Towards Conflict-Driven Learning for Virtual Substitution

Konstantin Korovin[1], Marek Košta[2], and Thomas Sturm[2]

[1] The University of Manchester, UK
korovin@cs.man.ac.uk
[2] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
mkosta@mpi-inf.mpg.de, sturm@mpi-inf.mpg.de

Abstract. We consider satisfiability modulo theory-solving for linear real arithmetic. Inspired by related work for the Fourier–Motzkin method, we combine virtual substitution with learning strategies. For the first time, we present virtual substitution—including our learning strategies—as a formal calculus. We prove soundness and completeness for that calculus. Some standard linear programming benchmarks computed with an experimental implementation of our calculus show that the integration of learning techniques into virtual substitution gives rise to considerable speedups. Our implementation is open-source and freely available.

1 Introduction

Recently there has been considerable progress in real satisfiability modulo theory-solving (SMT) triggered by the idea to adopt from Boolean satisfiability-solving (SAT) conflict analysis and learning techniques [6, 13, 9, 10, 7, 12, 1, 2]. On the other hand, during the past twenty years there has been a number of successful applications of real quantifier elimination methods, of which real SMT-solving is a special case [5, 18, 19, 17, 20, 21, 16].

Unfortunately, the SMT-solving community and the symbolic computation community working on quantifier elimination have been quite disconnected. The underlying frameworks are not really compatible, which makes it hard to even recognize that ideas from the respective other side are valuable. We would like to contribute to closing this gap by presenting as a calculus in the style of abstract DPLL [15] a special case of one successful approach to real quantifier elimination, viz. virtual substitution. In this paper we restrict ourselves to feasibility checking for systems of linear constraints. On that basis, we integrate conflict analysis and learning techniques in the spirit of the SMT ideas mentioned above.

Applying a very general technique like real quantifier elimination to that very special fragment can not compete with dedicated simplex-based methods. However, our approach has a considerable potential for generalizations, in particular to non-linear real arithmetic.

The plan of the paper is as follows: Section 2 provides a quick introduction into the concept of virtual substitution for readers not familiar with that theory.

In Section 3 we formalize virtual substitution for the special case considered here as a basic calculus where learning is only used to avoid cycles. This basic calculus essentially corresponds to a straightforward recursive implementation of the method. In Section 4 we proceed to an enhanced calculus featuring a learning technique based on linear algebra. Technically, that enhanced calculus is obtained by exchanging only one rule in the basic calculus so that soundness, completeness, and complexity results for the basic calculus can be mostly reused. Finally, in Section 5 we discuss computational experiments, comment on related work, and mention possible future work.

2 A Quick Introduction to Virtual Substitution

We consider formulas over the language $L = (0, 1, +, -, \geq)$ with the usual semantics over the real numbers. Given a quantifier-free L-formula Q an *elimination set* for Q and x is a finite nonempty set E of abstract elimination terms such that

$$\mathbb{R} \models \exists x[Q] \longleftrightarrow \bigvee_{e \in E} Q[x /\!\!/ e].$$

Here we are using a *virtual substitution* $[x /\!\!/ e]$, which is defined to map atomic formulas to quantifier-free formulas rather than terms to terms. In practice, this is combined with powerful simplification techniques for intermediate and final results.

In this paper we consider the special case where E contains only *linear elimination terms* of the form t/b, where t is an L-term and $b \in \mathbb{N} \setminus \{0\}$. Then virtual substitution can be defined as formally substituting t/b for x in a suitable extension language of L containing division and rewriting the result as an L-formula by dropping the positive denominator b. Although it is one of the principal strengths of virtual substitution methods that they can inherently deal with arbitrary Boolean combinations, we restrict ourselves here to conjunctions of atomic formulas as input.

