The present work was submitted to the LuFG Theory of Hybrid Systems

MASTER OF SCIENCE THESIS

IMPROVING INCREMENTAL LINEARIZATION FOR SATISFIABILITY MODULO NON-LINEAR REAL ARITHMETIC CHECKING

Philippe Specht

Examiners:
Prof. Dr. Erika Ábrahám
Prof. Dr. Christina Büsing

Additional Advisor:
Jasper Nalbach M.Sc.

Aachen, April 12, 2023

Acknowledgment

First, I would like to thank professor Erika Ábrahám for giving me the opportunity for this thesis and teaching me the required theoretical background. In addition, I would like to thank Jasper Nalbach for answering my every question in detail. Also, I would like to thank professor Christina Büsing for examining this thesis.

Last but not least, I would like to thank my family and friends, especially my parents and girlfriend, for their unconditional support.

Abstract

Satisfiability modulo theories (SMT) solving deals with checking the satisfiability of a given formula over a distinct *first-order logic* theory. This thesis proposes improvements to *incremental linearization* (IL) [Irf18][CGI+18], a procedure that can be used for SMT-solving the quantifier-free *non-linear real arithmetic* (NRA).

The value of this approach stems from its *incompleteness*. In this case, meaning that it is not guaranteed to terminate for every input. This distances the approach from complete NRA solvers that are guaranteed to be exponential in worst-case runtime.

Incremental linearization works by first abstracting the non-linear input formula into a linear one. The abstracted formula is then iteratively checked for satisfiability using a solver for *linear real arithmetic* (LRA). The result can possibly be transferred to the original, non-linear input. If not, the abstracted formula is refined by conjuncting linear formulas to it. For that, different types of formulas that we refer to as *axioms* can be utilized.

As the main contribution of this thesis, some existing axioms are adapted and removed. Most notably, however, a new type of axiom is introduced. We call it the *secant axiom*. It is intended as a (partial) counterpart to the existing tangent plane axiom.

Also, heuristics are introduced that specify how the initial abstraction is created. These are motivated by different ways of abstracting, leading to different possibilities for refinement.

Incremental Linearization has been implemented in SMT-RAT [smt] with the proposed changes. Experimental evaluation shows that the heuristics for initial abstraction do not decrease overall runtime. However, a significant, overall decrease in runtime was achieved by using the secant axiom.

Contents

Chapter 1

Introduction

In theoretical computer science, *satisfiablity modulo theories* (SMT) refers to the problem of checking whether there exists an assignment to the variables of a *first-order logic* formula or not. Or in other words, checking its satisfiability. To tackle this problem, SMT solvers are specified that can check the satisfiability of any formula for a specific first-order logic theory, or fragment thereof.

A main application for these solvers is program testing, analysis, and verification [BdM14]. Tools for these tasks are a critical factor in ensuring the correct functionality of software. In doing so, they guarantee safety and security in the software's usage. A theory fragment used in this context is the quantifier-free *non-linear real arithmetic* (NRA). Formulas over this theory fragment are Boolean combinations of (in)equalities that compare a multivariate, non-linear polynomial to zero.

Among other things, *incremental linearization* (IL) [Irf18][CGI+18] can be used to constitute a SMT solver for NRA. Such a solver initially abstracts the non-linear input formula to a linear one. Then, this abstraction is iteratively checked for satisfiability using a solver for quantifier-free *linear real arithmetic* (LRA). The result of this check can possibly be transferred to the original, non-linear formula. If this cannot be done, the abstracted formula is refined by conjuncting linear formulas. This refinement can be made using different types of formulas that we refer to as *axioms*.

The potential of this approach lies in the fact that it is *incomplete*. In this particular case, this means that the solver *can* return whether the input formula is satisfied or not. Nevertheless, it can also run ad infinitum without ever giving an answer. While this might sound grim, it can be seen as a trade-off for the runtime of a complete NRA solver, which is proven to have a worst-case exponential runtime. Of course, not terminating is worse! However, IL's main complexity comes from the utilized LRA solver, which can run in polynomial time. Thus, there is potential for the approach to solve some otherwise hard problems very quickly.

In this thesis, we propose changes that aim to improve IL over its initial form as presented in [Irf18]. More precisely, these changes aim to improve the refinement since, after the initial LRA check of the abstraction, it is the only thing that can change the procedure's outcome.

For one, we remove and modify some existing axioms. Above all, however, we introduce a new axiom that we call the *secant axiom*. It is intended as a (partial) counterpart to the very impactful, already existing tangent plane axiom. Also, we will specify different heuristics for creating the initial abstraction. That is because

different ways of abstracting can create different opportunities for refinement.

We implemented a NRA solver based on IL in SMT-RAT [smt] with these changes in place. Experimental evaluation shows that usage of the heuristics for initial abstraction did not lead to a decrease in overall runtime. The introduction of the secant axiom on the other hand, did exactly that with the decrease even being quite significant.

The rest of this thesis is structured as follows. Chapter 2 formally specifies SMT-solving for NRA and LRA. Also, IL is introduced as in [Irf18]. Then, the aforementioned changes to IL are proposed in Chapter 3. The experimental evaluation of IL can be found in Chapter 4. To conclude, a summary and ideas for future work are given in Chapter 5.

Chapter 2

Preliminaries

We will first introduce the concept of SMT-solving for the theories of NRA and LRA[1]. Then, we will present incremental linearization (IL) as proposed in [Irf18], which is the foundation that this thesis builds upon.

2.1 SMT-Solving for NRA and LRA

Generally, the problem of SMT-solving can be formulated as proving whether there exists an *assignment* that *satisfies* a given *formula* over a given first-order logic theory (or not). The theory for which we want to conduct SMT solving is quantifier-free *non-linear* real arithmetic (NRA). However, the theory of quantifier-free *linear* real arithmetic (LRA) will also be introduced since IL essentially relies on reducing NRA to LRA.

In the following, we will define the terms we used above in order to specify the problem of SMT solving for NRA/LRA, or short NRA/LRA-solving. We begin by defining the syntax of formulas, starting with their atomic components.

Definition 2.1.1 ((Linear) Constraint)**.** *Let $R \in \{\mathbb{Z}, \mathbb{Q}\}$ be a ring, $p \in R[x_1, \ldots, x_n]$ a polynomial and $\triangledown \in \{=, \neq, \geq, \leq, >, <\}$ a comparison operator. A constraint C over $R[x_1, \ldots, x_n]$ is of the form $p \triangledown 0$.*

C is linear if p is linear, i.e. p can be written as $\sum_{i=1}^{n} c_i \cdot x_i$ for some $c_1, \ldots, c_n \in R$.

Note that the multiplication of a constraint over $\mathbb{Q}[x_1, \ldots, x_n]$ with the product of the denominators of all its coefficients results in an equivalent constraint over $\mathbb{Z}[x_1, \ldots, x_n]$. Throughout this thesis, we will therefore consider polynomials in $\mathbb{Z}[x_1, \ldots, x_n]$, even though polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$ can be used equivalently. Now, constraints can be combined into a formula as follows.

Definition 2.1.2 (Formula, Clause)**.** *A NRA formula φ in CNF is of the form $\bigwedge_{i \in N} \bigvee_{j \in M} C_{ij}$ where N and M are finite index sets and C_{ij} is a constraint over $\mathbb{Z}[x_1, \ldots, x_n]$. Then for each $i \in N$, $\bigvee_{j \in M} C_{ij}$ is a clause. φ can be represented in clausal form as $\{\bigcup_{j \in M} \{C_{ij}\} \mid i \in N\}$.*

If all constraints in φ are linear, then it is (also) a LRA formula.

[1] Parts of Section 2.1 are adapted or directly taken from [Spe20]

Example 2.1.1. *φ_1 is a NRA formula in CNF in constraints over $\mathbb{Z}[x_1,x_2]$*

$$\varphi_1 = \underbrace{(\underbrace{3x_1^2x_2 + x_1x_2 - 1 > 0}_{a\ constraint} \vee x_1 - 5x_2 \leq 0)}_{a\ clause} \wedge (4x_1 + 7 \leq 0).$$

In clausal form

$$\varphi_1 = \{\{3x_1^2x_2 + x_1x_2 - 1 > 0, x_1 - 5x_2 \leq 0\}, \{4x_1 + 7 \leq 0\}\}.$$

φ_2 is a LRA formula in CNF in constraints over $\mathbb{Z}[x_1,x_2,z_1,z_2]$

$$\varphi_2 = (3z_2 + z_1 - 1 > 0 \vee x_1 - 5x_2 \leq 0) \wedge (4x_1 + 7 \leq 0).$$

Adding semantics, an *assignment* for a NRA/LRA formula, is a mapping of all contained variables to real values. The formula's truth value under the assignment can then be evaluated by replacing the variables with the values, followed by arithmetic, and then Boolean evaluation.

Example 2.1.2. *φ_1 from Example 2.1.1 is evaluated under the assignment $[x_1 \mapsto 1, x_2 \mapsto 2]$ as follows*

$$\begin{aligned}\varphi_1[x_1 \mapsto 1, x_2 \mapsto 2] &= (3 \cdot 1^2 \cdot 2 + 1 \cdot 2 - 1 > 0 \vee 1 - 5 \cdot 2 \leq 0) \wedge (4 \cdot 1 + 7 \leq 0) \\ &= (7 > 0 \vee -9 \leq 0) \wedge (11 \leq 0) \\ &= (\mathit{true} \vee \mathit{true}) \wedge \mathit{false} \\ &= \mathit{false}\end{aligned}$$

Since φ_1 evaluates to false, we say that the assignment does not satisfy the formula.

Now, the problem of SMT-solving for NRA/LRA can be formulated as follows: Given a NRA/LRA formula φ in variables $x_1, \ldots, x_n$. Is φ satisfiable? I.e., is there an assignment $a_1, \ldots, a_n \in \mathbb{R}$ for variables $x_1, \ldots, x_n$ so that φ evaluates to *true*?

2.2 Incremental Linearization for NRA-Solving

For NRA-solving as defined above, *incremental linearization* (IL) can be used. This approach was originally introduced for satisfiability and verification modulo theories for NRA as well as for transcendental functions by Irfan et al. [Irf18][CGI+18]. We will only focus on IL for NRA-solving though. Note that there have already been attempts at improving IL for NRA-solving [Zam19]. However, this thesis is independent of that. In the following, we present IL for NRA-solving as in [Irf18].

The overall procedure is shown in Algorithm 1. Its input and output conform with the problem formulation for NRA-solving. In the following sections, we will look at the algorithm along the occurring function calls.

2.2.1 Initial Abstraction from NRA to LRA

The first step of Algorithm 1 is to call INITIAL-ABSTRACTION (l.1). This function derives a linear formula φ from the non-linear input ψ that we call the *(linear) abstraction of ψ*. The idea to construct said abstraction is to replace every monomial

Algorithm 1: NRA-solve using IL (adapted from [Irf18])

```
   Input  : a NRA formula ψ in CNF
   Output: a Boolean res
1  φ := INITIAL-ABSTRACTION(ψ)
2  Γ := ∅
3  while true do
4  |  (res, μ) := LRA-CHECK(φ ∧ ⋀Γ)
5  |  if not res then
6  |  |  return res
7  |  res := NRA-LIFT(ψ, φ, μ)
8  |  if res then
9  |  |  return res
10 |  Γ' := REFINE(ψ, φ, μ)
11 |  Γ := Γ ∪ Γ'
```

$m = c \cdot x_1^{i_1} \cdot \ldots x_n^{i_n}$ that occurs in ψ with $m' = c \cdot z$ using a *fresh* variable z. Thereby, fresh means that it is newly defined and previously unused.

This is done by iteratively abstracting the products of two variables into a fresh one. In the base case, for example, the product x_1x_2 can be abstracted by z_1. Note that a more sophisticated term, $x_1^2x_2$ could then be replaced by x_1z_1. However, this would then have to be abstracted again into a fresh variable z_2 since x_1z_1 is still non-linear.

Example 2.2.1. *The non-linear formula ψ over $\mathbb{Z}[x_1,x_2]$*

$$\psi = 3\mathbf{x_1^2x_2} + \mathbf{x_1x_2} - 1 > 0 \vee x_1 - 5x_2 \leq 0$$

could be abstracted to a linear formula φ over $\mathbb{Z}[x_1,x_2,z_1,z_2]$ as follows

$$\begin{aligned}\varphi &\overset{z_1=x_1\cdot x_2}{\rightarrow} 3x_1\mathbf{z_1} + \mathbf{z_1} - 1 > 0 \vee x_1 - 5x_2 \leq 0\\ &\overset{z_2=x_1\cdot z_1}{\rightarrow} 3\mathbf{z_2} + z_1 - 1 > 0 \vee x_1 - 5x_2 \leq 0.\end{aligned}$$

It should also be clear that this process of abstracting is not distinct. For example, considering $x_1^2x_2$ as in Example 2.2.1, we could abstract x_1x_1 by z_1' and then $z_1'x_2$ by z_2' instead. Therefore, we will present heuristics for this process of abstracting in Section 3.1. We also experimentally evaluate how these heuristics affect the performance of IL in Section 4.2.4.

After the initial abstraction has been created in Algorithm 1, merely Γ is initialized to be the empty set (l.2) before entering the loop that constitutes the main functionality of the algorithm.

2.2.2 LRA-Solving the Abstracted Formula

The first step of the loop is checking φ for satisfiability via the function LRA-CHECK (l.4). This is done using an LRA solver. The solver in our implementation uses the simplex algorithm in the DPLL(T) framework [Nal20]. However, any complete solver can be used.

The solver returns whether the formula is satisfiable via the Boolean *res*. If *res* is *false*, the algorithm will terminate and return that ψ is unsatisfiable since its over-approximation φ is already unsatisfiable (ll.5-6). Otherwise, φ is satisfiable and the LRA solver will additionally return an assignment μ that satisfies φ.

2.2.3 Lift LRA Assignment to NRA

The assignment μ is then used in NRA-LIFT (l.7). This function tries to deduce an assignment μ' from μ so that it satisfies ψ. If this is successful, the algorithm will return that ψ is satisfiable and terminate (ll.8-9).

