant 1, we have $\mathring{\Pi}^p_{\mathbb{R}} \subseteq T^p_{\mathbb{R}}$, then $\varphi$ is $T^p_{\mathbb{R}}$-SAT (Lemma 2.2.3).
  - In the case of TEST_SAT, $\varphi^U$ is $\mathring{\Pi}^p_{\mathbb{R}}$-SAT $\implies$ there exist $M^p_{\mathbb{R}} \in \mathring{\Pi}^p_{\mathbb{R}}$ such that $\models_{M^p_{\mathbb{R}}} \varphi^V \implies (\varphi^V)^{M^c_{\mathbb{R}}} = 1$. In addition, because $\varphi^V$ is $\mathring{\Pi}^p_{\mathbb{R}}$-VALID (invariant 2) and $M^p_{\mathbb{R}} \in \mathring{\Pi}_{\mathbb{R}}$, we have $\models_{M^p_{\mathbb{R}}} \varphi^U$ or $(\varphi^U)^{M^p_{\mathbb{R}}} = 1$. Consider the evaluation of $\varphi$ under the model $M^p_{\mathbb{R}}$: $(\varphi)^{M^p_{\mathbb{R}}} = (\varphi^U \wedge \varphi^V)^{M^p_{\mathbb{R}}}$ (invariant 3) $= min((\varphi^U)^{M^p_{\mathbb{R}}}, (\varphi^V)^{M^p_{\mathbb{R}}}) = min(1,1) = 1 \implies \models_{M^p_{\mathbb{R}}} \varphi \implies \varphi$ is $\mathring{\Pi}^p_{\mathbb{R}}$-SAT $\implies \varphi$ is $T^p_{\mathbb{R}}$-SAT (because of invariant 1 and Lemma 3.2.1)
- If the system output UNSAT, there is only one transition of rule Π_UNSAT. Because $\emptyset \parallel \Pi \implies^! FailState$, $\Pi$ is unsatisfiable in the sense of propositional logic, and thus it cannot be satisfiable in terms of first order logic. As the result, by definition $\Pi^p_{\mathbb{R}}$ is empty which implies that $\varphi$ is $\Pi^p_{\mathbb{R}}$-UNSAT. By invariant 4, $\varphi$ is $T^p_{\mathbb{R}}$-UNSAT.

□

### 3.5.2 Completeness

**Definition 3.5.1.** *Let* $\Pi = \bigwedge_{v_i \in V} v_i \in \langle l_i, h_i \rangle$*, and* $\varphi = \bigwedge_{i=1}^{n} f_i > 0$*. An theory* $T$ *is complete with respect to the theory real numbers over polynomial constraint* $T_{\mathbb{R}}^{p}$ *if for each* $O \subset \langle l_1, h_1 \rangle \times \langle l_2, h_2 \rangle \times \cdots$*,* $c = (c_1, c_2, \cdots) \in O$*, and* $\delta > 0$*, there exists* $\gamma > 0$*,* $T' \subset T$*, such that:*

- $\langle c_1 - \gamma, c_1 + \gamma \rangle \times \langle c_2 - \gamma, c_2 + \gamma \rangle \times \cdots \subset O$*,*
- $\bigwedge_{i=1}^{n} (f_i(c) - \delta < f_i(x)) \wedge (f_i(x) < f_i(c) + \delta)$ *is* $T'$*-VALID, and*
- $T'$ *is an over-approximation of* $(\Pi')_{\mathbb{R}}^{p}$ *where* $\Pi' = \bigwedge_{v_i \in V} v_i \in \langle c_i - \gamma, c_i + \gamma \rangle$*.*

**Lemma 3.5.1.** *Let* $\Pi = \bigwedge_{v_i \in V} v_i \in \langle l_i, h_i \rangle$ *where* $\langle l_i, h_i \rangle \in \mathbb{I}$ *is open, and* $\varphi = \bigwedge_{j=1}^{n} f_j > 0$*. Denote*

- $S_\varphi = \{(r_1, r_2, \cdots) \mid \theta = \{v_i \mapsto r_i \mid v_i \in V\}$ *and* $\models_{\theta_{\mathbb{R}}^{p}} \varphi\}$ *be the set of points that satisfy the constraint* $\varphi$*, and*
- $S = (\langle l_1, h_1 \rangle \times \langle l_1, h_1 \rangle \times \cdots) \cap S_\varphi$ *be the set of points that*
  - *are inside the box represented by* $\Pi$*.*
  - *satisfy the constraint* $\varphi$*.*

*If* $S \neq \emptyset$*, then* $S$ *contain an open set.*

*Proof.* Because $S \neq \emptyset$, there exist $c = (c_1, c_2, \cdots) \in S$. By definition of $S$, for all $j \in \{1, \cdots, n\}$, $f_j(c) > 0$. Take $\delta = \min_{j=1 \cdots n} f_j(c)$, then $\delta > 0$. Because the polynomials are continuous, there exists $\gamma > 0$ such that for all $v \in \langle c - \gamma, c + \gamma \rangle$, $\bigwedge_{j=1}^{n} f_j(c) - \delta < f_j(v) < f_j(c) + \delta$ which implies for all $v \in \langle c - \gamma, c + \gamma \rangle$, $\bigwedge_{j=1}^{n} f_j(v) > 0$ (because $\delta = \min_{j=1 \cdots n} f_j(c) \leq f_j(c)$ for any $j$). Now consider the open interval

$$O = (\max(l_1, c_1 - \gamma), \min(h_1, c_1 + \gamma) \times (\max(l_2, c_2 - \gamma), \min(h_2, c_2 + \gamma) \times \cdots$$

It is easy to see that $O \subset \langle c - \gamma, c + \gamma \rangle \subset S_\varphi$ and $O \subset \langle l_1, h_1 \rangle \times \langle l_1, h_1 \rangle \times \cdots$ that implies $O \in S$. In addition because $\langle l_i, h_i \rangle$ is open for each $i = 1, 2, \cdots$, $O$ is open by its construction. $\square$

**Theorem 3.5.3.** *Let* $\Pi = \bigwedge_{v_i \in V} v_i \in \langle l_i, h_i \rangle$ *where* $\langle l_i, h_i \rangle \in \mathbb{I}$ *is open, and* $\varphi = \bigwedge_{j=1}^{n} f_j > 0$*.*

- *$S_\varphi = \{(r_1, r_2, \cdots) \mid \theta = \{v_i \mapsto r_i \mid v_i \in V\}$ and $\models_{\theta^p_{\mathbb{R}}} \varphi\}$ be the set of points that satisfy the constraint $\varphi$, and*

- *$S = (\langle l_1, h_1\rangle \times \langle l_1, h_1\rangle \times \cdots) \cap S_\varphi$ be the set of points that are inside the box represented by $\Pi$ and also satisfy the constraint $\varphi$.*

*If $S \neq \emptyset$, $(l_1, h_1) \times (l_2, h_2) \times \cdots$ is bounded, and the threshold $\epsilon$ is small enough; then raSATloop can detect the satisfiability of $\varphi$ with assumption that the theory of interval arithmetic $T^p_{IA}$ is complete.*