Lemma 1 *Consider a formula $F = I_1 \geq 0 \wedge \cdots \wedge I_n \geq 0$. Let x be a variable occurring in F. Then an elimination set E for F and x can be computed as follows: Turn each inequality in F that contains x into an equation. Then formally solve the equation with respect to x, and add the solution to E.* □

As an example consider

$$F = -2x_1 - x_2 + 5 \geq 0 \wedge x_1 + x_2 + 5 \geq 0 \wedge -x_1 + x_2 + 3 \geq 0.$$

By Lemma 1, an elimination set for F and x_1 is given by

$$E = \{(-x_2 + 5)/2, -x_2 - 5, x_2 + 3\}.$$

$$\begin{aligned}\exists x_1[F] &\longleftrightarrow F[x_1 /\!\!/ (-x_2+5)/2] \vee F[x_1 /\!\!/ -x_2-5] \vee F[x_1 /\!\!/ x_2+3] \\ &= (0 \geq 0 \wedge x_2+15 \geq 0 \wedge 3x_2+1 \geq 0) \vee \\ &\quad (x_2+15 \geq 0 \wedge 0 \geq 0 \wedge 2x_2+8 \geq 0) \vee \\ &\quad (-3x_2-1 \geq 0 \wedge 2x_2+8 \geq 0 \wedge 0 \geq 0) \\ &\longleftrightarrow x_2+4 \geq 0.\end{aligned}$$

In practice, there are many optimizations, which lead to smaller elimination sets. For instance, when there are both lower and upper bounds on the considered variable, only one of these has to be taken into account, i.e., in our example $E = \{-x_2 - 5\}$ would actually be sufficient. For our calculi to be presented in the next sections, we are exclusively going to use the elimination sets according to Lemma 1. It is going to be crucial that each elimination term originates from exactly one constraint.

Consider the elimination of several variables, say, $\exists x_2 \exists x_1 F$. Eliminating x_1 as above we obtain $\exists x_2 \bigvee_{e \in E} F[x_1 /\!\!/ e]$. Before eliminating x_2 we can move $\exists x_2$ inside the disjunction and eliminate x_2 independently within each disjunct, which our calculi are implicitly going to do. This idea reduces the asymptotic worst-case complexity of the procedure from doubly exponential to singly exponential deterministic time in the input word length [22]. Recall that the Fourier–Motzkin method [14], in contrast, is doubly exponential [3, Section 4-4], [23], and that the simplex method is singly exponential in the worst-case as well [8], although it is known to perform much better than this bound on practical input.

For a more thorough introduction into the virtual substitution framework, we refer the reader to [11, 5].

3 A Basic Calculus

In this section we introduce a conflict-driven calculus for deciding satisfiability of systems

$$F = I_1 \geq 0 \wedge \cdots \wedge I_n \geq 0$$

of linear inequalities. By $\mathrm{var}(F)$ we denote the finite set of variables occurring in F. Without loss of generality we assume that $\mathrm{var}(F) \neq \emptyset$. Our calculus is based on the virtual substitution method, which we combine with conflict analysis and learning. In this section we are going to present a basic version with a primitive concept of learning, which leads to exhaustive enumeration of all test terms in the sense of [22, 11]. That calculus will serve as a basis for an enhanced calculus with stronger learning techniques, which we are going to present in the next section.

3.1 States

The states of our calculus are either $\top$ (sat), $\bot$ (unsat), or triplets (F, S, L). F is the input system, which will not be modified by any calculus rule.

S is a stack $\langle (x_1, \nu_1), \ldots, (x_k, \nu_k)\rangle$, growing to the right. Given a stack $S = \langle (x_1, \nu_1), \ldots, (x_k, \nu_k)\rangle$, we denote $\langle (x_1, \nu_1), \ldots, (x_{k+1}, \nu_{k+1})\rangle$ by $S\,|(x_{k+1}, \nu_{k+1})$. For $i \in \{1, \ldots, k-1\}$, ν_i is a pair (t_i, J_i), where $J_i \in \{I_1, \ldots, I_n\}$ and