The easiest approach would be to set $\mu' := \mu$. Then the only remaining step is to check, *for every* variable z that abstracted the product of some variables x_1 and x_2 in INITIAL-ABSTRACTION, whether $\mu'(z) = \mu'(x_1) \cdot \mu'(x_2)$ holds.

Note that it might happen that variables x_1 and/or x_2 do not occur in φ and therefore are not assigned in μ/μ'. This is due to the fact that during INITIAL-ABSTRACTION, variables can vanish from the formula if all their occurrences are replaced. In that case, the assignment(s) have to be guessed. If possible, this should be done so that $\mu'(z) = \mu'(x_1) \cdot \mu'(x_2)$ holds.

In practice, however, this approach is unlikely to succeed on a consistent basis. That is because it merely checks whether the LRA assignment coincidentally is also consistent with the non-linear multiplications that have not been considered in the creation of said assignment.

To increase the chances of success, the authors of [Irf18] proposed a more sophisticated approach that aims to under-approximate ψ. In detail, a new linear formula φ' is derived from φ. It consists of two components. The first is the *truth assignment* of φ which can be seen in the first line of Equation (2.1). It is a conjunction of all constraints contained in φ where all constraints that evaluate to *false* under μ are negated. The second component are the *multiplication-line constraints* which can be seen in the second line of Equation (2.1). These constraints essentially assume all abstracted multiplications to be linear with respect to the assignment of x or y.

$$\begin{aligned}\varphi' = & \bigwedge_{\substack{C \in constraints(\varphi) \\ s.t.\ \mu \text{ satisfies } C}} C \quad \wedge \bigwedge_{\substack{C \in constraints(\varphi) \\ s.t.\ \mu \text{ does not satisfy } C}} \neg C \quad \wedge \\ & \bigwedge_{z = x \cdot y \text{ via initial abstraction}} \left(\begin{array}{l} (x = \mu(x) \wedge z = \mu(x) \cdot y) \vee \\ (y = \mu(y) \wedge z = \mu(y) \cdot x) \end{array} \right)\end{aligned} \tag{2.1}$$

Now, φ' under-approximates the non-linear formula ψ. This means that if φ' is satisfiable, ψ is satisfiable as well. Thus, NRA-LIFT returns the result of LRA-CHECK(φ') if this approach is used. Note, however, that since φ' is an under-approximation, if it is unsatisfiable, ψ can still be satisfiable.

Coming back to the algorithm, if the satisfiability of ψ could not be proven in ll.7-9, the next step is to refine φ.

2.2.4 Refinement in IL using Axioms

This refinement is done via the function REFINE (l.10). It returns a set of LRA formulas Γ' that is added to the continuously maintained set of formulas Γ (l. 11). From a theory perspective, the added formulas refine φ since they are conjuncted to

Monotonicity axioms:
$(|x_1| \leq |x_2| \land |y_1| \leq |y_2|) \rightarrow |z_1| \leq |z_2|$
$(|x_1| < |x_2| \land |y_1| \leq |y_2| \land y_2 \neq 0) \rightarrow |z_1| < |z_2|$
$(|x_1| \leq |x_2| \land |y_1| < |y_2| \land x_2 \neq 0) \rightarrow |z_1| < |z_2|$

Congruence axiom:
$(x_1 = x_2 \land y_1 = y_2) \rightarrow z_1 = z_2$

Figure 2.1: Axioms specifying a property of the multiplication function w.r.t. abstracted multiplications $z_1 = x_1 \cdot y_1$ and $z_2 = x_2 \cdot y_2$. Thereby, $|\cdot|$ is used as a unary function symbol that maps to the absolute value of the input (adapted from [Irf18]).

Zero axiom:
$(x = 0 \lor y = 0) \leftrightarrow z = 0$

Sign axioms:
$((x > 0 \land y > 0) \lor (x < 0 \lor y < 0)) \leftrightarrow z > 0$
$((x > 0 \land y < 0) \lor (x < 0 \lor y > 0)) \leftrightarrow z < 0$

Tangent plane axiom:
$(x = a \rightarrow z = a \cdot y) \land$
$(y = b \rightarrow z = b \cdot x) \land$
$(((x < a \land y < b) \lor (x > a \land y > b)) \rightarrow z > b \cdot x + a \cdot y - a \cdot b) \land$
$(((x > a \land y < b) \lor (x < a \land y > b)) \rightarrow z < b \cdot x + a \cdot y - a \cdot b)$

Figure 2.2: Axioms bounding the values of an abstracted multiplication $z = x \cdot y$. Thereby, $a = \mu(x)$ and $b = \mu(y)$ (adapted from [Irf18]).

it in the LRA-CHECK (l.4). As a result, the overall algorithm can yield different results after each iteration of the loop.

The formulas that are created for refinement aim at linearly specifying the nonlinear multiplications that were abstracted in INTIAL-ABSTRACTION (l.1). For that, schemes of formulas are created that can be instantiated for a singular or pair of abstracted multiplications. We call these schemes *axioms* since they axiomatize properties of the multiplication(s).

The following axioms are adapted from [Irf18], where (in addition to LRA) uninterpreted functions are used for abstraction. Because of that, not all axioms can be adapted, and for the remaining ones, the notation changes slightly. However, the meaning of the remaining axioms is exactly as described in [Irf18]. Broadly speaking, the axioms either *specify a property of the multiplication function* or *bound the values of a multiplication* using the value zero or the assignment μ.

The monotonicity axioms and the congruence axiom fall into the first category and are displayed in Figure 2.1. They precisely specify the name-giving property for a pair of abstracted multiplications.

The second category consists of the zero axiom, sign axioms, and the tangent plane axiom which are shown in Figure 2.2. The zero axiom and the sign axioms are rather straightforward, specifying the multiplication of two variables with respect to the sign function. Thereby, the zero axiom covers the special case of zero. Note that

in [Irf18], what we refer to as sign axioms were called zero axioms as well. Also, the term sign axioms was used for a different property that we can not replicate here due to us not using uninterpreted functions.

A more involved axiom is the tangent plane axiom. Its name comes from the tangent plane, which is defined as $z = bx + ay - ab$ for a point (a, b). This plane has the special property that it intersects with the multiplication function $z = x \cdot y$ in two distinct lines on the plane's surface that cross at (a, b, ab).

In Figure 2.3c, the multiplication function $z = x \cdot y$ and the tangent plane $z = 2x + 6y - 12$ for $(a, b) = (6, 2)$ are displayed. The multiplication function is also explicitly shown in Figure 2.3a-b for reference. In the plot, the above-mentioned lines and intersection at $(a, b, ab) = (6, 2, 12)$ can clearly be seen.

If we project the plot along the z-axis, we can observe that the x-y-domain is split orthogonally into 4 quadrants I_{ab}-IV_{ab} with respect to the projected lines, see Figure 2.3d. Now, the premises of the tangent plane axiom as given in Figure 2.2 can be concluded from this split.

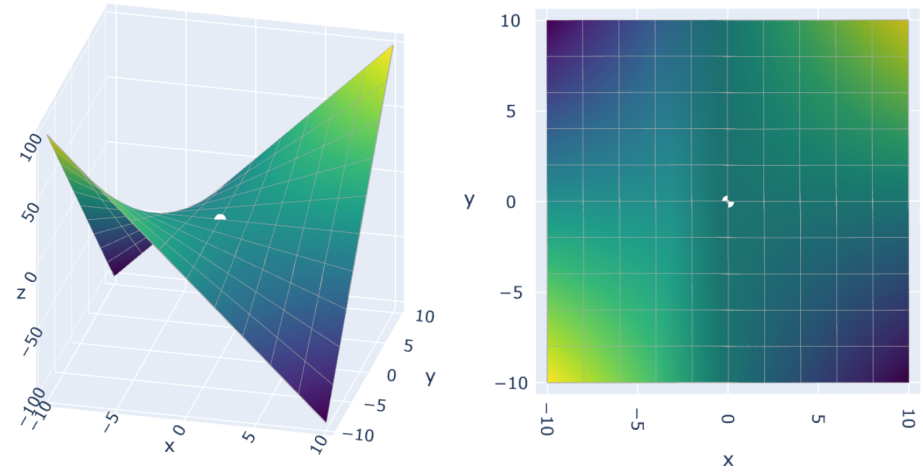
That is, the first two implications of the axiom describe the lines that separate the quadrants, which are located at $x = a$ and $y = b$. Additionally, the conjunctions contained in the premises of the third and fourth implications describe exactly the four quadrants. From visualization, we can also note that for quadrants I_{ab} and III_{ab}, the tangent plane is always below the multiplication function. Therefore, it is used as a lower bound in the third implication. Analogously, the tangent plane is used as an upper bound for quadrants II_{ab} and IV_{ab} in the fourth implication.

Now, the REFINE function returns at least one instance of an axiom that is not satisfied under the current assignment μ via Γ. How many and which instances are created and then returned is a question of heuristic. In [Irf18], an *eager* approach was implemented and a *lazy* one proposed for future work. Thereby, eager means that all axioms are instantiated for all individuals or pairs of abstracted multiplications, and the ones unsatisfied by μ are returned. Lazy, on the other hand, means that exactly one instance of an axiom that is unsatisfied by μ is returned. In concrete implementation, it has to be specified with respect to which axiom and abstracted multiplication(s) this instance is created.

In our own implementation, we will compare approaches that differ in eagerness. The concrete heuristics are presented in Section 4.1 and the experimental results regarding them can be found in Section 4.2.5.

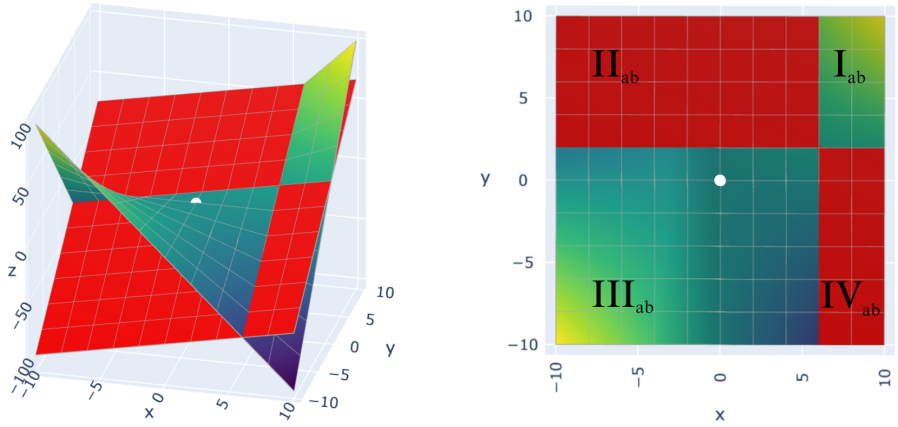
Overall, the REFINE method is a critical factor for the success of IL for NRA-solving. That is because after the initial calls to LRA-CHECK and NRA-LIFT, it is the only method that can change the algorithm's outcome. Therefore, the changes that are proposed in the following Chapter 3 all aim at improving the refinement.

As a final remark, it should be mentioned that IL for NRA-solving is *incomplete*. This means that, no matter the (linear) refinement strategy, the algorithm will either return unknown or run ad infinitum for some inputs. This can be seen as a trade-off for the runtime of complete NRA solvers, which is proven to be (worst-case) exponential in the number of variables in the input formula. IL's main complexity, on the other hand, lies in the LRA solver that is used in the main loop, which can be polynomial in runtime. So if IL solves a lot of problems, especially harder ones, in only a few iterations, it might be beneficial in application. We will see evidence that this is the case in Section 5.1.2.



(a) $z = x \cdot y$

(b) Projection to x- and y-axis



(c) $z = x \cdot y$ and tangent plane

(d) Projection to x- and y-axis

Figure 2.3: Plot of multiplication function $z = x \cdot y$ and tangent plane w.r.t. the point $(a, b, ab) = (6, 2, 12)$, i.e. $z = 2x + 6y - 12$ (adapted from [Irf18]).

Chapter 3

Improving Incremental Linearization

The improvements to IL as given in Algorithm 1 that we propose can be divided into two parts. The first one deals with introducing different heuristics for abstraction that can be used to implement INITIAL-ABSTRACTION. In the second part, REFINE will be improved by modifying and removing some axioms as well as introducing a new axiom.

3.1 Heuristics for the Initial Abstraction

Our first contribution is specifying two different ways of linearizing the polynomials contained in ψ, i.e., the input of Algorithm 1. Technically, this can also be seen as different specifications of the function INITIAL-ABSTRACTION which is called in line 1 of Algorithm 1. We will present both in the following sections.

3.1.1 Heuristic: Optimal w.r.t. Number of Fresh Variables

We named the first heuristic *optimal w.r.t. number of fresh variables*. That is because it minimizes the amount of abstraction that is done by minimizing the number of fresh variables/abstracted multiplications that are introduced.

This characteristic also motivates the heuristic. Fewer abstracted multiplications give fewer possibilities for refinement in REFINE of Algorithm 2. For example, the zero axiom and sign axiom can only be instantiated once per abstracted multiplications. Thus, minimizing this target directly translates to fewer axiom instantiations that could possibly be created. Therefore, we see potential for this heuristic to lead to faster termination. Algorithm 2 specifies the *optimal w.r.t. number of fresh variables* heuristic.

The algorithm is commented to improve understanding. Thus, we will only give a general description. In lines 1 to 7, initialization is done. Most notably, ψ is converted into a notation that is more accessible for our purposes (ll.1-3) and the map M is created in order to save the abstracted multiplications. Thereby, the unordered pair relation $\otimes$ is defined as follows:

Definition 3.1.1 (Unordered Pair Relation $\otimes$)**.** *For any sets S_1,S_2, let*

$$S_1 \otimes S_2 := \{[s_1,\, s_2] \mid s_1 \in S_1,\, s_2 \in S_2\}$$

be the set containing lists of all combinations of elements in S_1 and S_2.