*Proof.* Based on Lemma 3.5.1, there exist an open box $(l, h) \in S$ such that for all $(r_1, r_2, ...) \in (l, h)$, $\bigwedge_{j=1}^{n} f_j > 0$. Take any $c \in (l, h)$ and take $\delta = \min_{j=1\cdots n}(f_j(c))$. Because IA is complete by assumption, from Definition 3.5.1 there exists $\gamma > 0$ and $T \subset T^p_{IA}$ such that

- $\langle c - \gamma, c + \gamma\rangle \in (l, h)$,

- $\bigwedge_{j=1}^{n} f_j(c) - \delta < f_j(v) < f_j(c) + \delta$ is $T$-VALID, and

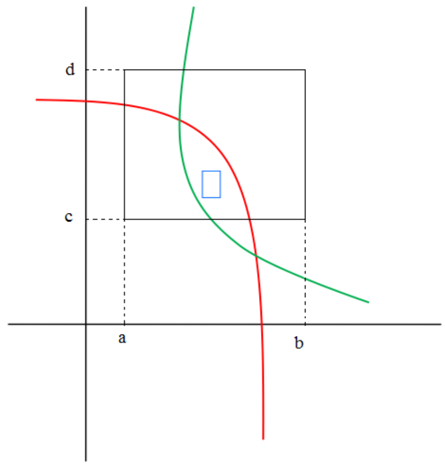
- $T$ is an over-approximation of $(\Pi')^p_{\mathbb{R}}$ where $\Pi' = \bigwedge_{v_i \in V} v_i \in \langle c_i - \gamma, c_i + \gamma\rangle$.

From above second and third conditions, IA can be used to prove that $\bigwedge_{j=1}^{n} f_j(c) - \delta < f_j(v) < f_j(c) + \delta$ is $(\Pi')^p_{\mathbb{R}}$-VALID (Theorem 3.1.1) which implies that $\bigwedge_{j=1}^{n} 0 < f_j(v)$ is $(\Pi')^p_{\mathbb{R}}$-VALID (because $\delta = \min_{j=1\cdots n}(f_j(c)) \implies f_j(c) \geq \delta$ for all $j = 1, 2, \cdots, n$) or $\varphi$ is $(\Pi')^p_{\mathbb{R}}$-VALID $\implies$ $\varphi$ is $(\Pi')^p_{\mathbb{R}}$-SAT $\implies$ $\varphi$ is $T^p_{\mathbb{R}}$-SAT.

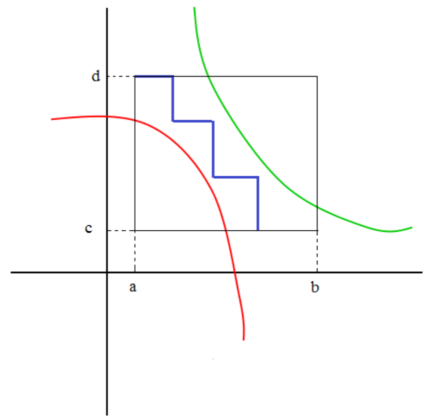
By taking $\gamma$ as the threshold in $(\Pi, \varphi, \emptyset, \emptyset, \gamma, \perp)$, raSATloop will terminate (Theorem 3.4.1). Furthermore, $\langle c-\gamma, c+\gamma\rangle \in (l, h) \implies (c+\gamma)-(c-\gamma) < h-l \implies 2\gamma < h-l$. As a consequence, decomposition eventually creates a box of size $\gamma$ inside $(h, l)$ which can be used to conclude the satisfiability of the constraint by IA. □

Figure 3.2a illustrates a simple case for this theorem where we have two inequalities and the initial box is represented by $\Pi = x \in \langle a, b\rangle \wedge y \in \langle c, d\rangle$. Here $a < b$ and $c < d$ make the box open. This box intersects with set of points that satisfy both inequalities. As a consequence, decomposition will create a box (the blue one) that can be used by IA to prove the satisfiability of two inequalities.

**Theorem 3.5.4.** *Let $\Pi = \bigwedge_{v_i \in V} v_i \in \langle l_i, h_i\rangle$ where $\langle l_i, h_i\rangle \in \mathbb{I}$ is open, and $\varphi = \bigwedge_{j=1}^{n} f_j > 0$. Denote $S = \{(r_1, r_2, \cdots) \mid \theta = \{v_i \mapsto r_i \mid v_i \in V\}, \models_{\theta^I} \Pi$ and $\models_{\theta^p_{\mathbb{R}}} \bigwedge_{j=1}^{n} f_j \geq 0\}$ be the set of all points which are inside the box represented by $\Pi$ and also satisfy $\bigwedge_{j=1}^{n} f_j \geq 0$.*




(a) Example of SAT completeness




(b) Example of UNSAT completeness

Figure 3.2: Examples on complete cases of **raSAT**

*If $S = \emptyset$, $(l_1, h_1) \times (l_2, h_2) \times \cdots$ is bounded, and the threshold $\epsilon$ is small enough; then raSATloop can prove the unsatisfiability of $\varphi$ with assumption that the theory of Interval Arithmetic $T^p_{IA}$ is complete.*

*Proof.* Let $f(v) = \min_{j=1}^{n} f_j(v)$, then $f(v)$ is continuous. Because $D = \langle l_1, h_1\rangle \times \langle l_2, h_2\rangle \times \cdots$ is compact, $\delta = |\max_{v\in D} f(v)|$ exists.

First we will prove that $\delta > 0$. In fact, suppose $\delta = 0 \implies |\max_{v\in D} f(v)| = 0 \implies$ for all $c \in D$, $f(c) = 0 \implies$ for all $c \in D$, $\min_{j=1}^{n} f_j(c) = 0 \implies$ for all $c \in D$, for all $j \in \{1, \cdots, n\}$, $f_j(c) \geq 0$. This contradicts with the assumption that $S = \emptyset$.

Because $T^p_{IA}$ is complete, by Definition 3.5.1, for any point $c \in D$, there exists $\gamma > 0$ and $T \subset T^p_{IA}$ such that

- $\langle c - \gamma, c + \gamma\rangle \in D$,
- $\bigwedge_{j=1}^{n} f_j(c) - \delta < f_j(v) < f_j(c) + \delta$ is $T$-VALID, and
- $T$ is an over-approximation of $(\Pi')^p_{\mathbb{R}}$ where $\Pi' = \bigwedge_{v_i\in V} v_i \in \langle c_i - \gamma, c_i + \gamma\rangle$.

From above second and third conditions, IA can be used to prove $\bigwedge_{j=1}^{n} f_j(c) - \delta < f_j(v) < f_j(c) + \delta$ is $(\Pi')^p_{\mathbb{R}}$-VALID (Theorem 3.1.1).