$$t_i = -\frac{1}{b}\big(\sum_{x \in V} a_x x + a_0\big)$$

is derived from $J_i[x_1 /\!\!/ t_1] \ldots [x_{i-1} /\!\!/ t_{i-1}]$, which equals $\sum_{x \in V} a_x x + b x_i + a_0 \geq 0$ with $V = \mathrm{var}(F) \setminus \{x_1, \ldots, x_i\}$. In other words, t_i is the formal solution of $J_i = 0$ with respect to x_i subject to choices for $x_1, \ldots, x_{i-1}$ based on S. We call t_i an *elimination term*. Since for given $\langle (x_1, \nu_1), \ldots, (x_{i-1}, \nu_{i-1})\rangle$ the elimination term t_i is uniquely determined by J_i and x_i, we allow ourselves the convenient notation $x_i \leftarrow t_i\,(J_i)$ instead of $(x_i, (J_i, x_i))$. The last stack element ν_k is either a pair as described above, or "?" or "$\bot$." In the last two cases, we also write $x_i \leftarrow ?$ and $x_i \leftarrow \bot$, respectively.

Finally, we have a set L of *lemmas*, each of which is a disjunction of negated equations $\sum_{x \in \mathrm{var}(F)} a_x x \neq 0$, where $a_x \in \mathbb{Z}$.

For a given system F of linear inequalities, the initial state of our calculus is $(F, \langle\rangle, \emptyset)$.

3.2 Rules

Before discussing the rules of our calculus we need some definitions. A quantifier-free formula Q is *trivially inconsistent* if it is ground and equivalent to "false."

Let Q be a quantifier-free formula. Given $S = \langle x_1 \leftarrow t_1\,(J_1), \ldots, x_k \leftarrow t_k\,(J_k)\rangle$ we define the successive—in contrast to simultaneous—virtual substitution of S into Q as

$$Q/S = Q[x_1 /\!\!/ t_1] \ldots [x_k /\!\!/ t_k].$$

Here $[x_i /\!\!/ t_i]$ denotes the virtual substitution of t_i for x_i in the sense of [22, 11]. While virtual substitution in general maps atomic formulas to quantifier-free formulas, it is easy to see that for our linear inequalities one generally obtains atomic formulas. Specifically, for $\varrho \in \{=, >, \geq, \neq\}$ we have

$$(c_1 x_1 + J\ \varrho\ 0)\left[x_1 /\!\!/ b^{-1}K\right] = (c_1 K + bJ\ \varrho\ 0),$$

where $J = c_2 x_2 + \cdots + c_m x_m$, $K = a_2 x_2 + \cdots + a_m x_m$, c_i, $a_i \in \mathbb{Z}$, and $b \in \mathbb{N} \setminus \{0\}$. Our definition naturally generalizes to sets of quantifier-free formulas, in particular to L. Furthermore, we observe that the very special situation of our linear constraints allows to define virtual substitution even for terms:

$$(c_1 x_1 + J)\left[x_1 /\!\!/ b^{-1}K\right] = c_1 K + bJ.$$

Note that this differs from the standard definition of term substitution by dropping a positive integer denominator b.

Our first two rules decide for the next variable to be eliminated and assign to it one possible elimination term, respectively:

DECIDE :

$(F, S, L) \;\vdash\; (F, S \,|\, x_{k+1} \leftarrow ?, L)$

where S does not contain "?" or "$\bot$"

if F/S is not trivially inconsistent, and $x_{k+1} \in \mathrm{var}(F/S)$.