Then the loop that progressively replaces variables begins. In lines 9 to 19, the pair of variables that will be replaced by a fresh variable is searched for. This is done by looking for the pair of variables among all pairs that (as a pair) occurs most often across all monomials. Then, all pairwise occurrences in the current representation of the formula S are replaced by a single occurrence of a fresh variable in lines 22 to 26.

It can also happen that no such pair is found in lines 9 to 19 due to no more pairwise occurrences of variables left in S. This translates to the underlying formula now being linear. Thus, the loop would be broken in line 21, and the linear abstraction φ can be reconstructed from S and ψ in line 32.

To conclude, we present a run of Algorithm 2 on the formula from Example 2.2.1.

Example 3.1.1. *The input formula is*

$$\psi = (3x_1^2x_2 + x_1x_2 - 1 > 0 \vee x_1 - 5x_2 \leq 0) \wedge (4x_1 + 7 \leq 0).$$

In lines 1-3, it is converted to the new representation

$$\overline{S} = [[x_1,\, x_1,\, x_2],\, [x_1,x_2],\, [x_1],\, [x_2],\, [x_1]],$$

essentially removing Boolean and arithmetic structure that is not relevant for abstraction.

After further initialization, the occurrence count (oc) of all pairs of variables in $\{x_1,\, x_2\} \otimes \{x_1,\, x_2\} = \{[x_1,\, x_2],\, [x_1,\, x_1],\, [x_2,\, x_2]\}$ is computed. The oc of $[x_1,\, x_2]$ is two since it occurs once in the first list in $\overline{S}$ and once again in the second list. Analogously, the oc of $[x_1,\, x_1]$ is one, and the oc of $[x_2,\, x_2]$ is zero. Since $[x_1,\, x_2]$ has the highest oc, it is chosen for abstraction.

Note that, while not happening here, the occurrence count can increase by more than one per list in S. For example, considering $[x_1,\, x_1,\, x_2,\, x_2]$ in S, the occurrence count of $[x_1,\, x_2]$ would be increased by two.

Now, all pairwise occurrences of x_1 and x_2 are replaced by the fresh variable z_1 in lines 23 to 27, resulting in

$$\overline{S} = [[x_1,\, z_1],\, [z_1],\, [x_1],\, [x_2],\, [x_1]].$$

In the next iteration, the oc of all elements of $\{x_1,\, x_2,\, z_1\} \otimes \{x_1,\, x_2,\, z_1\}$ is zero, except for $[x_1,\, z_1]$ which has an occurrence count of one. Replacement by the fresh variable z_2 results in

$$\overline{S} = [[z_2],\, [z_1],\, [x_1],\, [x_2],\, [x_1]].$$

Now, the oc of all pairs of variables is zero in the third iteration. Thus, the loop breaks and the linear formula φ is constructed from $\overline{S}$ using the structure of ψ so that

$$\varphi = (3z_2 + z_1 - 1 > 0 \vee x_1 - 5x_2 \leq 0) \wedge (4x_1 + 7 \leq 0).$$

This formula is then returned as the result of the algorithm.

Note that alongside the algorithm, the map

$$M : V_{all} \to V_{all} \otimes V_{all}, z_1 \mapsto [x_1,\, x_2], z_2 \mapsto [x_1,\, z_1]$$

where $V_{all} = \{x_1,\, x_2,\, z_1,\, z_2\}$ is also computed. It is not of relevance here, but it is necessary knowledge for the refinement in Algorithm 1.

Algorithm 2: Abstract ψ to LRA formula φ using heuristic *optimal*

Input : a NRA formula ψ in CNF
Output: a LRA formula φ

```
// S represents current state of formula during execution
// → 1:1 relation between monomials in ψ and elements of S
```
$S :=$ list of all non-constant monomials occurring in ψ in order of appearance
foreach *monomial* m *in* S **do**
  `// e.g. replace` $3x_1^2x_2$ `by` $[x_1, x_1, x_2]$
  $m :=$ list containing all variables in m, including multiple occurrences
`// keep track of all variables for later`
$V_{all} :=$ set of all variables occurring in S
`// keep track of variables that potentially need replacement`
$V_{tmp} :=$ set of all variables occurring in S
`// save which var abstracts multiplication of which two vars`
$M :=$ empty map $V_{all} \to V_{all} \otimes V_{all}$
`// greedily merge variables by occurrence count (oc)`
$i := 1$
while *true* **do**
  `// find pair of variables with maximal oc`
  $oc := 0$
  foreach $[v_1, v_2] \in V_{tmp} \otimes V_{tmp}$ **do**
    $oc_{tmp} := 0$
    foreach s *in* S **do**
      if $v1 = v2$ **then**
        oc_{tmp} += $\lfloor$(number of occurrences of v in s)$/2\rfloor$
      else
        oc_{tmp} += minimum number of occurrences of v_1 vs v_2 in s
    if $oc_{tmp} > oc$ **then**
      $oc := oc_{tmp}$
      $[w_1, w_2] := [v_1, v_2]$
  if $oc = 0$ **then**
    break
  `// create fresh` z_i`-variable and replace`
  $M(z_i) := [w_1, w_2]$
  $V_{all} := V_{all} \cup \{z_i\}$
  foreach s *in* S **do**
    while s *contains an occurrence of* w_1 *and* w_2 **do**
      `// add` z_i `and remove` w_1 `and` w_2 `exactly once`
      $s := s - w_1 - w_2 + z_i$
  `// clean up` V_{tmp}
  if w_1 *not in any* s *in* S **then**
    $V_{tmp} := V_{tmp} \setminus \{w_1\}$
  if w_2 *not in any* s *in* S **then**
    $V_{tmp} := V_{tmp} \setminus \{w_2\}$
  i += 1
`// All lists in` S `contain only one element`
$\varphi := \psi$ where vars of monomials are replaced by the corresponding variable in S
return φ

3.1.2 Heuristic: Exponents First

Another heuristic that we named *exponents first* follows a different idea. In particular, this idea is to first abstract the squares of single variables. Thus, with respect to the input formula, the present exponents are abstracted first.

Example 3.1.2. *The monomial $x_1^2 x_2^5$ could be abstracted as follows*

$$x_1^2 x_2^5 \overset{z_1 = x_1 \cdot x_1}{\rightarrow} z_1 x_2^5 \overset{z_2 = x_2 \cdot x_2}{\rightarrow} z_1 z_2^2 x_2 \overset{z_3 = z_2 \cdot z_2}{\rightarrow} z_1 z_3 x_2.$$

We suggest applying the *optimal w.r.t. number of fresh variables* heuristic to abstract the remaining non-linear monomials without exponents.

The advantage of this heuristic is that for abstracted squares of variables, the instances of axioms for refinement are less complex. For example, the tangent plane axiom (see Figure 2.2) can be instantiated for an abstracted multiplication $z = x \cdot y$. When this multiplication is a square ($x = y$) the axiom simplifies to

$$(x = a \rightarrow z = ax) \wedge ((x < a \vee x > a) \rightarrow z > 2ax - a^2).$$

Algorithm 3 shows a possible implementation of the heuristic. It is an adaptation of Algorithm 2, which was presented in the previous Section 3.1.1. Essentially, lines 10 to 19 are changed to only consider abstractions of squares of single variables. As an effect, no monomial in φ has an exponent left after line 13 of Algorithm 3. Abstracting these remaining monomials is then done by calling Algorithm 2 in line 14.

To show where the *exponents first* (EF) heuristic differs from the *optimal w.r.t. number of fresh variables* (OPT) heuristic, we restrict ourselves to a single polynomial.

Example 3.1.3. *EF and OPT abstract the polynomial $x_1^2 x_2 + x_1 x_2$ as follows*

$$\begin{aligned} EF &: x_1^2 x_2 + x_1 x_2 \overset{z_1 = x_1 \cdot x_1}{\rightarrow} z_1 x_2 + x_1 x_2 \overset{z_2 = z_1 \cdot x_2}{\rightarrow} z_2 + x_1 x_2 \overset{z_3 = x_1 \cdot x_2}{\rightarrow} z_2 + z_3 \\ OPT &: x_1^2 x_2 + x_1 x_2 \overset{z_1 = x_1 \cdot x_2}{\rightarrow} x_1 z_1 + z_1 \overset{z_2 = x_1 \cdot z_1}{\rightarrow} z_2 + z_1 \end{aligned}$$

On the one hand, this shows that EF can use more fresh variables for abstraction. In this case, EF introduces three new variables, while OPT only introduces two. On the other hand, only EF has done the abstraction $z_1 = x_1 \cdot x_1$ for which refinement will be easier in Algorithm 1.

To summarize this section, we presented heuristics for INITIAL-ABSTRACTION with solid evidence for their usefulness in application. Additionally, we specified them proficiently enough for implementation, which has not previously been done. The impact on performance of these heuristics is looked into experimentally in Section 4.2.4.

Algorithm 3: Abstract ψ to LRA formula φ using heuristic *exponents first*, implemented as an adaptation of Algorithm 2

```
   Input  : a NRA formula ψ in CNF
   Output: a LRA formula φ
   ...
   // greedily merge variables by occurrence count (oc)
 1 i := 1
 2 while true do
       // find pair of variables with maximal oc
 3     oc := 0
 4     foreach v ∈ V_tmp do
 5         oc_tmp := 0
 6         foreach s in S do
 7             oc_tmp += ⌊(number of occurences of v in s)/2⌋
 8         if oc_tmp > oc then
 9             oc := oc_tmp
10             [w_1, w_2] := [v, v]
11     if oc = 0 then
12         break
       // create fresh z_i-variable and replace
       ...
   // All lists in S contain no variable more than once
13 φ := ψ where vars of monomials are replaced by the corresponding variable in S
14 return INITIAL-ABSTRACTION-OPTIMAL(φ)
```

3.2 Improving the Refinement

In the following, we delete, modify, and introduce new axioms in order to improve the refinement process constituted by REFINE of Algorithm 1. The result of these changes is displayed in Figure 3.3 which is therefore a summary of this section's results. In the following, the changes are explained in detail.

3.2.1 Optimizing Existing axioms

We propose some changes to the existing axioms that were presented in Figure 2.1 and Figure 2.2.

Adapting Axioms to Our Implementation

We made some minor, general adjustments for IL to work better in our implementation. For one, we made all inequalities non-strict. This is useful in our case since we use the simplex algorithm [Nal20] in our LRA solver, which requires additional effort for dealing with strict inequalities [NÁK21]. From a theoretical perspective, this creates redundancy between some parts of the same formula. However, we do not see any potential for that slowing down the algorithm.

Also, we explicitly implemented *squared cases* for some axioms, i.e., when $x = y$ for abstracted multiplication $z = x \cdot y$. This by itself will most likely not cause a significant speedup since a sophisticated implementation of IL could implicitly simplify the standard case to the squared case formula every time the squared case occurs.

Nevertheless, this synergizes with our implementation of Algorithm 2/3 using two distinct maps to explicitly buffer abstracted multiplications that are squares. Additional synergy comes with using the *exponents first* heuristic.

Adapting the tangent plane axiom

The tangent plane axiom is slightly modified, which is expected to enhance its utility. For that, we replace some implications contained in the axiom with equivalences.

$$\begin{aligned}&(x = a \rightarrow z = ay) \ \wedge \ (y = b \rightarrow z = bx) \ \wedge \\ &(((x \geq a \wedge y \leq b) \vee (x \leq a \wedge y \geq b)) \leftrightarrow z \leq bx + ay - ab) \ \wedge \\ &(((x \leq a \wedge y \leq b) \vee (x \geq a \wedge y \geq b)) \leftrightarrow z \geq bx + ay - ab)\end{aligned}$$

This is possible because the tangent plane *exactly* separates the projection of $z = x \cdot y$ to x and y into the four quadrants I_{ab}-IV_{ab}, as visualized in Figure 2.3d. And the description of these quadrants is exactly formalized on the left-hand side of the now-equivalences. For example, $x \geq a \wedge y \geq b$ for quadrant I_{ab}.

Practically, this modification realizes that an instance of the tangent plane axiom cannot simply be satisfied by violating what had previously been the preconditions of the implications. Theoretically, it can also be seen as the addition of an entire new axiom. That is because previously, the tangent plane axiom bounded z based on x and y. Now, it also bounds x and y based on z.

Removing The Congruence and Monotonicity Axiom

We remove the congruence and the monotonicity axiom from the refinement process since neither, under any circumstance, *guarantees* an improvement of the assignment in the following iteration. Thereby, we consider improving in the sense of forcing a new assignment so that the multiplications $x_1 \cdot y_1 = z_1$ and/or $x_2 \cdot y_2 = z_2$ are enforced or at least closer to it. We substantiate this claim, starting with the congruence axiom.

Let x_1, x_2, y_1, y_2, z_1 and z_2 be variables with the current assignments of a_1, a_2, b_1, b_2, c_1 and c_2. The axiom is only violated if $a_1 = a_2$, $b_1 = b_2$ and $c_1 \neq c_2$. The only useful correction to the assignment that the axiom could cause is that if w.l.o.g. $a_1 \cdot b_1 = c_1$ and $a_2 \cdot b_2 \neq c_2$. Then it is *possible* that c_1 will be assigned the value of c_2 in the next iteration. However, reassigning a_1, a_2, b_1 and/or b_2 also satisfies the axiom. Even worse, the axiom can also be satisfied by assigning the value of c_1 to c_2.

Very similar arguments can be made for the monotonicity axiom. Under the same assumptions, the axiom is only violated if $|a_1| \leq |a_2|$, $|b_1| \leq |b_2|$ and $|c_1| > |c_2|$. If now, w.l.o.g., $a_1 \cdot b_1 \neq c_1$, for example, assigning c_2 the value of c_1 in the next iteration will satisfy the axiom but will not be an improvement. Otherwise, if w.l.o.g. $a_1 \cdot b_1 = c_1$, it is *possible* that c_2 will be assigned to a better-fitting value in the next iteration. However, it is also possible to reassign c_1 in order to satisfy the axiom, which is the opposite of an improvement.

We substantiate the correctness of these statements experimentally in Section 4.2.6.