Let $f_k(c) = \min_{j=1}^{n} f_j(c)$ for $k \in \{1, 2, \cdots, n\}$ then $f_k(c) < 0$ otherwise a contradiction with the assumption that $S = \emptyset$ exists. In addition, by definition of $f(v)$ we have $f_k(c) = f(c)$

which implies $|f_k(c)| \leq |\max_{v \in D} f(v)| \implies f_k(c) + \delta \leq 0$. Moreover, IA can prove that $f_k(v) < f_k(c) + \delta$ is $(\Pi')^p_{\mathbb{R}}$-VALID. We have $f_k(v) < f_k(c) + \delta$ is $(\Pi')^p_{\mathbb{R}}$-VALID $\implies$ $f_k(v) < 0$ is $(\Pi')^p_{\mathbb{R}}$-VALID $\implies \neg(f_k(v) < 0) = f_k(v) \geq 0$ is $(\Pi')^p_{\mathbb{R}}$-UNSAT (Lemma 2.2.2) $\implies f_k(v) > 0$ is $(\Pi')^p_{\mathbb{R}}$-UNSAT $\implies \bigwedge_{j=1}^{n} f_j(v) > 0$ is $(\Pi')^p_{\mathbb{R}}$-UNSAT.

In conclusion, for any point in $D$, we can find a small box containing that point in which the constraint can be proved to be unsatisfiable by IA. As a result, if $\varepsilon$ is small enough, **raSAT** loop will terminate (Theorem 3.4.1) and proves the unsatisfiability of the constraint after checking all the small decomposed boxes. □

Figure 3.2b illustrates a simple example for this theorem where we have two inequalities and the initial box is represented by $\Pi = x \in \langle a, b\rangle \wedge y \in \langle c, d\rangle$. Here $a < b$ and $c < d$ make the box open. The constraint is unsatisfiable inside the box and decomposition will eventually separate two satisfiable areas of two inequalities.

The limitation of UNSAT detection comes from the case of kissing situation. Figure 3.3 presents an example of this with the constraint

$$x^2 + y^2 < 4 \wedge (x-4)^2 + (y-3)^2 < 9$$

which is UNSAT but Interval Arithmetic can not uses boxes to separate the satisfiable areas around the touching points of two inequalities. The condition $S = \emptyset$ in the above Theorem avoids such a kissing situation.

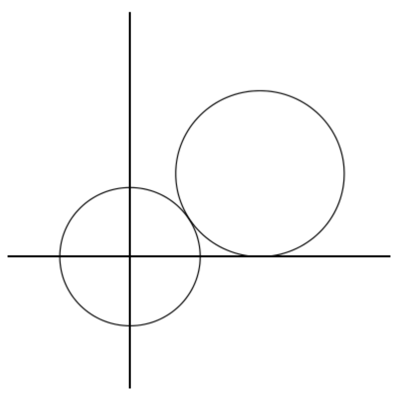


Figure 3.3: Kissing situation

Note that is Theorem 3.5.3 and 3.5.3 requires that the threshold $\epsilon$ is small enough. Although computing this enough small $\epsilon$ is not easy, **raSAT** achieves this by incremental deepening (Chapter 5) strategy in which the value of threshold is made to be smaller if **raSAT** fails to conclude the constraint.

# Chapter 4

# Variations of 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 4.1.
4. $I^p_{CI}(\mathbf{1}) = \langle 1, 1\rangle$
5. $\forall v \in V$; $I^p_{CI} \in U^p_{CI}$

| 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 | $\langle 0, 0\rangle$ |

Table 4.1: Definition of $\otimes_{CI}$

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 $\Pi^p_{CI}$ 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 the interpretation of $x^2 - x \times y$ is

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

Types of Affine Interval vary by choices of estimating multiplications $\epsilon_1^2$ and $\epsilon_1\epsilon_2$:

1. AA [3, 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 $\epsilon_1^2$ by the fixed noise symbols $\epsilon_+$ or $\epsilon_-$.

These variations of Affine Interval are discussed in details in [11]. This thesis focuses on AF2 because currently it is used in **raSAT**.

## AF2

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 $\Pi^p_{CI}$ as the model represented by $\Pi = \{x \in \langle l, h \rangle | v \in V\}$.

**Theorem 4.2.1.** *AF2 is an IA.*

*Proof.* Easy. $\square$

# Chapter 5

# 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 $\varphi = \bigwedge_{j=1}^{m} f_j > 0$ be the constraint to be solved.

### 5.1.1 Incremental Windening and Deepening

Given $0 < \gamma_0 < \gamma_1 < \cdots$ and $\varepsilon_0 > \varepsilon_1 > \cdots > 0$ raSAT's algorithm design with incremental widening and deepening is described in Algorithm 1. The idea here is that raSAT starts searching within small interval and large value of threshold. If SAT is detected, the result can be safely returned. If the current intervals cannot satisfy the constraint (UNSAT is detected), larger intervals are consider. In the case of UNKNOWN, the threshold is not small enough to detect either SAT or UNSAT. As the result, raSAT decreases it and restarts the search. In Theorem 3.5.3 and Theorem 3.5.4, the threshold $\gamma$ is not easy to calculate, but by incremental deepening, such threshold can be eventually reached because it does exist.

### 5.1.2 Incremental Testing

One obstacle in testing is the exponentially large number of test instances (number of selected models). 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:

**Algorithm 1** Incremental Widening and Deepening

$i \leftarrow 0$
$j \leftarrow 0$
**while** true **do**
  $\Pi = \bigwedge_{v_i \in V} v_i \in \langle -\gamma_i, \gamma_i \rangle$
  **if** $(\Pi, \varphi, \emptyset, \emptyset, \emptyset, \varepsilon_j, \bot) \rightarrow SAT$ **then**
    **return** SAT
  **else if** $(\Pi, \varphi, \emptyset, \emptyset, \emptyset, \varepsilon_j, \bot) \rightarrow UNSAT$ **then**
    **if** $\gamma_i = +\infty$ **then**
      **return** UNSAT
    **else**
      $i \leftarrow i + 1$
    **end if**
  **else**
    $j \leftarrow j + 1$
  **end if**
**end while**

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 10 inequalities by SAT-likelihood.
   - Select 1 variable of each selected API using sensitivity.
2. Incrementally generate test values for variables to prune test cases that do not satisfy an inequality. This was proposed by Khanh and Ogawa in [11]:
   - Dynamically sort the IA-SAT inequalities by SAT-likelihood such that the inequality which is less likely to be satisfiable will be prioritized.
   - Generate the test values for variables of selected inequalities.