SUBSTITUTE :

$(F, S \,|\, x_k \leftarrow ?, L) \;\vdash\; (F, S \,|\, x_k \leftarrow \mathrm{eterm}(F, S, L, x_k), L)$

Given F, S, and $x \in \mathrm{var}(F/S)$, we denote the elimination set described in Lemma 1 by $E(F/S, x)$. Recall that every elimination originates from a single constraint. For our purposes here we are going to assume that $E(F/S)$ actually contains pairs (t, J), where t is an elimination term originating from J/S. Formally we write:

$$\mathbb{R} \models \exists x[F/S] \longleftrightarrow \bigvee_{(t,J)\in E(F/S,x)} F/S \,|\, x \leftarrow t\,(J). \tag{1}$$

We call an elimination term (t, J) *L-admissible*, if $L/(S \,|\, x \leftarrow t\,(J))$ is not trivially inconsistent. The elimination term function $\mathrm{eterm}(F, S, L, x)$ enumerates all L-admissible elements of $E(F/S, x)$. In the end it returns "$\bot$."

The following two rules handle the situation that our trial substitutions have led to an inconsistency. We learn just enough not to repeat our unlucky decisions in the future. Afterwards, we successively remove elements from the top of S until F/S is not trivially inconsistent anymore.

LEAF CONFLICT :

$\big(F, S, L\big) \;\vdash\; \big(F, S, L \cup \big\{\bigvee_{i=1}^{k} J_i \neq 0\big\}\big)$

where $S = \langle x_1 \leftarrow t_1\,(J_1), \ldots, x_k \leftarrow t_k\,(J_k)\rangle$, $k \geq 1$

if F/S is trivially inconsistent, and L/S is not trivially inconsistent.

LEAF BACKTRACK :

$(F, S \,|\, x_i \leftarrow t_i\,(J_i) \,|\, \ldots \,|\, x_k \leftarrow t_k\,(J_k), L) \;\vdash\; (F, S \,|\, x_i \leftarrow ?, L)$

if $L/S \,|\, x_i \leftarrow t_i\,(J_i)$ is triv. inconsistent, and L/S is not triv. inconsistent.

It is not hard to see that based on our limited learning in LEAF CONFLICT we leaf-backtrack exactly one step. This is going to be improved with our enhanced calculus in the next section.

The following two rules are concerned with the situation that the enumeration of some elimination set $E(F/S, x_k)$ has ended with eterm delivering "$\bot$" in SUBSTITUTE. Similarly to LEAF CONFLICT we learn in INNER CONFLICT not to return to the particular subproblem to eliminate x_k from F/S. Afterwards,

Inner Backtrack can backtrack exactly one step:

Inner Conflict :

$$\left(F, S \,|\, x_k \leftarrow \bot, L\right) \;\vdash\; \left(F, S \,|\, x_k \leftarrow \bot, L \cup \left\{\textstyle\bigvee_{i=1}^{k-1} J_i \neq 0\right\}\right)$$

where $S = \langle x_1 \leftarrow t_1\,(J_1), \ldots, x_{k-1} \leftarrow t_{k-1}\,(J_{k-1})\rangle$

if L/S is not trivially inconsistent.

Inner Backtrack :

$$\left(F, S \,|\, x_{k-1} \leftarrow t_{k-1}\,(J_{k-1}) \,|\, x_k \leftarrow \bot, L\right) \;\vdash\; \left(F, S \,|\, x_{k-1} \leftarrow ?, L\right)$$

if $L/S \,|\, x_{k-1} \leftarrow t_{k-1}\,(J_{k-1})$ is trivially inconsistent.

Finally, we fail when the elimination set for the first-chosen variable is exhausted. We succeed when F/S becomes ground and equivalent to "true:"

Fail :

$$\left(F, \langle x_1 \leftarrow \bot\rangle, L\right) \;\vdash\; \bot$$

Succeed :

$$\left(F, S, L\right) \;\vdash\; \top$$

if $\mathrm{var}(F/S) = \emptyset$, and F/S is equivalent to "true."

To conclude the discussion of our basic calculus we would like to point out that it is deterministic in the following sense: Every reachable state (F, S, L) matches the premise of exactly one of the rules.