3.2.2 Introducing the New Secant Axiom

We now present a (partial) counterpart to the tangent plane axiom in the form of the *secant axiom*. Let us first look at the squared case of this axiom, in order to transfer the idea to the standard case later on.

Squared Case of the Secant Axiom

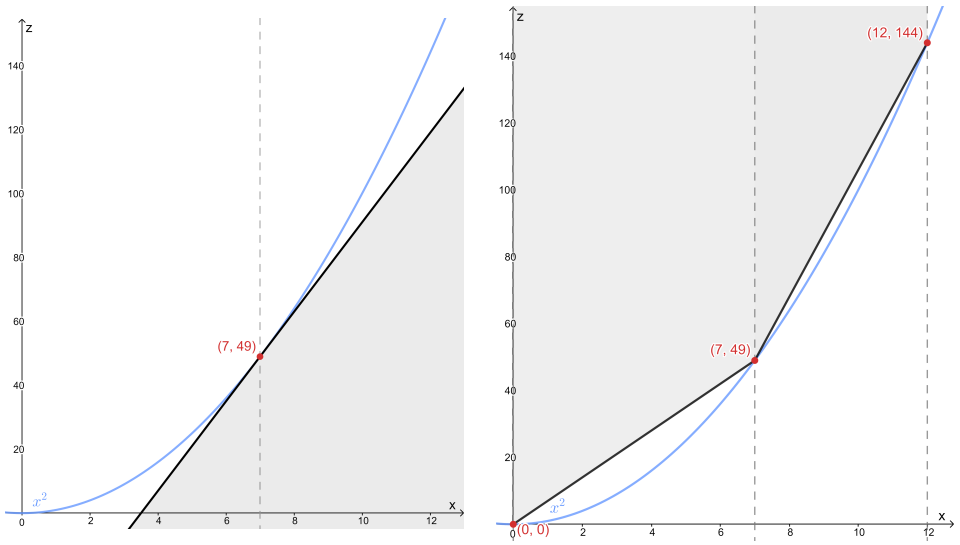
The squared case of the tangent plane axiom gives a lower bound to an abstracted multiplication $z = x \cdot x$ with respect to a, the currently assigned value to x. We visualize this exemplary in Figure 3.1a for $a = 7$. It can be seen that the axiom creates a tangent to $x \cdot x$ through the point (7, 49). In the context of the axiom, this tangent is a lower bound. So the grayed out area underneath it describes value combinations of x and z for which the axiom evaluates to *false*. We now design an axiom that gives an upper bound for the same multiplication.

Since $x \cdot x$ is convex, it is impossible to define a single, linear bound over the entire domain as done for the tangent plane axiom. Instead, we have to restrict ourselves to some interval on the domain of x. Because there is no clear choice for a good interval size, we determine the width using a parameter $d > 0$ that can be heuristically chosen in implementation.

The interval with respect to d over which we will define our axiom is then $[0, a+d]$ for $a > 0$ and $[a - d, 0]$ for $a < 0$. We create two different axioms to make precisely this case distinction. In notation, we differentiate the axioms via the quadrants of the coordinate system that they are defined for. In this case, these are I ($a > 0$) and II ($a < 0$). The restriction to these intervals is encoded into the premises contained in the axioms, see Figure 3.3.

Note that we also explicitly define an axiom for the case that $a = 0$, i.e., case 0. Essentially, this axiom is the conjunction of I and II for $a = 0$. Therefore, it covers the interval $[-d, d]$. We will not further address this axiom here since it can be seen as a special case derived from I and II.

Now to the conclusions contained in I and II. To have a good approximation without over-complicating things, we create two secants to bound the multiplication. The first one always intersects $x \cdot x$ at (0, 0) and (a, a^2). And the second one intersects



(a) Tangent plane axiom

(b) Secant axiom

Figure 3.1: Combinations of x- and z-values that are excluded by different axioms for $a = 7$ in the squared case.

the function at (a, a^2) and $(a+d, (a+d)^2)$ or $(a-d, (a-d)^2)$ respectively. This way, precisely the above-mentioned intervals on x are covered. We visualize this exemplary for I in Figure 3.1b with $a = 7$ and $d = 5$. The grayed-out area over the secants shows value combinations of x and z, for which I evaluates to *false*.

Formally deriving these secants is relatively simple. Each secant can be calculated by inserting the two intersection points from above into the general formula for a line $mx + b = y$ and solving the resulting system of equations for m and b. For example, considering I, using (a, a^2) and $(a + d, (a + d)^2)$, the secant $(2a + d)x + (-a^2 - da)$ can be derived.

As briefly mentioned, choosing the parameter d is a question of heuristic. We do not see any indicators for a choice that is beneficial on a theoretically founded level. Thus, for our implementation, we will test the values 1, 10, 100 and 1000 experimentally in Section 4.2.3.

Choosing d dynamically with respect to a could enhance the axiom's utility, since for a static d, the approximation gets more narrow with a growing towards $\infty/-\infty$. That is because $x \cdot x$ becomes increasingly steeper with x tending towards its limits. However, we leave this for future work.

To conclude the squared case of the secant axiom, we prove its *correctness*. Thereby, we mean that Algorithm 1 stays correct when it is utilized. The only way an axiom can potentially violate this is if it changes the fact that $\varphi \wedge \bigwedge \Gamma$ *over-approximates* ψ, i.e., any assignment that satisfies ψ also satisfies $\varphi \wedge \bigwedge \Gamma$. We conclude the following:

Lemma 3.2.1. *An axiom is correct w.r.t. Algorithm 1 if it over-approximates* ψ.

Proof. Let ψ be a non-linear formula and $\varphi \wedge \bigwedge \Gamma$ be a linear over-approximation of ψ as in Algorithm 1. Also, let ϑ be an axiom, i.e., a LRA formula, so that ϑ over-approximates ψ.

Assume there is a satisfying assignment for ψ. In order for the axiom to be correct w.r.t. Algorithm 1, this assignment needs to satisfy $\varphi \wedge \bigwedge \Gamma \wedge \vartheta$. Now, since both $\varphi \wedge \bigwedge \Gamma$ and ϑ over-approximate ψ, the assumed assignment satisfies both of them and thus also their conjunction. □

Theorem 3.2.2. *The secant axioms I and II (squared case) as displayed in Figure 3.3 are correct w.r.t. Algorithm 1.*

Proof. We only provide a proof for I since II can be proven analogously. Let there be a satisfying assignment for ψ. As stated in Lemma 3.2.1, it is to be shown that this assignment also satisfies I. Now, per Algorithm 1, it holds that

$$\psi \equiv \varphi \wedge \bigwedge_{\substack{\overline{z}=\overline{x}\cdot\overline{y} \text{ via} \\ \text{initial abstraction}}} \overline{z} = \overline{x} \cdot \overline{y}.$$

Thus, the assignment satisfies $z = x \cdot x$, where x and z are the variables contained in I. Using this equality and extracting the boolean structure from I, we can reformulate what remains to be shown. Let $a > 0$ and $d > 0$,

1. if $0 \leq x \leq a$ then $x^2 \leq ax$ and
2. if $a \leq x \leq a + d$ then $x^2 \leq (2a + d)x + (-a^2 - da)$.

1. Let $0 \leq x \leq a$. Because $x \cdot x$ is strictly convex (trivial for x^2) and ax is linear, $x^2 \leq ax$ holds if it holds for the extreme values of the assumed domain for x. For $x = 0$, it follows that

$$\begin{aligned} & 0^2 \leq a \cdot 0 \\ \Leftrightarrow \quad & 0 \leq 0. \end{aligned}$$

For $x = a$, it follows that

$$\begin{aligned} & a^2 \leq a \cdot a \\ \Leftrightarrow \quad & 0 \leq 0. \end{aligned}$$

2. Analogously, let $a \leq x \leq a + d$ and insert the extreme values into the inequality $x^2 \leq (2a + d)x + (-a^2 - ad)$. For $x = a$, it follows that

$$\begin{aligned} & a^2 \leq (2a + d)a + (-a^2 - ad) \\ \Leftrightarrow \quad & a^2 \leq 2a^2 + ad - a^2 - ad \\ \Leftrightarrow \quad & a^2 \leq a^2 \\ \Leftrightarrow \quad & 0 \leq 0. \end{aligned}$$

For $x = a + d$, it follows that

$$\begin{aligned} & (a + d)^2 \leq (2a + d)(a + d) + (-a^2 - ad) \\ \Leftrightarrow \quad & a^2 + 2ad + d^2 \leq 2a^2 + ad + 2ad + d^2 - a^2 - ad \\ \Leftrightarrow \quad & a^2 + 2ad + d^2 \leq a^2 + 2ad + d^2 \\ \Leftrightarrow \quad & 0 \leq 0. \end{aligned}$$

□

Standard Case of the Secant Axiom

In the following, we propose an axiom that bounds an abstracted multiplication $z = x \cdot y$ with respect to the assignments a to x and b to y. These bounds are intended to be a (partial) counterpart to the bounds employed by the tangent plane axiom (standard). Unlike the tangent plane axiom, however, there does not exist a mathematical generalization like a "secant plane" that we could make use of. That is why the name of our axiom does not include the term "plane"[1]. Never the less, we will transfer the sentiment of secants to this case.

The general idea for the secant axiom (standard) is derived from the squared case. In that case, the secants that we constructed can be seen as a rotation of the tangent around the point (a, a^2), compare Figure 3.1. Similarly, we will rotate the tangent plane $z = bx + ay - ab$ around the point (a, b, ab). However, a plane requires not a point but an axis to be rotated around. We propose a rotation around the axis that is (i) parallel to the x-y-plane, (ii) lies in the tangent plane, and (iii) runs through (a, b, ab).

This can be realized by introducing a parameter $\alpha > 0$ that modifies the tangent plane so that $z = \alpha bx + \alpha ay - (2\alpha - 1)ab$. The idea is that we want to equally

[1] While our axiom does use planes for bounding the multiplication, we avoid the term "secant plane" since it is mathematically not well defined for this context.

scale the x- and y-component of the tangent plane. So initially, we came up with $z = \alpha bx + \alpha ay - \beta ab$. However, this plane does not generally intersect with the point (a, b, ab). To realize that, we merely have to insert the point into the formula. Doing this yields that $\beta = 2\alpha - 1$ must hold for our plane. From now on, we will refer to this plane as

$$\mathit{Secant}_{a,b,\alpha}(x,y) := \alpha bx + \alpha ay - (2\alpha - 1)ab.$$

Rotating the tangent plane in both directions by using an $\alpha > 1$ and an $\alpha < 1$ yields two different planes. Roughly speaking, the first creates a bound between the coordinate origin and the point (a, b, ab). This can always be achieved by setting $\alpha = \frac{1}{2}$. In Figure 3.2a, this is exemplary displayed for $a = 6$ and $b = 2$ with the corresponding tangent plane in gray for reference. The second plane bounds the multiplication from (a, b, ab) to the point $(\alpha a, \alpha b, \alpha^2 ab)$ using α as a heuristic parameter. For $\alpha = 1.2$, the plane is shown in Figure 3.2b for the same example as above.

Note that using only these two planes, the secant axiom will only be a counterpart for I_{ab} and III_{ab}, or II_{ab} and IV_{ab} of the tangent plane (see Figure 2.3 for reference). We are content with this for now, but extension of the axiom to the other two quadrants is discussed as future work in Section 5.1.1.

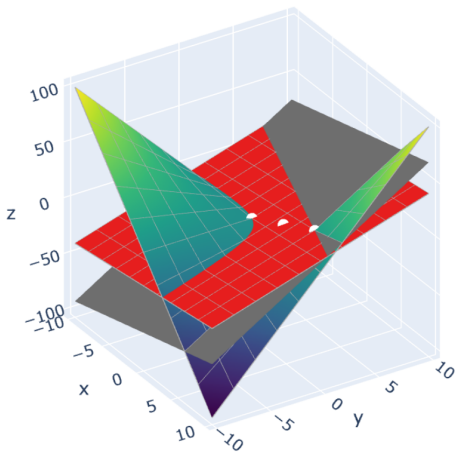
We now derive the secant axioms (standard) as displayed in Figure 3.1. First, we propose to separate cases I-IV, one for each quadrant of the projection of the three-dimensional space to x and y. So, for example, II is supposed to restrict the multiplication for given $a < 0$ and $b > 0$. This way, the appropriate axiom will bound z locally around the given point (a, b, ab).

The conclusions for each case are rather straight forward. Analogous to the tangent plane axiom, the aforementioned planes $\mathit{Secant}_{a,b,\frac{1}{2}}(x,y)$ and $\mathit{Secant}_{a,b,\alpha}(x,y)$ for $\alpha > 1$ are used to bound z. The premises then have to be fitted to formalize the domain over which the bound from the conclusion actually holds. This is more involved because, analogous to the squared case, our bounds are on the other side of the curvature of $z = x \cdot y$ compared to the tangent plane. So bounding the entire x-y-domain is not possible.

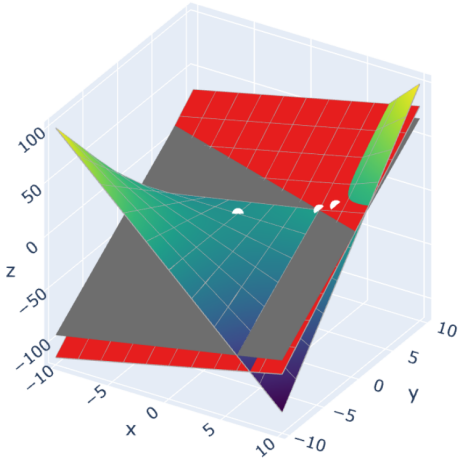
We derive the bounds as displayed in Figure 3.3 exemplary for I. For all other axioms, merely some inequalities have to be flipped. In order to derive the bounds, we consider the two proposed planes and look over which domain they can make guarantees. The first plane we consider is $\mathit{Secant}_{a,b,\alpha}(x,y)$ with $\alpha > 1$.