   **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 $x^2 > 4$ .*

**Definition 5.1.1.** *Given an assignment from variables to intervals $\theta = \{v \mapsto i | v \in V\}$ in which $i \in \mathbb{I}$ and an inequality $f > 0$. Let $\langle l, h \rangle = f^{\theta^I}$, then the SAT-likelihood of $f > 0$ is $|\langle l, h \rangle \cap \langle l, h \rangle| / (h - l)$ which is denoted as $\varpi(f > 0, \theta)$.*

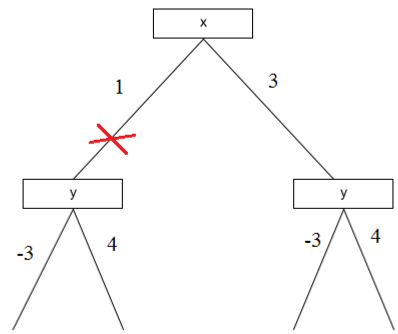



Figure 5.1: Incremental Testing Example

**Definition 5.1.2.** *Given an assignment from variables to intervals in the form of affine interval* $\theta = \{v_i \mapsto a_{0i} + a_{1i}\epsilon_i | v_i \in V\}$ *and a polynomial* $f$*. Let* $a_0 + \sum_{j=1}^{n} a_j\epsilon_j = f^{\theta^I}$*, then we define sensitivity of variable* $v_i \in V$ *by the value of* $a_i$*.*

## 5.2 Refinement Heuristics

Suppose the number of variable is $nVar$ and initially the intervals assignment is represented by $\Pi = \bigwedge_{v_i \in V} v_i \in \langle l_i, h_i \rangle$. If the interval of each variable is decomposed into two smaller ones, the new interval constraint becomes $\Pi' = \bigwedge_{v_i \in V} (v_i \in \langle l_i, c_i \rangle \vee v_i \in \langle c_i, h_i \rangle)$ where $c_i$ is the decomposed point such that $l_i < c_i < h_i$. The number of boxes becomes $2^{nVar}$. The exponentially increase in the number of boxes affect the scalability of raSAT. To this point, raSAT applies two strategies for boosting SAT detection:

1. In REFINE transition, the interval of one variable is selected for decomposition, raSAT chooses such variables through the following steps:

    - Choose the inequality $f > 0$ in $\varphi^U$ with the least value of SAT-likelihood.
    - Within $f$, choose the variable $v_k$ with the largest value of sensitivity.

2. After the interval of $v_k$ is decomposed, basically the box represented by $\mathring{\Pi} = \bigwedge_{v_i \in V} v_i \in \langle l_i, h_i \rangle$ will become two boxes which are represented by $\mathring{\Pi}_1 = v_0 \in \langle l_0, h_0 \rangle \wedge \cdots \wedge v_k \in \langle l_k, c_k \rangle \wedge \cdots \wedge v_{nVar} \in \langle l_{nVar}, h_{nVar} \rangle$ and $\mathring{\Pi}_2 = v_0 \in \langle l_0, h_0 \rangle \wedge \cdots \wedge v_k \in \langle c_k, h_k \rangle \wedge \cdots \wedge v_{nVar} \in \langle l_{nVar}, h_{nVar} \rangle$. raSAT prepares the following strategies to choose one box to explore:

    - the box with higher/smaller value of SAT-likelihood to explore in the next iteration, i.e. the result of SAT operation in Π_SAT is controlled so that the desired box will be selected, and

- the box with higher/smaller inequalities that can be solved by IA and Testing.

**Definition 5.2.1.** *Given an assignment from intervals to variables* $\theta = \{v \mapsto i | v \in V\}$ *in which* $i \in \mathbb{I}$ *and a constraint* $\bigwedge_{i=1}^{n} f_i > 0$. *The SAT-likelihood of* $\theta$ *is defined as* $\min_{i=1}^{n} \varpi(f_i > 0, \theta)$.

## 5.3 UNSAT Core

In IA_UNSAT rule, the negation of $\mathring{\Pi}$ is added into the interval constraint so that $\mathring{\Pi}$ will not be explore again later because it make the constraint unsatisfiable. If we can find $\mathring{\Pi}'$ such that $\mathring{\Pi} = \mathring{\Pi}' \wedge \mathring{\Pi}''$ and $\varphi$ is $\{\mathring{\Pi}'^{p}_{IA}\}$-UNSAT, we can add $\neg\mathring{\Pi}'$ into interval constraint instead of $\mathring{\Pi}$ to reduce the search space.

**Example 5.3.1.** *Consider the constraint* $\varphi = x^2 + y^2 < 1$. *Suppose in the* IA_UNSAT *rule, we have* $\Pi = (x \in \langle 2,3\rangle \vee x \in \langle 0,2\rangle) \wedge (y \in \langle 0,1\rangle y \in \langle -1,0\rangle)$ *and* $\mathring{\Pi} = x \in \langle 2,3\rangle \wedge y \in \langle 0,1\rangle$. *The conditions of* IA_UNSAT *are satisfied,* $\neg\mathring{\Pi}$ *is added into the interval constraint which becomes* $\Pi \wedge \neg\mathring{\Pi}$. *The new interval constraint contains* $\{x \in \langle 2,3\rangle, y \in \langle -1,0\rangle\}$ *as one of its solution. However, with* $\mathring{\Pi}' = x \in \langle 2,3\rangle$, *we have* $\varphi$ *is* $\{\mathring{\Pi}'^{p}_{IA}\}$*-UNSAT and by adding* $\neg\mathring{\Pi}'$ *to the interval constraint,* $\{x \in \langle 2,3\rangle, y \in \langle -1,0\rangle\}$ *is also removed from the search space.*

The constraint $\varphi = \bigwedge_{j=1}^{n} f_j > 0$ is $\mathring{\Pi}$-UNSAT when $f_k > 0$ is $\mathring{\Pi}$-UNSAT with some $k \in \{1, 2, \cdots, n\}$. We have two ideas for computing UNSAT core.

1. *UNSAT core 1:* A sub-polynomial $f'_k$ of $f_k$ such that
   - $f'_k > 0$ is $\mathring{\Pi}$-UNSAT implies that $f_k$ is $\mathring{\Pi}$-UNSAT, and
   - $f'_k$ is in fact $\mathring{\Pi}$-UNSAT.

   In this case, we just take $\mathring{\Pi}' = \bigwedge_{v_i \in var(f_k)} v_i \in \langle l_i, h_j\rangle$.

2. *UNSAT core 2:* Check all the possible cases if $\mathring{\Pi}'$.

**Example 5.3.2.** *Consider again the constraint* $\varphi = x^2 + y^2 < 1$ *or* $1 - x^2 - y^2 > 0$. *In the* IA_UNSAT *rule, we also have* $\Pi = (x \in \langle 2,3\rangle \vee x \in \langle 0,2\rangle) \wedge (y \in \langle 0,1\rangle y \in \langle -1,0\rangle)$ *and* $\mathring{\Pi} = x \in \langle 2,3\rangle \wedge y \in \langle 0,1\rangle$. *Here,* $\varphi$ *is* $\Pi$*-UNSAT. In addition,* $1 - x^2$ *is the UNSAT core of* $1 - x^2 - y^2$ *because*

- $1 - x^2$ *is* $\Pi^p_{IA}$*-UNSAT implies that* $1 - x^2 - y^2$ *is* $\Pi^p_{IA}$*-UNSAT, and*
- *the constraint* $1 - x^2 > 0$ *is in fact* $\Pi^p_{IA}$*-UNSAT.*

## 5.4 Test Case Generation

The value of variable's sensitivity can also be used to approximate how likely the value of a polynomial increases when the value of that variable increases. Consider the constraint $f = -x_{15} * x_8 + x_{15} * x_2 - x_{10} * x_{16} > 0$. With $x_2 \in [9.9, 10], x_8 \in [0, 0.1], x_{10} \in [0, 0.1], x_{15} \in [0, 10]$, and $x_{16} \in [0, 10]$. The result of AF2 for $f$ is: $0.25\epsilon_2 - 0.25\epsilon_8 - 0.25\epsilon_{10} + 49.5\epsilon_{15} - 0.25\epsilon_{16} + 0.75\epsilon_{+-} + 49.25$. The coefficient of $\epsilon_2$ is positive (0.25), then we expect that if $x_2$ increases, the value of $f$ also increase. As the result, the test case of $x_2$ is as high as possible in order to satisfy $f > 0$. We will thus take the upper bound value of $x_2$, i.e. 10. Similarly, we take the test cases for other variables: $x_8 = 0, x_{10} = 0, x_{15} = 10, x_{16} = 0$. With these test cases, we will have $f = 100 > 0$.