3.3 Soundness

Lemma 2 (Invariants of the Calculus) *Consider*

$$(F, \langle\rangle, \emptyset) \vdash^n (F, S' \,|\, x_k \leftarrow \nu_k, L'),$$

where $S' = \langle x_1 \leftarrow t_1\,(J_1), \ldots, x_{k-1} \leftarrow t_{k-1}\,(J_{k-1})\rangle$.

(i) If $\nu_k = (t_k, J_k)$*, then for all* $l \in \{1, \ldots, k\}$ *the following holds:*

$$\mathbb{R} \models \exists[F/\langle x_1 \leftarrow t_1\,(J_1), \ldots, x_l \leftarrow t_l\,(J_l)\rangle] \longleftrightarrow \exists\left[F \wedge \textstyle\bigwedge_{i=1}^{l} J_i = 0\right].$$

(ii) For $\nu_k = ?$ *or* $\nu_k = \bot$ *the equivalence in (i) holds for all* $l \in \{1, \ldots, k-1\}$.

Proof. We simultaneously prove (i) and (ii) by induction on n.

The stack of the initial state is empty, and since $\bigwedge \emptyset$ is defined as "true," we obtain $\mathbb{R} \models \exists[F] \longleftrightarrow \exists[F]$. This proves both (i) and (ii) for $n = 0$.

Consider now $(F, \langle\rangle, \emptyset) \vdash^n (F, S, L) \vdash (F, S' \,|\, x_k \leftarrow \nu_k, L')$. Assume that both (i) and (ii) hold for (F, S, L). We show by case distinction on the rule applied in the last derivation step that $(F, S' \,|\, x_k \leftarrow \nu_k, L')$ satisfies both (i) and (ii).

DECIDE yields $S' = S$, and $\nu_k = ?$ so that we are in case (ii), which holds by the induction hypothesis.

With SUBSTITUTE we have $S = S' \,|\, x_k \leftarrow ?$ and either $\nu_k = (t_k, J_k)$ or $\nu_k = \bot$. In the first case we see that

$$\mathbb{R} \models \exists\big[\big[F/S'\big]/\langle x_k \leftarrow t_k\,(J_k)\rangle\big] \longleftrightarrow \exists\big[\big[F \wedge \textstyle\bigwedge_{i=1}^{k-1} J_i = 0\big] \wedge J_k = 0\big].$$

In the second case we can directly apply the induction hypothesis.

LEAF CONFLICT and INNER CONFLICT both yield $S' \,|\, x_k \leftarrow \nu_k = S$, and we apply the induction hypothesis.

With LEAF BACKTRACK and INNER BACKTRACK we obtain $S = S'T$ for some possibly empty stack T. Furthermore, $\nu_k = ?$, i.e., we are in case (ii). The fact that S' is a prefix of S allows us to apply the induction hypothesis.

Finally, FAIL and SUCCEED do not match our considered derivation. □

Lemma 3 *Consider* $(F, \langle\rangle, \emptyset) \vdash^n (F, S', L')$. *Then the following hold:*

(i) $\mathbb{R} \models F \longrightarrow \bigwedge L'$.
(ii) If $S' = S'_1 \,|\, x_k \leftarrow \bot$, *then* $\mathbb{R} \models \neg\exists[F/S'_1]$, *i.e.,* F/S'_1 *is unsatisfiable.*

Proof. To start with, we remark that FAIL and SUCCEED rules do not match our situation. We simultaneously prove (i) and (ii) by induction on n.

(i) For $n = 0$, since $\bigwedge \emptyset$ is defined as "true," we obtain $\mathbb{R} \models F \longrightarrow \bigwedge \emptyset$. Consider now $(F, \langle\rangle, \emptyset) \vdash^n (F, S, L) \vdash (F, S', L')$ and assume that both (i) and (ii) hold for n.

With DECIDE, LEAF BACKTRACK, INNER BACKTRACK, or SUBSTITUTE we have $L' = L$.