The intersection of $x \cdot y$ and this plane is $xy = \alpha bx + \alpha ay - (2\alpha - 1)$. This intersection can be seen very well for our example in Figure 3.2f. We can observe that this intersection is a hyperbola and therefore non-linear. Thus, we can not define it exactly in our axiom. Graphically speaking, we would like to formalize the red area in Figure 3.2f. We bound the parabola on the bottom left by $x \geq 6$ and $y \geq 2$ which generalizes to $x \geq a \wedge y \geq b$ for the axiom. The hyperbola on the top right can be under-approximated very well with the lines $x = 7.2$ and $y = 2.4$ since it tends towards these values in each direction. This generalizes to $x \leq \alpha a$ and $y \leq \alpha b$. Overall, we thus use $\mathit{Secant}_{a,b,\alpha}(x,y)$ as an upper-bound for $x \cdot y$ over the visualized L-shaped domain that can be formalized as $x \geq a \wedge y \geq b \wedge (x \leq \alpha a \vee y \leq \alpha b)$.

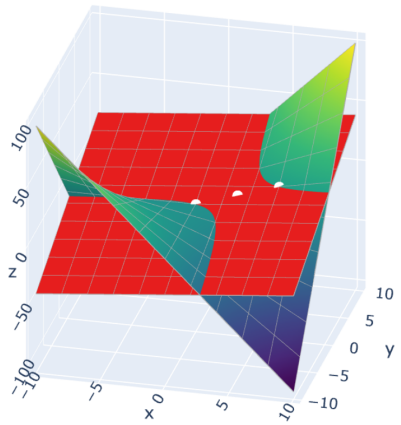
The second plane whose domain we need to derive is $\frac{1}{2}bx + \frac{1}{2}ay$, i.e. $\mathit{Secant}_{a,b,\frac{1}{2}}(x,y)$. We need to under-approximate the red area displayed in Figure 3.2d whose edge is again a hyperbola. Similar to the previous bound, we can safely approximate the L-shape that can be formalized as $x \geq 0 \wedge y \geq 0 \wedge (x \leq \frac{1}{2}a \vee y \leq \frac{1}{2}b)$.



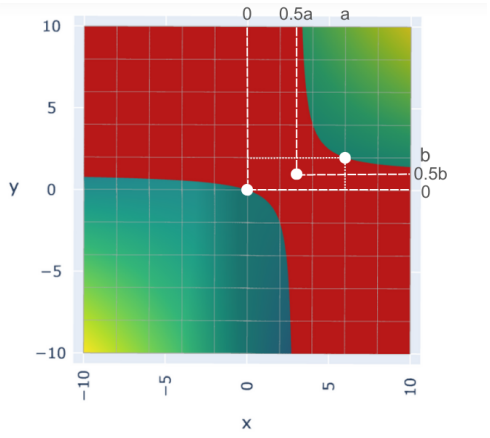
(a) $\alpha = 0.5$ with TP as reference



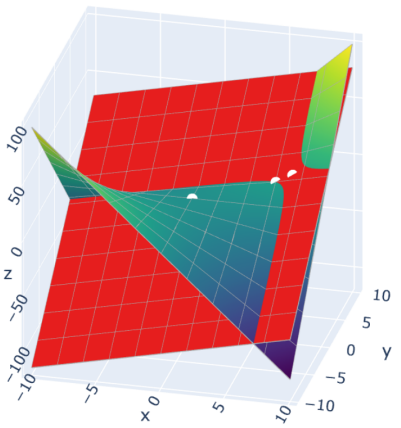
(b) $\alpha = 1.2$ with TP as reference



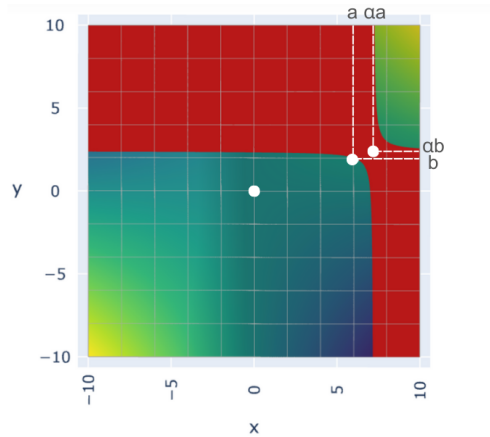
(c) $\alpha = 0.5$



(d) Projection to x- and y-axis



(e) $\alpha = 1.2$



(f) Projection to x- and y-axis

Figure 3.2: Plot of multiplication function $z = x \cdot y$ and rotated tangent planes $Secant_{6,2,\alpha}(x,y)$ for $\alpha = 0.5$ and $\alpha = 1.2$. Also, the origin, as well as the points (a, b, ab) and $(\alpha a, \alpha b, Secant_{6,2,\alpha}(\alpha a, \alpha b))$ are marked.

But only using this L-shaped bound, there would be a gap in the domain between $(\frac{1}{2}a,\ \frac{1}{2}b)$ and $(a,\ b)$. Therefore, we also include the box-shaped area $x \geq 0 \wedge y \geq 0 \wedge x \leq a \wedge y \leq b$ in the premise of the axiom. Combining the two formulas, we get the formalization of the domain $x \geq 0 \wedge y \geq 0 \wedge (x \leq \frac{1}{2}a \vee y \leq \frac{1}{2}b \vee (x \leq a \wedge y \leq b))$ over which $\mathit{Secant}_{a,b,\frac{1}{2}}(x,y)$ now constitutes an upper-bound for $x \cdot y$.

One last issue is the choice of α for $\mathit{Secant}_{a,b,\alpha}(x,y)$. Choosing it statically is not a good idea, since for larger values of a and b, the approximation becomes tighter. That is because, similar to the squared case, the multiplication becomes increasingly steeper over a fixed interval for growing absolute values of a and b.

As a solution, we introduce a "proxy parameter" d from which we can derive α, making its choice dynamic. Our proposal is to choose

$$\alpha = \frac{d}{\sqrt{a^2 + b^2}} + 1.$$

This will always yield an α that is *strictly* greater than one, thus guaranteeing a proper rotation of $\mathit{Secant}_{a,b,\alpha}(x,y)$. The idea behind this is to set d as the distance between $(a,\ b)$ and $(\alpha a,\ \alpha b)$ in the x-y-plane. Then, simply applying the Pythagorean theorem, we can derive α in dependency of d

$$\begin{aligned}
& & d^2 &= (\alpha a - a)^2 + (\alpha b - b)^2 \\
\Leftrightarrow & & d^2 &= a^2 \cdot (\alpha - 1)^2 + b^2 \cdot (\alpha - 1)^2 \\
\Leftrightarrow & & d^2 &= (\alpha - 1)^2 (a^2 + b^2) \\
\Leftrightarrow & & (\alpha - 1)^2 &= \frac{d^2}{a^2 + b^2} \\
\Rightarrow & & \alpha &= \frac{d^2}{\sqrt{a^2 + b^2}} + 1.
\end{aligned}$$

Different values for d are experimentally evaluated in Section 4.2.3. Analogous to the squared case, it might be useful to choose d dynamically as well. However, we leave this for future work.

As a minor note, we also explicitly propose and implemented a case 0 for the secant axiom that can be applied when $a = 0$ and $b = 0$, see Figure 3.3. It essentially bounds $x \cdot y$ over $[-d,\ d] \times [-d,\ d]$ using $\mathit{Secant}_{\pm d,\pm d,\frac{1}{2}}(x,y)$. But since it constitutes a special case, we will not further address it.

To conclude this section, we prove that the secant axiom is also correct in the standard case.

Theorem 3.2.3. *The secant axioms I-IV (standard case) as displayed in Figure 3.3 are correct w.r.t. Algorithm 1.*

Proof. We only provide a proof for I since II-IV can be proven analogously. Let there be a satisfying assignment for ψ. As stated in Lemma 3.2.1, it is to be shown that this assignment also satisfies I. Now, per Algorithm 1, it holds that

$$\psi \equiv \varphi \wedge \bigwedge_{\substack{\overline{z}=\overline{x}\cdot\overline{y} \text{ via} \\ \text{initial abstraction}}} \overline{z} = \overline{x} \cdot \overline{y}.$$

Thus, the assignment satisfies $z = x \cdot y$, where x, y and z are the variables contained in I. Using this equality and extracting some boolean structure from I, we can reformulate what remains to be shown. Let $a > 0$, $b > 0$ and $\alpha > 1$,

1. if $a \leq x \leq \alpha a$ and $b \leq y$, or $b \leq y \leq \alpha b$ and $a \leq x$ then $xy \leq \mathit{Secant}_{a,b,\alpha}(x,y)$

2. if $x \geq 0$ and $y \geq 0$ and $x \leq \frac{1}{2}a \vee y \leq \frac{1}{2}b \vee (x \leq a \wedge y \leq b)$ then $xy \leq \mathit{Secant}_{a,b,\frac{1}{2}}(x,y)$.

1. Let $a \leq x \leq \alpha a$ and $y \geq b$, or $b \leq y \leq \alpha b$ and $x \geq a$, as well as $\alpha > 1$. Looking merely at the lower bounds of the given domain, let $x = a$. Then

$$\begin{aligned} & ay \leq \alpha ba + \alpha ay - (2\alpha - 1) \cdot ab \\ \overset{a \geq 0}{\Leftrightarrow} \quad & y \leq \alpha b + \alpha y - 2\alpha b + b \\ \Leftrightarrow \quad & y \leq \alpha y - \alpha b + b \\ \Leftrightarrow \quad & y - b \leq \alpha(y - b) \\ \overset{y \geq b}{\Leftrightarrow} \quad & \alpha \geq 1 \end{aligned}$$

which holds per assumption. For $y = b$, the inequality holds analogously, meaning that overall $xy \leq \mathit{Secant}_{a,b,\alpha}(x,y)$ at the lower bounds $x = a \wedge y \geq b$ and $y = b \wedge x \geq a$.

Now, the gradients of the functions xy and $\mathit{Secant}_{a,b,\alpha}(x,y)$ are

$$\nabla xy = \begin{bmatrix} \frac{\partial xy}{\partial x} \\ \frac{\partial xy}{\partial y} \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix} \qquad \text{and}$$

$$\nabla \mathit{Secant}_{a,b,\alpha}(x,y) = \begin{bmatrix} \frac{\partial \alpha ba + \alpha ay - (2\alpha-1)\cdot ab}{\partial x} \\ \frac{\partial \alpha ba + \alpha ay - (2\alpha-1)\cdot ab}{\partial y} \end{bmatrix} = \begin{bmatrix} \alpha b \\ \alpha a \end{bmatrix}.$$

This means that from any point (a, y), where $y \geq b$, onward in positive direction of y, the growth of $\mathit{Secant}_{a,b,\alpha}(x,y)$ is greater than the growth of xy up to at least $x = \alpha a$. Therefore, $xy \leq \mathit{Secant}_{a,b,\alpha}(x,y)$ must hold for $a \leq x \leq \alpha a$. Arguing analogously with (x, b), where $x \geq b$, proves that the inequality must also hold for $b \leq y \leq \alpha b$.

2. Let $x \geq 0$ and $y \geq 0$ and $x \leq \frac{1}{2}a \vee y \leq \frac{1}{2}b \vee (x \leq a \wedge y \leq b)$. The inequality trivially holds for $x = 0$ or $y = 0$, so assume $x > 0$ and $y > 0$. Rewriting yields

$$\begin{aligned} & xy \leq \frac{1}{2}bx + \frac{1}{2}ay \\ \Leftrightarrow \quad & 1 \leq \frac{b}{2y} + \frac{a}{2x}. \end{aligned}$$

Since $x > 0$ and $y > 0$, we only need to test the upper values of the assumed domain for x and y, since these minimize the right-hand side of the inequality. For $x = \frac{1}{2}a$, it follows that

$$\begin{aligned} & \frac{1}{2}ay \leq \frac{1}{2}b\frac{1}{2}a + \frac{1}{2}ay \\ \Leftrightarrow \quad & ay \leq \frac{1}{2}ab + ay \\ \Leftrightarrow \quad & ab \geq 0 \end{aligned}$$

which holds per assumption. For $y = \frac{1}{2}b$, the inequality holds analogously. For $x = a$,

it follows that

$$\begin{aligned} & ay \leq \frac{1}{2}ab + \frac{1}{2}ay \\ \Leftrightarrow \quad & y \leq \frac{1}{2}b + \frac{1}{2}y \\ \Leftrightarrow \quad & y \leq b \end{aligned}$$

which holds per assumption. For $y = b$, the inequality holds analogously. □

Zero axiom (squared): $x = 0 \leftrightarrow z = 0$

Zero axiom (standard): $(x = 0 \vee y = 0) \leftrightarrow z = 0$

Sign axiom (squared): $z \geq 0$

Sign axioms (standard):
$((x \geq 0 \wedge y \geq 0) \vee (x \leq 0 \vee y \leq 0)) \leftrightarrow z \geq 0$
$((x \geq 0 \wedge y \leq 0) \vee (x \leq 0 \vee y \geq 0)) \leftrightarrow z \leq 0$

Tangent plane axiom (squared):
$(x = a \rightarrow z = ax) \ \wedge z \geq 2ax - a^2$

Tangent plane axiom (standard):
$(x = a \rightarrow z = ay) \ \wedge \ (y = b \rightarrow z = bx) \wedge$
$(((x \geq a \wedge y \leq b) \vee (x \leq a \wedge y \geq b)) \leftrightarrow z \leq bx + ay - ab) \wedge$
$(((x \leq a \wedge y \leq b) \vee (x \geq a \wedge y \geq b)) \leftrightarrow z \geq bx + ay - ab)$

Secant axiom (squared) w.r.t. their quadrant:
I: $(0 \leq x \leq a \rightarrow z \leq ax) \ \wedge \ (a \leq x \leq a + d \rightarrow z \leq (2a + d)x + (-a^2 - da))$
II: $(0 \geq x \geq a \rightarrow z \leq ax) \ \wedge \ (a \geq x \geq a - d \rightarrow z \leq (2a - d)x + (-a^2 + da))$
0: $(-d \leq x \leq 0 \rightarrow z \leq -dx) \ \wedge \ (0 \leq x \leq d \rightarrow z \leq dx)$