## 5.5 Box Decomposition

Currently raSAT applies balanced decomposition in REFINE rule, e.g. $x \in \langle 0, 10\rangle$ will be decomposed into $x \in \langle 0, 5\rangle$ and $x \in \langle 5, 10\rangle$. We intend to use the same approximation as in Section 5.4 to guide the decomposition. Take the same example in Section 5.4 and suppose $x_{15}$ is selected for decomposition. Because the coefficient of $\epsilon_{15}$ is 49.5 which is positive, we expect the high values for $x_{15}$ so that $f > 0$ will be satisfied. As the result, the interval $x_{15} \in \langle 0, 10\rangle$ can be decomposed into $x_{15} \in \langle 0, 10 - \epsilon\rangle$ and $x_{15} \in \langle 10 - \epsilon, 10\rangle$ where $\epsilon$ is the threshold for 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.

In order to avoid soundness bugs from round-off/over-flow errors, we integrated **iR-RAM**[1] to check the SAT instances provided by Testing module. If a bug is detected, raSAT continues searching other boxes instead of concluding satisfiability of the constraint.

## 6.1 Experiments on Strategy Combinations

### Experiments on incremental testing and refinements heuristics

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

Our combinations of strategies mentioned in Section 5.2 are,

| Selecting a test-UNSAT API | Selecting a box (to explore): | Selcting a variable: |
|---|---|---|
| (1) Least SAT-likelyhood. | (3) Largest number of SAT inequalities. | (8) Largest sensitibity. |
| (2) Largest SAT-likelyhood. | (4) Least number of SAT inequalities. | |
| | (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, 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.

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

| 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: Combinations of **raSAT** strategies on NRA/Zankl,Meti-Tarski benchmark

## Preliminary Experiments on UNSAT core

Strategy *UNSAT core 1* was already implemented in [10] and the other strategy was implemented in this work. We did the experiments on the SMT-LIB family Hong where each problem is unsatisfiable and of the following form:

$$\sum_{i=1}^{n} x_i^2 < 1 \wedge \prod_{i=1}^{n} x_i > 1$$

where $n$ ranges from 1 to 20. Table 6.2 shows the experiments for raSAT with/without UNSAT core strategies where we fixed the initial intervals to be $\langle 0, 10 \rangle$ and the threshold 0.1 (i.e. no incremental deepening and widening). *UNSAT core 2* works fine because problems in Hong family contain polynomials with lots of UNSAT cores which this strategy tends to calculate. On the other hand, *UNSAT core 1* has not been well implemented in the way raSAT search for a sub-polynomial. Experiments for these strategies on other benchmarks of SMT-LIB did not show improvements in the result, we need further investigations.

## Preliminary Experiments on Test Case Generation and Box Decomposition Strategies

Table 6.3 illustrates the experiments of this strategy together with (1)-(5)-(8). In comparison with (1)-(5)-(8), this strategy solves more satisfiable constraints and the solving time is generally smaller for the same constraint.

| Problem | No UNSAT core | | UNSAT core 1 | | UNSAT core 2 | |
|---|---|---|---|---|---|---|
| | Time (s) | Result | Time (s) | Result | Time (s) | Result |
| hong_1 | 0 | UNSAT | 0 | UNSAT | 0.004 | UNSAT |
| hong_2 | 0.00838 | UNSAT | 0.016 | UNSAT | 0.016 | UNSAT |
| hong_3 | 0.007441 | UNSAT | 0.016 | UNSAT | 0.016 | UNSAT |
| hong_4 | 0.114857 | UNSAT | 0.124 | UNSAT | 0.016 | UNSAT |
| hong_5 | 0.27588 | UNSAT | 0.272 | UNSAT | 0.028 | UNSAT |
| hong_6 | 1.20687 | UNSAT | 1.288 | UNSAT | 0.052 | UNSAT |
| hong_7 | 9.29289 | UNSAT | 9.996 | UNSAT | 0.112 | UNSAT |
| hong_8 | 153.619 | UNSAT | 164.288 | UNSAT | 0.68 | UNSAT |
| hong_9 | 117.937 | UNSAT | 129.044 | UNSAT | 0.08 | UNSAT |
| hong_10 | 307.208 | UNSAT | 281.696 | UNSAT | 0.152 | UNSAT |
| hong_11 | 478.605 | UNSAT | 412.028 | UNSAT | 0.236 | UNSAT |
| hong_12 | 500 | Timeout | 500 | Timeout | 0.456 | UNSAT |
| hong_13 | 500 | Timeout | 500 | Timeout | 0.752 | UNSAT |
| hong_14 | 500 | Timeout | 500 | Timeout | 1.572 | UNSAT |
| hong_15 | 500 | Timeout | 500 | Timeout | 2.756 | UNSAT |
| hong_16 | 500 | Timeout | 500 | Timeout | 5.98 | UNSAT |
| hong_17 | 500 | Timeout | 500 | Timeout | 10.864 | UNSAT |
| hong_18 | 500 | Timeout | 500 | Timeout | 24.352 | UNSAT |
| hong_19 | 500 | Timeout | 500 | Timeout | 47.968 | UNSAT |
| hong_20 | 500 | Timeout | 500 | Timeout | 103.484 | UNSAT |

Table 6.2: Experiments on UNSAT core computations

| Benchmark | SAT | | UNSAT | |
|---|---|---|---|---|
| Zankl/matrix-1 (53) | 24 | 511.07 (s) | 2 | 0.009(s) |
| Zankl/matrix-2,3,4,5 (98) | 13 | 477.62 (s) | 8 | 0.39(s) |

Table 6.3: raSAT with sensitivity in testing

| Benchmark | SAT | | UNSAT | |
|---|---|---|---|---|
| Zankl/matrix-1 (53) | 24 | 510.55 (s) | 2 | 0.01(s) |
| Zankl/matrix-2,3,4,5 (98) | 9 | 1030.35 (s) | 8 | 0.38(s) |

Table 6.4: raSAT with sensitivity in decomposition

The experiments on box decomposition strategy is shown in Table 6.4 shows when it is used together with (1)-(5)-(8) and the strategy in Section 5.4. Basically the result has not been improved in comparison with the result in Section 5.4, we need further investigation.