With LEAF CONFLICT we have $S' = S = \langle x_1 \leftarrow t_1\,(J_1), \ldots, x_k \leftarrow t_k\,(J_k)\rangle$ for $k \geq 1$. By definition of the rule, we know that F/S' is trivially inconsistent, in particular F/S' is unsatisfiable. Lemma 2(i) implies that $F \wedge \bigwedge_{i=1}^{k} J_i = 0$ is unsatisfiable. We equivalently transform

$$\mathbb{R} \models \neg\underline{\exists}\left[F \wedge \bigwedge_{i=1}^{k} J_i = 0\right] \longleftrightarrow \underline{\forall}\left[F \longrightarrow \bigvee_{i=1}^{k} J_i \neq 0\right],$$

i.e., F implies also the lemma newly learned in LEAF CONFLICT.

With INNER CONFLICT we have $S' = S = S'_1 \,|\, x_k \leftarrow \bot$, where $S'_1 = \langle x_1 \leftarrow t_1\,(J_1), \ldots, x_{k-1} \leftarrow t_{k-1}\,(J_{k-1})\rangle$. By the induction hypothesis for (ii), F/S'_1 is unsatisfiable. By Lemma 2(ii) it follows that also $F \wedge \bigwedge_{i=1}^{k-1} J_i = 0$ is unsatisfiable, and we proceed in analogy to the previous case.

(ii) For $n = 0$, we observe that the stack of the initial state is empty. Again, consider $(F, \langle\rangle, \emptyset) \vdash^n (F, S, L) \vdash (F, S', L')$ and assume that both (i) and (ii) hold for n.

With DECIDE, LEAF BACKTRACK, INNER BACKTRACK, or LEAF CONFLICT the top element of S' is not of the form $x_k \leftarrow \bot$.

With SUBSTITUTE assume that $S' = S'_1 \mid x_k \leftarrow \bot$. According to (1), we know that

$$\mathbb{R} \models \exists x_k[F/S'_1] \longleftrightarrow \bigvee_{(t_k,J_k)\in E(F/S'_1,x_k)} F/S'_1 \mid x_k \leftarrow t_k\,(J_k).$$

On the other hand, by (i), $\mathbb{R} \models F \longrightarrow \bigwedge L$, in particular

$$\mathbb{R} \models [F/S'_1 \mid x_k \leftarrow t_k\,(J_k)] \longrightarrow \bigwedge L/S'_1 \mid x_k \leftarrow t_k\,(J_k).$$

Inspection of SUBSTITUTE shows that $\mathrm{eterm}(F, S'_1, L, x_k) = \bot$, which means that $L/S'_1 \mid x_k \leftarrow t_k\,(J_k)$ is equivalent to "false" for all $(t_k, J_k) \in E(F/S'_1, x_k)$. Together $\mathbb{R} \models \exists x_k[F/S'_1] \longrightarrow$ false, i.e., F/S'_1 is unsatisfiable.

With INNER CONFLICT we have $S' = S$, and we can directly apply the induction hypothesis. □

Theorem 4 (Soundness) *(i) If $(F, \langle\rangle, \emptyset) \vdash^* \bot$, then F is unsatisfiable.*
(ii) If $(F, \langle\rangle, \emptyset) \vdash^* \top$, then F is satisfiable.

Proof. (i) Since $\bot$ is reachable only by FAIL we are in the following situation:

$$(F, \langle\rangle, \emptyset) \vdash^* (F, \langle x_1 \leftarrow \bot\rangle, L) \vdash \bot.$$

Now choose $S'_1 = \langle\rangle$ in Lemma 3(ii).