Secant axioms (standard) w.r.t. their quadrant:
I: $[(x \geq a \wedge y \geq b \wedge (x \leq \alpha a \vee y \leq \alpha b)) \rightarrow z \leq \mathit{Secant}_{a,b,\alpha}(x,y)] \wedge$
$[(x \geq 0 \wedge y \geq 0 \wedge (x \leq \frac{1}{2}a \vee y \leq \frac{1}{2}b \vee (x \leq a \wedge y \leq b))) \rightarrow z \leq \mathit{Secant}_{a,b,\frac{1}{2}}(x,y)]$
II: $[(x \leq a \wedge y \geq b \wedge (x \geq \alpha a \vee y \leq \alpha b)) \rightarrow z \geq \mathit{Secant}_{a,b,\alpha}(x,y)] \wedge$
$[(x \leq 0 \wedge y \geq 0 \wedge (x \geq \frac{1}{2}a \vee y \leq \frac{1}{2}b \vee (x \geq a \wedge y \leq b))) \rightarrow z \geq \mathit{Secant}_{a,b,\frac{1}{2}}(x,y)]$
III: $[(x \leq a \wedge y \leq b \wedge (x \geq \alpha a \vee y \geq \alpha b)) \rightarrow z \leq \mathit{Secant}_{a,b,\alpha}(x,y)] \wedge$
$[(x \leq 0 \wedge y \leq 0 \wedge (x \geq \frac{1}{2}a \vee y \geq \frac{1}{2}b \vee (x \geq a \wedge y \geq b))) \rightarrow z \leq \mathit{Secant}_{a,b,\frac{1}{2}}(x,y)]$
IV: $[(x \geq a \wedge y \leq b \wedge (x \leq \alpha a \vee y \geq \alpha b)) \rightarrow z \geq \mathit{Secant}_{a,b,\alpha}(x,y)] \wedge$
$[(x \geq 0 \wedge y \leq 0 \wedge (x \leq \frac{1}{2}a \vee y \geq \frac{1}{2}b \vee (x \leq a \wedge y \geq b))) \rightarrow z \geq \mathit{Secant}_{a,b,\frac{1}{2}}(x,y)]$
0: $[(x \geq 0 \wedge y \geq 0 \wedge (x \leq d \vee y \leq d)) \rightarrow z \leq dx + dy] \wedge$
$[(x \leq 0 \wedge y \leq 0 \wedge (x \geq d \vee y \geq d)) \rightarrow z \leq -dx - dy] \wedge$
$[(x \leq 0 \wedge y \geq 0 \wedge (x \geq d \vee y \leq d)) \rightarrow z \geq dx - dy] \wedge$
$[(x \geq 0 \wedge y \leq 0 \wedge (x \leq d \vee y \geq d)) \rightarrow z \geq -dx + dy] \wedge$

Monotonicity axiom:
$(|x_1| \leq |x_2| \wedge |y_1| \leq |y_2|) \rightarrow |z_1| \leq |z_2|$

Congruence axiom:
$(x_1 = x_2 \wedge y_1 = y_2) \rightarrow z_1 = z_2$

Figure 3.3: All implemented axioms w.r.t. summarized multiplication $z = x \cdot y$ (standard), $z = x \cdot x$ (squared), or summarized multiplications $z_1 = x_1 \cdot y_1$ and $z_2 = x_2 \cdot y_2$. Thereby $a = \mu(x)$, $b = \mu(y)$ and $c = \mu(z)$. Also, $\alpha > 1$ and $d > 0$ are constants.

Chapter 4

Evaluation

First, we will give details on the implementation and setup of our solver. Then, its performance will be analyzed with a special focus on the impact of heuristic parameters.

4.1 Implementation Details and Setup

We implemented incremental linearization [Irf18] with the addition of all the modifications that we proposed in the previous Chapter 3. It is implemented as a module in the open source C++ toolbox for strategic and parallel SMT-solving SMT-RAT [smt][CKJ+15]. In the following, we will consider this module as a standalone NRA solver, simply referring to it by IL.

In the module, heuristic parameters have to be specified that influence the run of the module. Most of these have been mentioned over the past two chapters. We will now name and explain the parameters, as well as the specific values they can adopt.

The first parameter is the *initial abstraction strategy* (IAS). It influences the execution of INITIAL-ABSTRACTION (Section 2.2.1) by changing the criterion by which the next multiplication to be replaced is chosen. The possible values for IAS are

Optimal Implements the *optimal w.r.t. number of fresh variables heuristic* as described in Section 3.1.1.

Exponents first Implements the *exponents first* heuristic, see Section 3.1.2.

First Always abstract the first product of two variables that occurs in the formula.

The second parameter is the *axiom selection strategy* (AS). This parameter determines how many instances of axioms are returned from the REFINE method. We implemented this method so that the different axiom types are considered serially. The order imposed on the axioms is

$$\text{zero} \rightarrow \text{sign}(\rightarrow \text{congruence} \rightarrow \text{monotonicity}) \rightarrow \text{tangent plane} \rightarrow \text{secant}.$$

When an axiom type is considered, the axiom is instantiated for all pairs of variables (or pairs of pairs of variables), for each instance checking whether it is satisfied under the current assignment or not. Remember that only the instances that are not satisfied are returned by REFINE. Now, we implemented three different strategies

Full lazy Only the first instance of an axiom generated in the above-mentioned order that is not satisfied under the current assignment, is returned.

Less lazy Only all zero and sign axioms are instantiated to potentially be returned. If all these instances are satisfied, the above order is continued *full lazy*.

Full eager Return all (unsatisfied) instances of all axiom types.

The third implemented parameter is the *secant distance* (SD). This is the d that is directly used in the squared secant axiom and implicitly used in the standard secant axiom to derive α. In general, it can take any value greater than zero. The values we will test are 1, 10, 100 and 1000.

The last parameter is simply created to toggle the use of the specific axiom on or off. This parameter is used when we test the module with versus without (a) the secant axiom and (b) the congrunence and monotonicity axiom.

To test our implementation, we used the set of benchmarks contained in the SMT-LIB-BENCHMARKS QF_NRA library [QFN][BST+10]. The set consists of NRA formulas, which we will refer to as *problems*. On the hardware side, each problem gets 5 minutes of runtime with 6GB of RAM available on a 2.1GHz thread of an Intel Xeon Platinum 8160.

4.2 Experimental Evaluation

We will proceed with the evaluation as follows. First, we will show general statistics of IL and compare it to a state-of-the art solver for NRA. Then, we will analyze the impact on performance for each of the heuristic parameters of the module. We begin with the general statistics.

4.2.1 General Statistics and Comparison to MCSAT

For a first impression, we want to look at the performance of the best-performing configuration of IL. This configuration uses *first* as the initial abstraction strategy, the axiom selection strategy *full eager*, and a secant distance of 1. This configuration will be experimentally determined in the following sections.

As a reference, we use the best-performing strategy of SMT-RAT for solving NRA. We call this solver MCSAT since it implements the MCSAT Framework [JdM12][dMJ13] but uses multiple different approaches for creating explanations in series without passing on information. The first approach implements *Fourier Motzkin variable elimination* [JBDM13] and is thereby only applicable for linear constraints. If the input is not linear, *Interval Constraint Propagation* [Kre19] and then *Virtual Substitution* [ÁNK17] are used. These are both incomplete, so a result is still not guaranteed. If neither of these is successful, the final, complete approach, which uses single cell construction[Spe20][NÁS+23], is used.

To get an initial impression in regard to runtime, we create performance profiles. That is, we plot runtime against the number of problems solved in that time for each solver. Additionally, the virtual best is plotted. That being an imaginary solver that for each problem has the smaller runtime between IL and MCSAT.

The profiles are displayed Figure 4.1. We can see that IL overall performs significantly worse than MCSAT. That, however, was to be expected, seeing that this solver on its own does not even have the capabilities to solve all problems given enough

time due to the approaches incompleteness. Thus, MCSAT should rather be seen as a ceiling for performance. Under this interpretation, IL performs very well, solving up to roughly 6500 problems in the given 5 minutes. Additionally, the virtual best not being on par with the curve of MCSAT means that for any runtime within the timeout, there are problems that IL solves faster than MCSAT. As a minor detail, IL performs slightly better than MCSAT for about the first 150 ms. The difference is rather small though, and most likely due to some technicality like MCSAT having a comparably large overhead for problems with low complexity.

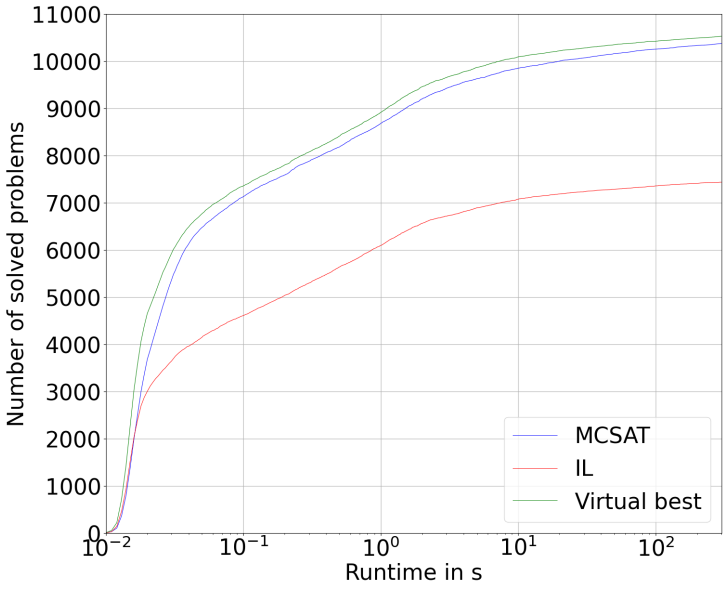


Figure 4.1: Performance profile of MCSAT and IL in its best performing configuration, i.e. IAS *first*, AS *full eager* and SD 1.

To slightly extend our impression, we list the number of solved problems and respective mean runtimes of the overall, SAT (satisfiable), and UNSAT (unsatisfiable) problems for both solvers. This can be seen in Table 4.1. Note that the runtimes in the table cannot be directly compared since they are not calculated on the same set of problems. Nevertheless, it can be seen that IL solves more UNSAT than SAT problems while having a similar mean runtime for both. On the other hand, MCSAT solves almost equally many problems of both kinds, but in contrast to IL has a far higher mean runtime on UNSAT problems compared to SAT ones.

	IL	MCSAT
Number of solved problems	7434	10371
Mean runtime	3.65	3.86
Number of solved SAT problems	3166	5181
Mean runtime	3.77	2.62
Number of solved UNSAT problems	4268	5190
Mean runtime	3.56	5.09

Table 4.1: General statistics for IL and MCSAT

In order to further investigate this, we create a scatter plot of all individual problems with respect to their kind, i.e., SAT or UNSAT. It is shown in Figure 4.2. To see the difference in performance, the plot should be considered with a diagonal split. The upper left triangle contains the problems that MCSAT performs better on. Similarly, the problems in the bottom right triangle are the ones that IL performs better on.

While the same holds for MCSAT, it can be seen that there are quite a few problems that IL performs better on than MCSAT. On some of these, performance is even so much better that MCSAT did not solve them within the timeout. These problems can be seen above the 300-second mark on the axis for MCSAT. This shows that IL does not just solve a subset of the problems that MCSAT solves but instead seems to have use cases where it performs better. This motivates its usage of some kind in a complete NRA solver, as will be discussed in Section 5.1.2.

The plot also once again shows that IL performs well on UNSAT problems. This might be due to multiple factors. For one, the approach likely works better for UNSAT problems since solving SAT problems relies on the chance that a satisfying assignment for the linear abstraction is also satisfying for the non-linear input. Whereas UNSAT problems are identified as such when the linear abstraction becomes unsatisfiable, which is unambiguous as soon as it happens. Additionally, MCSAT generally performs worse on UNSAT problems since the algorithm essentially constitutes a search for a satisfying assignment. Thus, unsatisfiability can only be concluded when the entire search space has been covered.

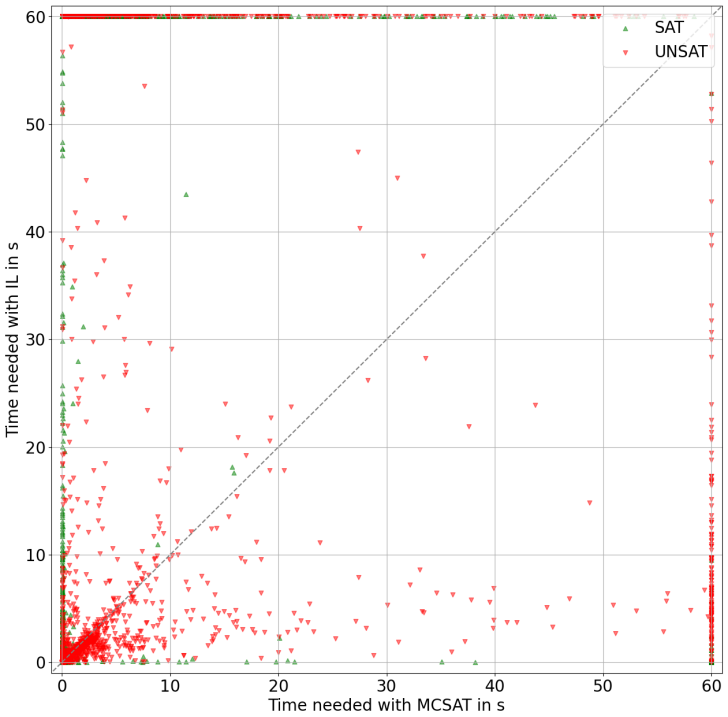


Figure 4.2: Scatter plot for MCSAT and IL in its best performing configuration, i.e. IAS *first*, AS *full eager* and SD 1, plotting all solved problems against each other based on runtime.

Lastly, we will look at some statistics specific to IL. These are displayed in Table 4.2. First of all, we see that the mode is always the smallest number it can be. So at least half of the solved problems are low in complexity with respect to IL.