## 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
- *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.

| 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 6.5: Comparison among SMT solvers

| raSAT | | | | Z3 | | | |
|---|---|---|---|---|---|---|---|
| SAT | | UNSAT | | SAT | | UNSAT | |
| 156 | 244.6(s) | 2 | 0.03 (s) | 205 | 1.12 (s) | 22 | 0.05 (s) |

Table 6.6: Experiments on 217 QEPCAD problems from Hidenao Iwane

## 6.3 Experiments with QE-CAD Benchmark

We also did experiments on QE-CAD problems provided by Mohab Safey El Din[2] - LIP6 who is working on QE-CAD simplification (checking SAT/UNSAT only and the complexity is from DEXP to EXP) and the QEPCAD problems collected by Hidenao Iwane[3].

The benchmarks from LIP6 are all unsatisfiable and contain long polynomials (in terms of monomials number). From the experiments with raSAT and Z3, these benchmarks show that they are very difficult to be solved (Table 6.7). These benchmarks may be suitable targets for our UNSAT core strategies in the future.

In terms of QEPCAD benchmarks, while Z3 solves all 217 problems, raSAT solves 156/205 satisfiable and 2/22 unsatisfiable benchmarks. Apart from a number of problems which raSAT has errors in parsing, the unsolved ones are mostly unsatisfiable. This can be tackled by strategies on UNSAT core which is left for our future works.

[2] `http://www-polsys.lip6.fr/~safey/`
[3] `https://github.com/hiwane/qepcad/`

| Problem | raSAT | | Z3 | |
|---|---|---|---|---|
| | Time (s) | Result | Time (s) | Result |
| f23 | 3600 | Timeout | 3599.55 | Timeout |
| f22 | 3600 | Timeout | 3599.51 | Timeout |
| pol | 3600 | Timeout | 0.02499 | UNSAT |
| f13 | 3600 | Timeout | 3599.17 | Timeout |
| pol1 | 3600 | Timeout | 3601.61 | Timeout |
| f12 | 3600 | Timeout | 3599.27 | Timeout |

Table 6.7: Experiments on problems from LIP6

# Chapter 7

# Extensions: Equality Handling and Polynomial Constraint over Integers

## 7.1 SAT on Equality by Intermediate Value Theorem

### Single Equation

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.

**Lemma 7.1.1.** *For* $\varphi = \bigwedge_j^m f_j > 0 \ \wedge \ g = 0$*, $F$ is SAT, if there is a box represented by* $\Pi = \bigwedge_{v_i \in V} v_i \in (l_i, h_i)$ *such that*

*(i)* $\bigwedge_j^m f_j > 0$ *is* $\Pi_{\mathbb{R}}^p$*-VALID, and*

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

*Proof.* It is clear from the Intermediate Value Theorem that there exist an point $\vec{t_0}$ between $\vec{t}$ and $\vec{t'}$ such that $g(\vec{t_0}) = 0$. In addition, because $\bigwedge_j^m f_j > 0$ is $\Pi_{\mathbb{R}}^p$-VALID, $\vec{t_0}$ also satisfies $\bigwedge_j^m f_j > 0$. As a result, $\varphi$ is satisfiable with $\vec{t_0}$ as the SAT instance. □

**Example 7.1.1.** *Consider the constraint* $\varphi = f(x, y) > 0 \wedge g(x, y) = 0$*. Suppose we can find a box represented by* $\Pi = x \in \langle a, b\rangle \wedge y \in \langle c, d\rangle$ *such that* $f(x, y) > 0$ *is* $\Pi_{\mathbb{R}}^p$*-VALID (Figure 7.1). In addition, inside that box, we can find two points* $(u_1, v_1)$ *and* $(u_2, v_2)$ *such that* $g(u_1, v_1) > 0$ *and* $g(u_2, v_2) < 0$*. By Lemma 7.1.1, the constraint is satisfiable.*

**raSAT** first tries to find a box of variables' intervals (by refinements) such that $\bigwedge_j^m f_j > 0$ is VALID inside that box. Then it tries to find 2 instances for $g > 0$ and $g < 0$ by testing.

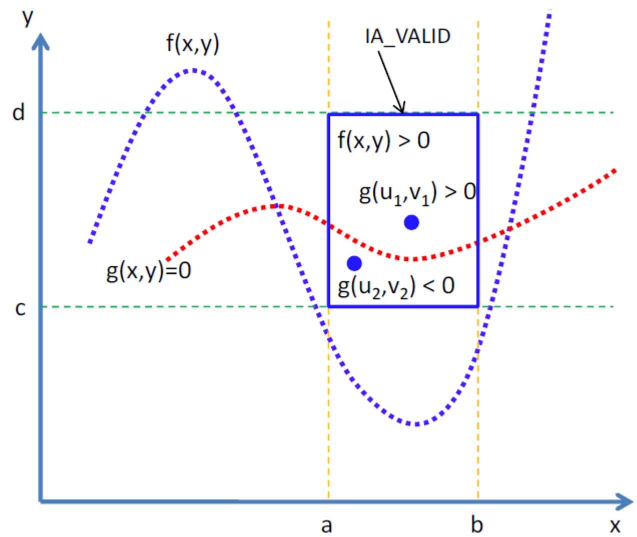



Figure 7.1: Example on solving single equation using the Intermediate Value Theorem

Intermediate Value Theorem guarantees the existence of an SAT instance in between. Note that this method does not find an exact SAT instance.

## Multiple Equations

The idea of using the Intermediate Value Theorem can also be used for solving multiple equations. Consider $n$ equations ($n \geq 1$): $\bigwedge_{i=1}^{n} g_i = 0$ and an interval constraint $\bigwedge_{v_i \in V} v_i \in \langle l_i, h_i \rangle$. If we can find a set $\{V_1, \cdots, V_n\}$ that satisfies the following properties, then we can conclude that $\bigwedge_{i=1}^{n} g_i = 0$ is satisfiable in $\bigwedge_{v_i \in V} v_i \in \langle l_i, h_i \rangle$.

- For all $i = 1, \cdots, n$; we have $V_i \subset var(g_i)$.
- For all $i \neq j$, we have $V_i \neq V_j$.
- For all $i = 1, \cdots, n$; let $k_i = |V_i|$ and $V_i = \{v_{ij} \mid 1 \leq j \leq k_i\}$. Then, there exist two values $(v_{i1}, \cdots, v_{ik_I}) = (x_{i1}, \cdots, x_{ik_i})$ and $(v_{i1}, \cdots, v_{ik_I}) = (x'_{i1}, \cdots, x'_{ik_i})$ such that
$$g_i(x_{i1}, \cdots, x_{ik_i}, \cdots, v_{ik}, \cdots) > 0$$
and
$$g_i(x'_{i1}, \cdots, x'_{ik_i}, \cdots, v_{ik}, \cdots) < 0$$
for all values of $v_{ik}$ in $\langle l_{ik}, h_{ik} \rangle$ where $v_{ik} \in var(g_i) \setminus V_i$. We denote $ivt(g_i, V_i, \Pi)$ to represent that the polynomial $g_i$ enjoy this property with respect to $V_i$ and $\Pi$.