(ii) Since $\top$ is reachable only by SUCCEED we are in a situation:

$$(F, \langle\rangle, \emptyset) \vdash^* (F, S, L) \vdash \top,$$

where $S = \langle x_1 \leftarrow t_1\,(J_1), \ldots, x_k \leftarrow t_k\,(J_k)\rangle$, and F/S does not contain variables and is equivalent to "true." Denote by E_i the set $E(F/\langle x_1 \leftarrow t_1\,(J_1), \ldots, x_{i-1} \leftarrow t_{i-1}\,(J_{i-1})\rangle, x_i)$. According to (1), it is easy to see that $\exists x_1 \ldots \exists x_k F$ is equivalent to

$$\bigvee_{(s_k,M_k)\in E_k} \cdots \bigvee_{(s_1,M_1)\in E_1} F/\langle x_1 \leftarrow s_1\,(M_1), \ldots, x_k \leftarrow s_k\,(M_k)\rangle.$$

Furthermore, in exactly one of the above disjuncts we have $(s_1, M_1) = (t_1, J_1)$, $\ldots$, $(s_k, M_k) = (t_k, J_k)$, i.e., our stack S is substituted into F. Since F/S is equivalent to "true," the entire disjunction is equivalent to "true." Hence, F is satisfiable. □

3.4 Completeness

Our goal is to assign to each state $\mathcal{S} = (F, S, L)$ of a derivation with our calculus a weight $(n, k, h, l) \in \mathbb{N}^4$, which lexicographically decreases with each derivation step. For the definition of the weight we proceed in three stages:

1. The input constraints F in a given state $\mathcal{S}$ determine a finite abstract *F-tree* $\mathcal{T}$. Each node of $\mathcal{T}$ is either a *variable node* containing a variable chosen by DECIDE or a *term node* containing an elimination term or the unique root node, which is considered a term node, as well.

2. Using L in $\mathcal{S}$, each term node is labeled either *active* or *inactive*. Using S in $\mathcal{S}$ exactly one node of $\mathcal{T}$ is *selected*. If the selected node is a variable node, then it is additionally labeled "?" or "$\bot$."
3. From the labeled tree $\mathcal{T}$ we can finally determine a weight for our state $\mathcal{S}$.

We are now going to make precise these three steps. In the following we denote by τ_i the elimination term (t_i, J_i). Firstly, $\mathcal{T}$ is uniquely described by giving for each node N the path from the unique root node to N: The empty path () describes the root node. Given a path $(x_1, \tau_1, \ldots, x_k, \tau_k)$ to a term node, we obtain a variable node $(x_1, \tau_1, \ldots, x_k, \tau_k, x_{k+1})$ for each variable x_{k+1} occurring in $F/\langle x_1 \leftarrow \tau_1, \ldots, x_k \leftarrow \tau_k\rangle$. Given a path $(x_1, \tau_1, \ldots, x_k, \tau_k, x_{k+1})$ to a variable node, we obtain a term node with path $(x_1, \tau_1, \ldots, x_k, \tau_k, x_{k+1}, \tau_{k+1})$ for each elimination term $\tau_{k+1} \in E(F/\langle x_1 \leftarrow \tau_1, \ldots, x_k \leftarrow \tau_k\rangle, x_{k+1})$.

Secondly, a term node $(x_1, \tau_1, \ldots, x_k, \tau_k)$ is inactive if $L/\langle x_1 \leftarrow \tau_1, \ldots, x_k \leftarrow \tau_k\rangle$ is trivially inconsistent; otherwise it is active. The selected node essentially corresponds to the stack S: If $S = \langle x_1 \leftarrow \tau_1, \ldots, x_k \leftarrow ?\rangle$ or $S = \langle x_1 \leftarrow \tau_1, \ldots, x_k \leftarrow \bot\rangle$, then we select the variable node $(x_1, \tau_1, \ldots, x_{k-1}, \tau_{k-1}, x_k)$ and label it "?" or "$\bot$," respectively. If $S = \langle x_1 \leftarrow \tau_1, \ldots, x_k \leftarrow \tau_k\rangle$, then we select the term node $(x_1, \tau_1, \ldots, x_k, \tau_k)$.