The mean value is generally more consistent with the third quartile for each statistic. Only the added number of tangent plane axioms constitutes an exception. There, we can see that the mean is inflated by at most 25% of the solved problems in the set. Even considering this inflation, though, the mean and third quartile are still very high compared to the other statistics. This indicates that the tangent plane axiom is of high relevance for the success of IL. A reason for that could be the adaptation of making implications in the tangent plane axiom equivalences, see Section 3.2.1. These equivalences are also a major difference in contrast to the secant axiom. We therefore propose similar changes to the secant axiom in Section 5.1.1.

Seeing that 75% of the solved problems are solved within at most 3 iterations indicates that incremental linearization has the potential to be used with a limit of iterations, e.g., for pre-processing before using a complete solver. We further discuss this in Section 5.1.2.

	Mean # of	Mode of	Third quartile
Added zero axioms	3.35	0	3
Added sign axioms	3.31	0	2
Added tangent plane axioms	27.86	0	12
Added secant axioms	3.35	0	3
Iterations	2.57	1	3

Table 4.2: Specific statistics for IL, only considering problems that have been solved within the 5 minute timeout.

The rest of this chapter will investigate the impact of the aforementioned heuristic parameters on the performance of IL in the given test setting.

4.2.2 Impact of the Secant Axiom

First, we would like to determine whether the secant axiom is useful in practice. For that, we created a performance profile of IL in two configurations. One utilizes the secant axiom, and the other does not. Both use *first* as the IAS since this is the "most random" strategy. That is, a specific target is not optimized, which the secant axiom could potentially make use of. They both also use the *full eager* AS, since this leads to the most possible usage of the secant axiom. Also, the secant distance is set to 1.

The performance profiles are displayed in Figure 4.3 and the result is very clear. With access to the secant axiom, IL performs significantly better across the entire runtime. This justifies the usage of the secant axiom in all following investigations and in the potentially best configuration of the module (for this set of representative benchmarks).

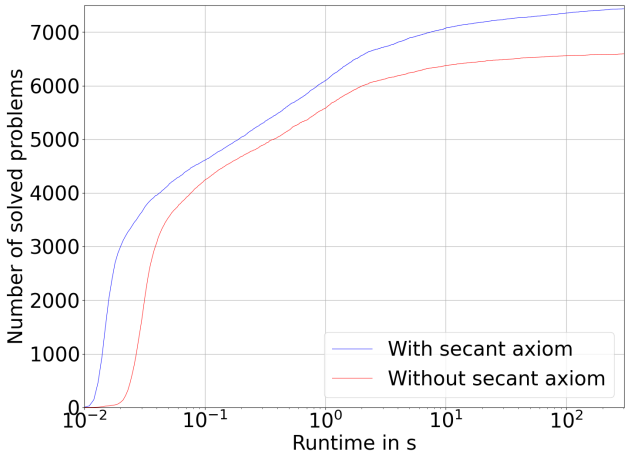


Figure 4.3: Performance profiles for IL with IAS *first*, AS *full eager* and SD 1 with varying usage of the secant axiom for refinement.

4.2.3 Impact of the Secant Distance

The next parameter that we analyze is the secant distance. For that, we once again create a performance profile. This time for four different configurations of the module, so that the SDs 1, 10, 100 and 1000 can be tested. For all, we use *exponents first* as the IAS to potentially have a few more square cases occur, and again, *full eager* as the AS to have the computation of secant axioms occur as much as possible.

In Figure 4.4, the performance profiles are shown. Here, the results are not as clear. The profiles are all very close to each other, so the parameter does not seem to have a big impact on performance. However, we can see that up to the runtime mark of about 3 ms, the SD of 1000 seems to perform the best. But onward from that point in time, it is overtaken by SD 1 which then is slightly better than all other distances over the entire remaining runtime. Therefore, we fix the SD at 1 for IL.

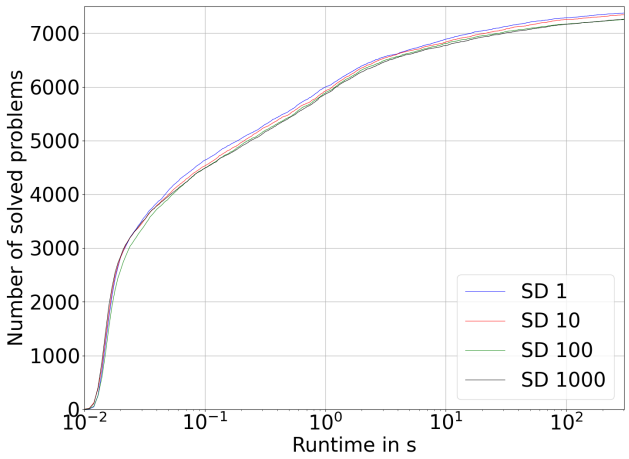


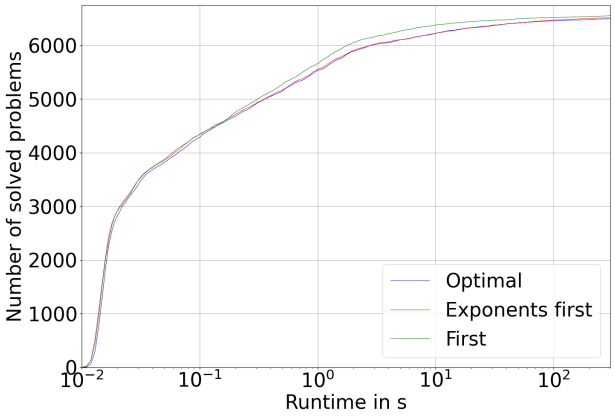
Figure 4.4: Performance profiles for IL with IAS *exponents first*, AS *full eager* and secant distances varying between 1, 10, 100 and 1000.

4.2.4 Impact of the Initial Abstraction Strategy

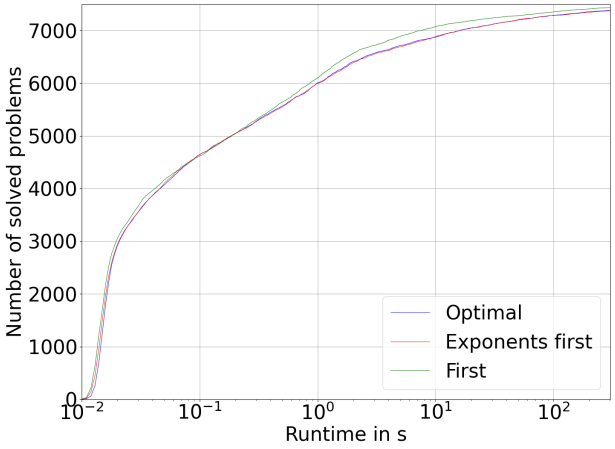
Now, we would like to consider the impact of the initial abstraction strategy on the runtime of the module. For that, we create performance profiles again. The *first* strategy might profit from quick refinement, which is provided by the *full lazy* AS. On the other side, IASs *exponents first* as well as *optimal* optimize variables and therefore might perform better when more axioms are computed, that is, *full eager* is used as the AS. Therefore, we make two comparisons. One with *full eager* and one with *full lazy* as the AS. For SD, 1 is used as determined previously.

Figure 4.5 shows the performance profiles. Contrary to our proposition, *exponents first* and *optimal* do not perform better using the *full eager* AS when compared to *first*. It looks like *first* is even a slight bit further off the other curves there. In general, however, *first* seems to be the best performing strategy.

Now, *exponents first* and *optimal* have similar overhead and are relatively on par in the profiles. On the other hand, *first* has a far lower overhead, and its profile has a relatively small but constant offset to the other two curves. Thus, the most likely reason for the difference in performance is the additional overhead.



(a) AS *full lazy*



(b) AS *full eager*

Figure 4.5: Performance profiles for IL with SD 1 and varying IAS for different ASs.

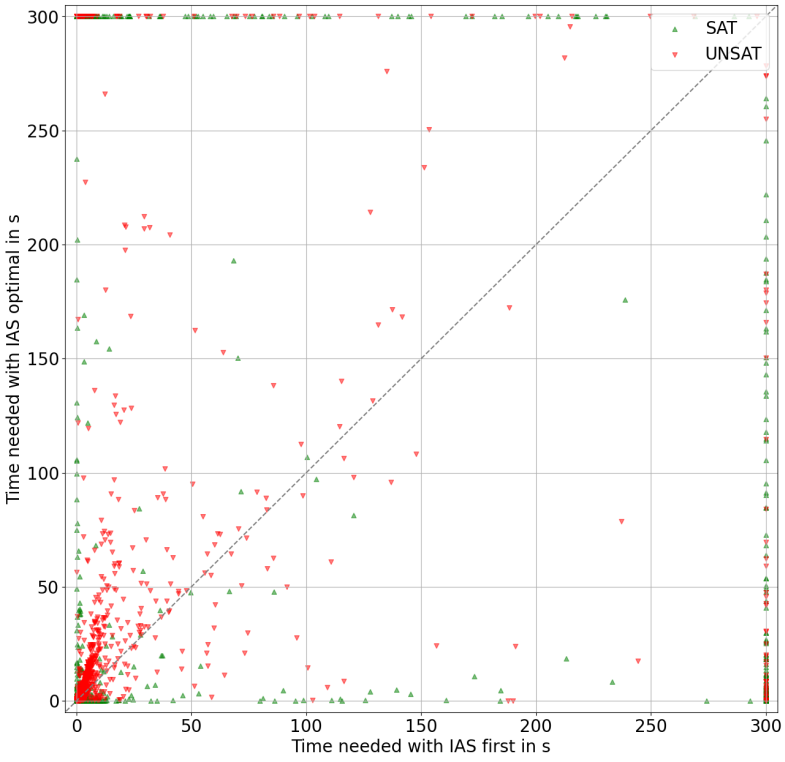
To further analyze this, we look at the mean number of fresh variables created for the initial abstraction. In Table 4.3, we provide this statistic with the further distinction of variables that abstract squares of variables. And we can see that our two heuristics indeed do what they are supposed to. That is because *optimal* has the lowest number of new variables, and *exponents first* introduces the most squared variables.

	Mean # of new variables	Mean # of squared new variables
Optimal	30.56	7.42
Exponents first	31.78	8.46
First	31.61	8.01

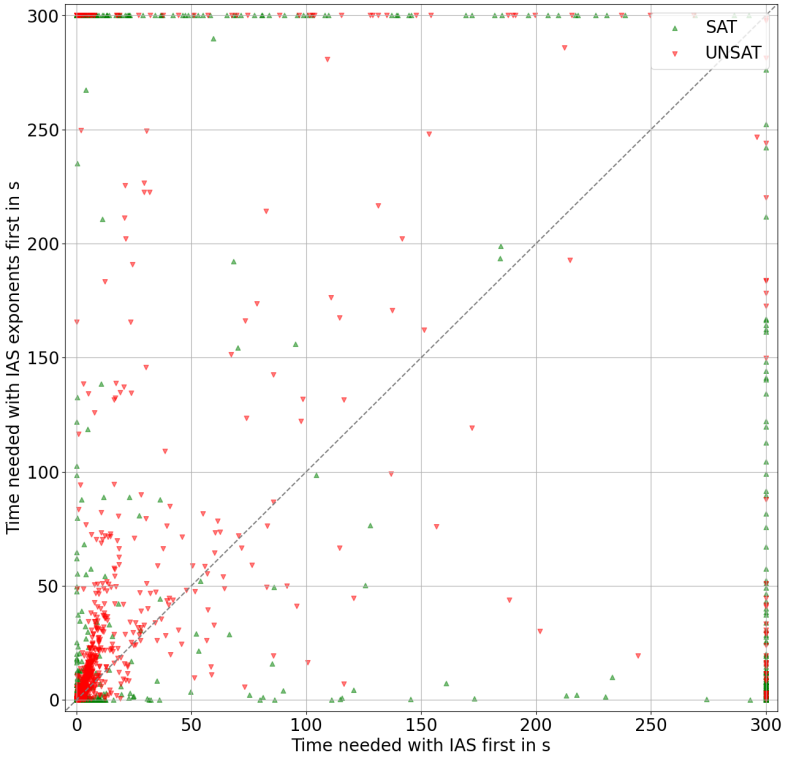
Table 4.3: Mean of the number of newly introduced variables for each IAS.

So it seems, the optimizations done by the *optimal* and *exponents first* heuristics are not substantial enough to justify the additional effort. Nevertheless, for specific problems, especially bigger ones where the additional overhead does not matter so much, these two strategies might still be well suited. To underline this, we created scatter plots as before in Figure 4.6, however, this time, comparing IL with itself using different IASs. The plots both show similar results. First of all, the IAS *first* performs better on easy inputs, as can be seen from the clustering in the lower left that is mostly above the diagonal. But second of all, there are also a few problems that *optimal* and *exponents first* solve faster than *first*. There are even quite a few problems that are solved that *first* does not solve at all.

So generally, *first* does perform best overall. However, there are problems where *optimal* and *exponents first* do perform better. These are most likely ones that are high in complexity. From now on, though, we will use *first* as the IAS for testing.



(a) *first* vs *optimal*



(b) *first* vs *exponents first*

Figure 4.6: Scatter plot for IL using AS *full eager* and SD 1, plotting all solved problems against each other based on runtime for different IASs.

4.2.5 Impact of the Axiom Selection Strategy

Next, we want to answer the question that has already been posed in [Irf18]: Which axiom selection strategy performs the best? In order to answer it, we again compute performance profiles. For that, we use the configuration of our module that performs the best. That being the IAS *first* and a secant distance of 1.

The performance profiles that are displayed in Figure 4.7 show that the *full eager* AS generally performs the best. From about 0.3 s on, the strategy even performs significantly better than the other two. Only for the brief window from roughly 30 ms to just below 200 ms, the *less lazy* AS outperforms *full eager* very slightly. However, before and after that window, this strategy performs significantly worse. Therefore, we consider *full eager* as the AS for our best module configuration.