By the first two properties, this method restricts that the number of variables must be greater than or equal to the number of equations.

**Example 7.1.2.** *Consider two equations $g_1(x,y) = 0$ and $g_2(x,y) = 0$ (Figure 7.2) which satisfy the above restriction on the number of variables, and the variable intervals is $\Pi = x \in \langle c_1, d_1\rangle \wedge y \in \langle d_2, c_2\rangle$. Let $V_1 = \{x\}$ and $V_2 = \{y\}$, we have:*

$$g_1(c_1, y) < 0 \textit{ and } g_1(d_1, y) < 0 \textit{ for all } y \in \langle d_2, c_2\rangle\textit{; and}$$

$$g_2(x, d_2) > 0 \textit{ and } g_2(x, c_2) < 0 \textit{ for all } x \in \langle c_1, d_1\rangle$$

*Thus we can conclude that $g_1(x,y) = 0 \wedge g_2(x,y) = 0$ has a solution inside the box represented by $\Pi$.*

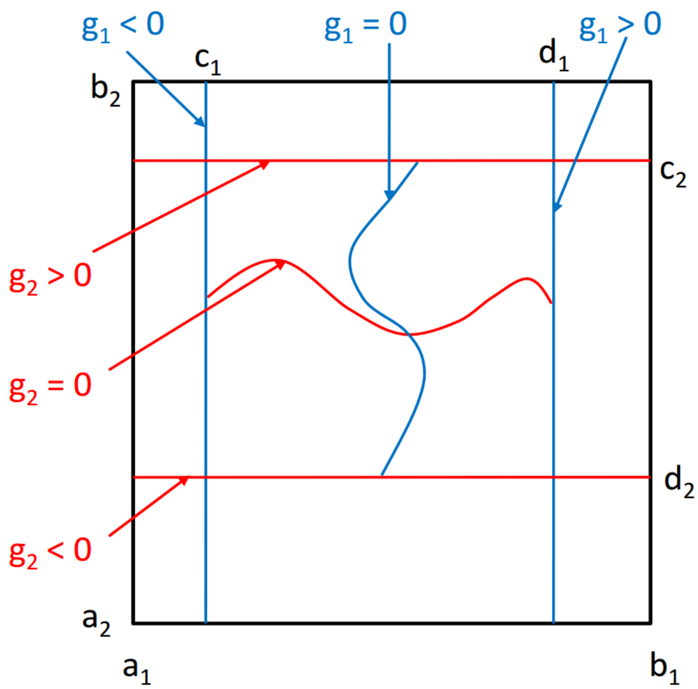



Figure 7.2: Example on solving single equation using the Intermediate Value Theorem

Our current implementation of handling multiple equations is very naive which is described in Algorithm 2 because for each polynomial, **raSAT** checks every possible subsets of its variables. As the result, given the constraint $\bigwedge_{i=1}^{n} g_i = 0$, in the worst case **raSAT** will check $2^{|var(g_1)|} * \cdots * 2^{|var(g_n)|}$ cases. As a future work, we may use variables' sensitivity to give priority on subsets of variables.

## Experiments on Benchmarks

In Table 7.1 we show preliminary experiment for 15 problems that contain polynomial equations in Zankl family. 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**.

**Algorithm 2** Solving multiple equations $\bigwedge_{i=1}^{n} g_i = 0$ with interval constraint $\Pi = \bigwedge_{v_i \in V} v_i \in \langle l_i, h_i \rangle$

**function** EQUATIONSPROVER($\bigwedge_{i=j}^{n} g_i = 0, \Pi, V_0$)
    **if** $j > n$ **then** ▷ All equations are checked
        **return** SAT
    **end if**
    **for** $V_j \in P(var(g_j))$ **do** ▷ $P(var(g_j))$ is the powerset of $var(g_j)$
        **if** $V_j \cap V = \emptyset$ and $ivt(V', g_j, \Pi)$ **then**
            $V_0 \leftarrow V_0 \cup V'$
            **if** EQUATIONSPROVER($\bigwedge_{i=j+1}^{n} g_i = 0, \Pi, V_0$) = SAT **then**
                **return** SAT
            **end if**
        **end if**
    **end for**
    **return** UNSAT
**end function**
EQUATIONSPROVER($\bigwedge_{i=1}^{n} g_i = 0, \Pi, \emptyset$)

| Problem Name | No. Variables | No. Equalities | No. Inequalities | **Z3 4.3** (15/15) | | **raSAT** (15/15) | |
|---|---|---|---|---|---|---|---|
| | | | | Result | Time(s) | Result | Time(s) |
| gen-03 | 1 | 1 | 0 | SAT | 0.01 | SAT | 0.001 |
| gen-04 | 1 | 1 | 0 | SAT | 0.01 | SAT | 0.001 |
| gen-05 | 2 | 2 | 0 | SAT | 0.01 | SAT | 0.003 |
| gen-06 | 2 | 2 | 1 | SAT | 0.01 | SAT | 0.005 |
| gen-07 | 2 | 2 | 0 | SAT | 0.01 | SAT | 0.002 |
| gen-08 | 2 | 2 | 1 | SAT | 0.01 | SAT | 0.009 |
| gen-09 | 2 | 2 | 1 | SAT | 0.03 | SAT | 0.007 |
| gen-10 | 1 | 1 | 0 | SAT | 0.02 | SAT | 0.002 |
| gen-13 | 1 | 1 | 0 | UNSAT | 0.05 | UNSAT | 0.002 |
| gen-14 | 1 | 1 | 0 | UNSAT | 0.01 | UNSAT | 0.002 |
| gen-15 | 2 | 3 | 0 | UNSAT | 0.01 | UNSAT | 0.03 |
| gen-16 | 2 | 2 | 1 | SAT | 0.01 | SAT | 0.006 |
| gen-17 | 2 | 3 | 0 | UNSAT | 0.01 | UNSAT | 0.03 |
| gen-18 | 2 | 2 | 1 | SAT | 0.01 | SAT | 0.002 |
| gen-19 | 2 | 2 | 1 | SAT | 0.05 | SAT | 0.046 |

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

## 7.2 Polynomial Constraints over Integers

**raSAT** loop can be slightly modified to handle NIA (nonlinear arithmetic over integers) constraints from NRA, by setting $\gamma_0 = 1$ in incremental deepening in Section 5.1 and restricting test data generation on integers. 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 solve any unsatisfiable problem, since they have large coefficients which lead exhaustive search on large area.

# Chapter 8

# Related Works

## 8.1 Methodologies for Polynomial Constraints over Real Numbers

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)** [1] 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 root plus an infinitesimal $\epsilon$. 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 [17] 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 [5] 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 [4], dReal [7], and RSOLVER [18]. 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" [4] is practically often more efficient than algebraic computation.

Because **raSAT** in the same category with iSAT3 and dReal, next section is going to take a look at details of methodologies used in these solvers.

## 8.2 Solvers using Interval Constraint Propagation

### iSAT3

Although **iSAT3** also uses Interval Arithmetic, its algorithm integrates IA with DPLL procedure tighter than one of **raSAT**. During DPLL procedure, in addition to an assignment of literals, **iSAT3** also prepares a data structure to store interval boxes where each box corresponds to one decision level of DPLL procedure's assignment. In **Unit-Propagation** rule, intead of using standard rule, **iSAT3** searches for clauses that have all but one atoms being inconsistence with the current interval box. When some atom are selected for the literals assignment, this tool tries to use the selected atoms to contract the corresponding box to make it smaller. In order to do this, **iSAT3** convert each inequality/equation in the given constraints into the conjunction of the atoms of the following form by introducing additional variables:

$$
\begin{array}{lcl}
\text{atom} & ::= & \text{bound} \mid \text{equation} \\
\text{bound} & ::= & \text{variable relation rational\_constant} \\
\text{relation} & ::= & < \mid \leq \mid = \mid \geq \mid > \\
\text{equation} & ::= & \text{variable} = \text{variable bop variable} \\
\text{bop} & ::= & + \mid - \mid \times
\end{array}
$$

In other words, the resulting atoms are of the form, e.g., either $x > 10$ or $x = y + z$. For example, the constraint

$$x^2 + y^2 < 1$$

is converted into:

$$\begin{cases} x_1 = x^2 \\ x_2 = y^2 \\ x_3 = x_1 + x_2 \\ x_3 < 1 \end{cases}$$

From the atoms of these form, the contraction can be easily done for interval boxes:

- For the bound atom of the form, e.g., $x > 10$, if the bound is $x \in \langle 0, 100\rangle$, then the contracted box contain $x \in \langle 10, 100\rangle$.
- For the equations of three variables $x = y$ bop $z$, from bounds of any two variables, we can infer the bound for the remaining one. For example, from

$$\begin{cases} x = y.z \\ x \in \langle 1, 10\rangle \\ y \in \langle 3, 7\rangle \end{cases}$$

  we can infer that

$$z \in \langle \frac{1}{7}, \frac{10}{3}\rangle$$

When the **UnitPropagation** and contraction can not be done, **iSAT3** split one interval (decomposition) in the current box and select one decomposed interval to explore which corresponds to **decide** step. If the contraction yields an empty box, a conflict is detected and the complement of the bound selection in the last split needs to be asserted. This is done via **learn** the causes of the conflict and **backjump** to the previous bound selection of the last bound selection. In order to reason about causes of a conflict, **iSAT3** maintains an implication graph to represents which atoms lead to the asserting of one atom.

## dReal

In stead of showing satisfiability/unsatisfiability of the polynomial constraints $\varphi$ over the real numbers, **dReal** proves that either

- $\varphi$ is unsatisfiable, or
- $\varphi^\delta$ is satisfiable.

Here, $\varphi^\delta$ is the $\delta$-weakening of $\varphi$. For instance, the $\delta$-weakening of $x = 0$ is $|x| \leq \delta$. Any constraint with operators in $\{<, \leq, >, \geq, =, \neq\}$ can be converted into constraints that contains only $=$ by the following transformations.

- **Removing** $\neq$: Each formula of the form $f \neq 0$ is transformed into $f > 0 \vee f < 0$.
- **Removing** $<$ **and** $\leq$: Each formula of the form $f < 0$ or $f \leq 0$ is transformed into $-f \geq 0$ or $-f > 0$ respectively.
- **Removing** $>$ **and** $\geq$: Each formula of the form $f > 0$ or $f \geq 0$ is transformed into $f - x = 0$ by introducing an auxiliary variable $x$ that has bound $[0, m]$ or $(0, m]$ respectively. Here, $m$ is any rational number which is greater than the maximum of $f$ over intervals of variables. As the result, **dReal** requires the input that ranges of variables must be compact.

Note that the satisfiability of $\varphi^\delta$ does not imply that of $\varphi$. **dReal**'s methodology [6] also cooperates DPLL with ICP in the lazy manner as in **raSAT**.

# Chapter 9

# Conclusion

This thesis presented improvement and extensions for an SMT solved **raSAT** including heuristics to deal with exponential exploration of boxes, extensions for handling equations and handling constraints over integer numbers. From the experiments on standard SMT-LIB benchmarks, **raSAT** is able to solve large constraints (in terms of the number of variables) which are difficult for other tools. The contributions of this work are as follows:

1. To deal with exponential growth of the number of boxes during refinement (interval decomposition), two strategies for *selecting one variable* to decomposed and *selecting one box* were proposed:

   - **Selecting one box.** The box with more possiblity to satisfy the constraint is selected to explore, which is estimated by several heuristic measures, called *SAT likelyhood*, and *the number of unsolved polynomial inequalities*.
   - **Selecting one variable.** The most influential variable is selected for multiple test cases and decomposition. This is estimated by *sensitivity* which is determined during the approximation process.

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 exhaustive search, it starts searching with a threshold that each interval will be decomposed no smaller than it. If neither satisfiability nor unsatisfiability is detected, a smaller threshold is taken and raSAT restarts.
   - **Incremental widening**. Starting with a small intervals, if **raSAT** detects UNSAT, it enlarges input intervals and restarts. This strategy is effective in detecting satisfiability of constraints because small intervals reduce the number of boxes after decomposition.

3. *Satisfiability confirmation* step by an error-bound guaranteed floating point package **iRRAM**[1], to avoid soundess bugs caused by roundoff errors.

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

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

5. **raSAT** is also extended to *handle constraints over integer numbers* by simple extension in the approximation process.

# Future Directions

A number of ideas for the future works are:

1. **UNSAT core**: Two strategies *SAT likelihood* and *the number of unsolved inequalities* aim at boosting satisfiability detection. For unsatisfiable constraints, *UNSAT core* is the key in expeditious detection. Although we have some ideas for this but they did not show much improvement in experiments. As a future work, more investigation is needed for *UNSAT core*.

2. **Test case generation:** Currently **raSAT** randomly generates test cases in testing phase. We had the idea of using sensitivity to guide testing but this needs to be investigated in detail.

3. **Box Decomposition:** The key in UNSAT detection is also how to early isolate unsatisfiable intervals that is done through decomposition. Note that at the moment the decomposition strategy of *raSAT* is taking its middle point.

# Bibliography


[1] 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`.

[2] 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`.

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

[4] 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.

[5] 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.

[6] Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. δcompletee decision procedures for satisfiability over the reals. In *Proceedings of the 6th International Joint Conference on Automated Reasoning*, IJCAR'12, pages 286–300, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-31364-6. doi: 10.1007/978-3-642-31365-3_23. URL `http://dx.doi.org/10.1007/978-3-642-31365-3_23`.

[7] Sicun Gao, Soonho Kong, and EdmundM. Clarke. dreal: An smt solver for non-linear 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] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Abstract dpll and abstract dpll modulo theories. In *In LPAR04, LNAI 3452*, pages 36–50. Springer, 2005.

[17] 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.

[18] 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`.

[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`.