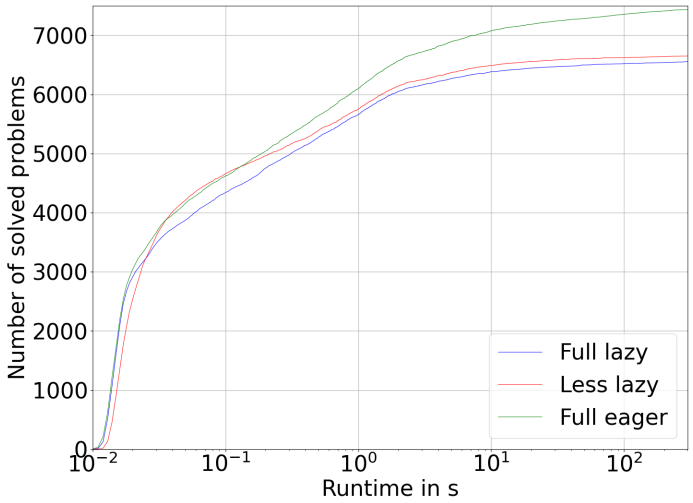


Figure 4.7: Performance profiles for IL with IAS *first* and SD 1 with varying ASs.

4.2.6 Impact of the Monotonicity and Congruence Axiom

In Section 3.2.1, we argued that the monotonicity and the congruence axiom in general are not helpful for refinement. We indicate the correctness of this assessment by testing IL in different configurations that only differ by the inclusion of the two axioms for refinement. We therefore created performance profiles that otherwise use the (for these benchmarks) optimal configuration of IAS *first*, AS *full eager*, and SD 1.

And indeed, our claims are confirmed by the profiles displayed in Figure 4.8. The best configuration is the one that uses neither of the axioms, and the worst is the one that uses both. Individually, it seems that monotonicity axiom decreases performance more than the congruence axiom.

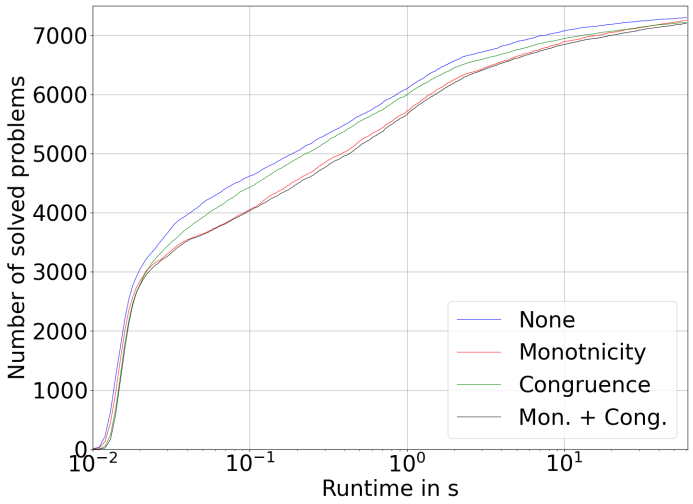


Figure 4.8: Performance profiles for IL with IAS *first*, AS *full eager* and SD 1 with varying inclusion of monotonicity and congruence axiom.

Chapter 5

Conclusion

5.1 Future Work

5.1.1 Improving the Secant Axiom

As far as we are aware, this thesis is the first to introduce a viable (partial) counterpart for the tangent plane axiom. Therefore, there is very likely room for improvement. In the following, we make some proposals that could potentially improve the axiom.

Extending the Axiom

The secant axioms as presented here only give a (partial) counterpart to either I_{ab} and III_{ab}, or II_{ab} and IV_{ab}. To be more accurate, it would be useful to also bound the other two quadrants.

Without guaranteeing correctness, we propose an extension of the secant axiom that also covers the remaining two quadrants, exemplary presenting I:

$$\begin{aligned}
&[(x \geq a \wedge y \geq b \wedge (x \leq \alpha a \vee y \leq \alpha b)) \rightarrow z \leq \alpha b x + \alpha a y - (2\alpha - 1)ab] \wedge \\
&[(x \geq 0 \wedge y \geq 0 \wedge x \leq a \wedge y \leq b) \rightarrow z \leq \frac{1}{2} b x + \frac{1}{2} a y] \wedge \\
&[(x \geq a \wedge y \geq 0) \rightarrow z \geq a y] \wedge \\
&[(x \geq 0 \wedge y \geq b) \rightarrow z \geq b x]
\end{aligned}$$

Note that we also changed the premise of the second implication. This is done to restrict the formalized x-y-domain so that it subsets III_{ab}, simultaneously reducing the complexity of the formula.

Backward Direction of the Axiom

In Section 3.2.1, we proposed to turn the key implications contained in the original tangent plane axiom from [Irf18] into equivalences. While this is a very minor syntactical change, the impact is quite big, essentially adding a whole new axiom. Having such a backward direction for the implications in the secant axiom might further elevate its usefulness.

However, simply turning the current implications contained in the secant axiom into equivalences would make the axiom incorrect. That is because the premises

under-approximate the domain over which the conclusion holds. Therefore, essentially an entire new axiom has to be designed.

5.1.2 Using IL for Complete NRA Solving

We have seen that IL as a standalone NRA solver cannot compete with complete, state-of-the art NRA solvers, see Section 4.2.1. Therefore, it makes sense to explore how IL can be used as part of a complete solver.

Similar to how different, partly incomplete explanations are used in series in MCSAT, IL could be used in series, before a complete NRA solver. This would, of course, implicitly already solve all linear problems. But with a possibly minor increase in runtime, a lot of non-linear problems could be solved as well before resorting to a complete NRA solver.

To substantiate this idea, we create a bar plot, that for each number of iterations IL could do, shows the number of problems that were solved using that exact number of iterations, see Figure 5.1. Note that the iterations axis has been trimmed to 18. The real maximum value is 35. Nevertheless, within the omitted range, there are only 40 problems overall, with a maximum number of 7 problems for a single iteration.

The plot shows that most problems are solved within very few iterations, with by far the most being solved in a single iteration. This indicates that setting a limit to the number of iterations could be a good way to get good results within a reasonable time frame. The mean solving times that are annotated for each bar support this claim, seeing that they are very low.

However, we also looked at the maximum solving time for each iteration. And for example, for one iteration this time is 276.8 s which is very high since the timeout was set at 300 s. So the additional use of a timeout might be advised to solve exactly the many problems with low runtimes indicated by the mean.

So overall, running IL for 1-5 iterations with a timeout of about at most 1m before resorting to a complete NRA solver could constitute a complete solver that has improved performance. Of course, other, possibly more interconnected, approaches are conceivable in order to make use of IL for complete NRA-solving.

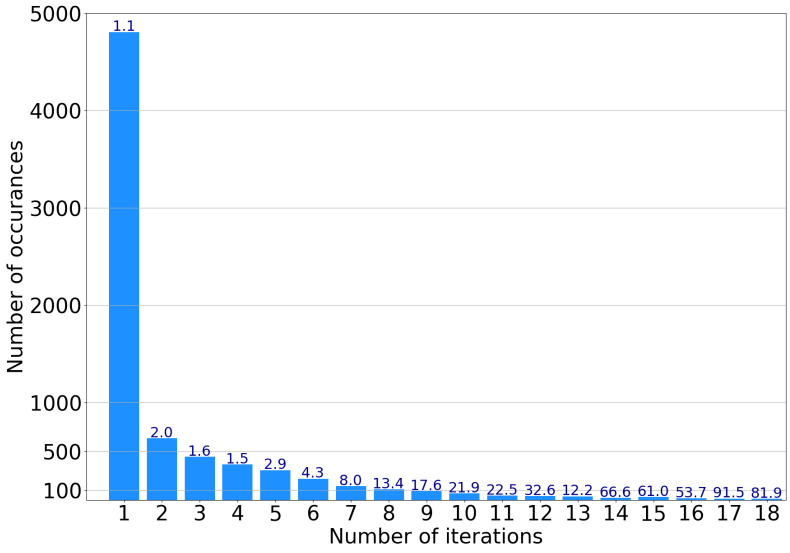


Figure 5.1: Bar plot showing the number of solved problems using IL with 5 m timeout for each number of iteration the algorithm did. Also, the mean solving time in seconds is annotated for each number of iterations.

5.2 Summary

In this thesis, we proposed improvements to the existing method of incremental linearization (IL) for *incomplete* satisfiability modulo non-linear real arithmetic (NRA) checking. We first introduced IL as presented in [Irf18] and [CGI+18].

To summarize, this method initially abstracts the non-linear input formula ψ as a linear formula φ, also storing the background information as to how this transformation was done and thereby could be reversed. Said background information can be viewed as non-linear constraints of the form $z = x \cdot y$ for variables x, y and z.

After the initial abstraction, a loop is entered. In this loop, it is first checked whether φ is satisfiable using a LRA solver. The result of this check can possibly be transferred to ψ. If not, φ will be refined by conjuncting formulas that linearly axiomatize the previously abstracted, non-linear multiplications $z = x \cdot y$ to it. We categorize different types of such formulas under the term *axiom*.

The first change to IL we presented was different heuristics for the initial abstraction. We named these *optimal w.r.t. number of fresh variables* and *exponents first*. Thereby, the first aims at minimizing the background information, i.e., the number of constraints $z = x \cdot y$. The latter tries to maximize the number of constraints that are squares ($z = x \cdot x$).

The second change deals with the axioms that are used for refinement. Next to some modifications of existing axioms, the most prominent change is the addition of a new type of axiom, the *secant axiom*. It is supposed to act as a (partial) counterpart to the already existing tangent plane axiom. For example, in the case of a squared multiplication, $z = x \cdot x$, the tangent plane axiom provides a lower-bound for the parabola $x \cdot x$ in the form of a tangent. The secant axiom then provides an upper bound in the form of multiple secants.

We implemented IL, including all changes in SMT-RAT [smt] as a standalone NRA solver and tested it on the SMT-LIB-BENCHMARKS QF_NRA library [QFN][BST+10]. The evaluation of these tests mainly focused on the optimization of the heuristic parameters of the module.

One of these is the choice of heuristic for initial abstraction. The evaluation showed that both proposed heuristics successfully do what they were intended for. However, they are generally outperformed by a more primitive, greedy approach to abstraction. That is most likely due to lower overhead. Nevertheless, the proposed heuristics could have specific use for complex benchmarks.

To evaluate our other major contribution, we checked whether the secant axiom improves performance or not. And here, the results are clear. Usage of the secant axiom leads to a significant, overall decrease in runtime.

Lastly, we proposed future work motivated by this thesis. For one, we see room for improvement with regard to the secant axiom. Secondly, we presented an idea for making use of incremental linearization for complete NRA-solving. This is of high relevance since IL as a standalone solver does not provide competitive performance compared to state-of-the art solvers.

Bibliography

[ÁNK17] Erika Ábrahám, Jasper Nalbach, and Gereon Kremer. Embedding the virtual substitution method in the model constructing satisfiability calculus framework. In *CEUR Workshop Proceedings*, volume 1974. RWTH Aachen, 2017.

[BdM14] Nikolaj Bjørner and Leonardo de Moura. Applications of smt solvers to program verification. *Notes for the Summer School on Formal Techniques*, 2014.

[BST+10] Clark Barrett, Aaron Stump, Cesare Tinelli, et al. The SMT-LIB standard: Version 2.0. In *Proceedings of the 8th International Workshop on Satisfiability Modulo Theories (Edinburgh, England)*, volume 13, page 14, 2010.

[CGI+18] Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani. Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions. *ACM Transactions on Computational Logic (TOCL)*, 19(3):1–52, 2018.

[CKJ+15] Florian Corzilius, Gereon Kremer, Sebastian Junges, Stefan Schupp, and Erika Ábrahám. SMT-RAT: An open source C++ toolbox for strategic and parallel SMT solving. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 360–368. Springer, 2015.

[dMJ13] Leonardo de Moura and Dejan Jovanović. A model-constructing satisfiability calculus. In *Verification, Model Checking, and Abstract Interpretation*, pages 1–12. Springer Berlin Heidelberg, 2013.

[Irf18] Ahmed Irfan. *Incremental Linearization for Satisfiability and Verification Modulo Nonlinear Arithmetic and Transcendental Functions.* PhD thesis, University of Trento, 2018.

[JBDM13] Dejan Jovanović, Clark Barrett, and Leonardo De Moura. The design and implementation of the model constructing satisfiability calculus. In *2013 Formal Methods in Computer-Aided Design*, pages 173–180. IEEE, 2013.

[JdM12] Dejan Jovanović and Leonardo de Moura. Solving non-linear arithmetic. In *Automated Reasoning*, pages 339–354. Springer Berlin Heidelberg, 2012.

[Kre19] Gereon Kremer. *Cylindrical Algebraic Decomposition for Nonlinear Arithmetic Problems.* PhD thesis, RWTH Aachen University, 2019.

[NÁK21] Jasper Nalbach, Erika Ábrahám, and Gereon Kremer. Extending the fundamental theorem of linear programming for strict inequalities. In *ISSAC '21: International Symposium on Symbolic and Algebraic Computation*, pages 313–320. ACM, 2021.

[Nal20] Jasper Nalbach. *A novel adaption of the Simplex algorithm for linear real arithmetic*. Master thesis, RWTH Aachen University, 2020.

[NÁS+23] Jasper Nalbach, Erika Ábrahám, Philippe Specht, Christopher W. Brown, James H. Davenport, and Matthew England. Levelwise construction of a single cylindrical algebraic cell. *Journal of Symbolic Computation*, 2023.

[QFN] Satisfiability modulo theories library for QFNRA. Available at, https://clc-gitlab.cs.uiowa.edu:2443/SMT-LIB-benchmarks/QF_NRA.

[smt] SMT-RAT, a toolbox for strategic and parallel satisfiability modulo theories solving. Available at, https://github.com/ths-rwth/smtrat.

[Spe20] Philippe Specht. *A Level-wise Variant of Single Cell Construction in Cylindrical Algebraic Decomposition*. Bachelor thesis, RWTH Aachen University, 2020.

[Zam19] Aklima Zaman. Incremental linearization for sat modulo real arithmetic solving. Master's thesis, RWTH Aachen University, 2019.