Finally, the weight of $\mathcal{T}$ is $(n, k, h, l) \in \mathbb{N}^4$, where

- n is the number of active nodes,
- k is the number of inactive nodes on the path from the root to the selected node,
- h is the depth of $\mathcal{T}$ minus the length of the path from the root to the selected node,
- $l = 1$ if the selected node is labeled "?," otherwise $l = 0$.

Lemma 5 *Consider states (F, S, L) and (F, S', L') with weights (n, k, h, l) and (n', k', h', l'), respectively:*

$$(F, S, L) \vdash (F, S', L') \Longrightarrow (n, k, h, l) >_{lex} (n', k', h', l').$$

Proof. Decide yields $n = n'$, $k = k'$, $h > h'$, $l < l'$. Substitute depends on the return value of eterm; in either case $n = n'$, $k = k'$, $h \geq h'$, $l > l'$. Leaf Conflict and Inner Conflict yield $n > n'$, $k = k'$, $h = h'$, $l = l'$. Leaf Backtrack and Inner Backtrack yield $n = n'$, $k > k'$, $h < h'$, $l < l'$. Finally, notice that Fail and Succeed cannot produce (F, S', L'). □

At this point it is clear that our calculus performs only sound derivations and always terminates. For our main result we have to make sure that for all reachable states different from "$\bot$" and "$\top$" there is always at least one calculus rule applicable. We had already mentioned in Subsection 3.2 that in fact exactly one rule is applicable. For the sake of completeness, we state this formally once more.

Lemma 6 *Consider $(F, \langle\rangle, \emptyset) \vdash^* (F, S, L)$. There is one and only calculus rule and at least one state $\mathcal{S}$ such that $(F, S, L) \vdash \mathcal{S}$.* □

Theorem 7 (Completeness) *Given a system F of linear inequalities, every derivation beginning in the initial state $(F, \langle\rangle, \emptyset)$ terminates either in state "$\bot$" or in state "$\top$."*

Proof. By Lemma 5 every derivation starting in the initial state terminates. Assume for a contradiction that the final state is neither "$\bot$" nor "$\top$," then Lemma 6 admits another derivation step, a contradiction. □

3.5 Complexity

From earlier complexity results on the virtual substitution method [22] it is clear that there is a singly exponential upper bound on the worst-case complexity of the calculus bounding the number of substitutions as well as the overall number of derivation steps. This complexity bound persists when extending the approach to strict inequalities, which is straightforward. It is noteworthy that this bound is one exponential step smaller than the known lower bound for the Fourier–Motzkin method.

4 An Enhanced Calculus

Recall from Subsection 3.2 our definition of virtual substitution into terms, which differs from regular substitution by eliminating positive denominators. Similar to regular substitution, that virtual substitution can be expressed via linear combinations:

Lemma 8 *Consider a stack $S = \langle x_1 \leftarrow t_1\,(J_1), \ldots, x_k \leftarrow t_k\,(J_k)\rangle$ and a linear term K. Then one can compute $\alpha_1, \ldots, \alpha_k \in \mathbb{Q}$, $b \in \mathbb{N} \setminus \{0\}$ such that*

$$K/S = b\sum_{i=1}^{k} \alpha_i J_i + bK. \tag{2}$$

Proof. Let $K' = K[x_1/t_1]\ldots[x_k/t_k]$ be the result of *regular substitution* of stack S into K. It is clear that there exist $\alpha_1, \ldots, \alpha_k \in \mathbb{Q}$ such that $K' = K + \sum_{i=1}^{k} \alpha_i J_i$. On the other hand, there exists $b \in \mathbb{N} \setminus \{0\}$ such that $K/S = bK'$. It follows that (2) has a solution. Compute $b := \frac{K/S}{K'}$. Using this fixed b consider $f = 0$, where f is the recursive polynomial

$$bK + b\sum_{i=1}^{k} \alpha_i J_i - K/S \in \mathbb{Z}[\alpha_1, \ldots, \alpha_k][\mathrm{var}(F)].$$

Solve the linear system $p_1 = 0, \ldots, p_{|\mathrm{var}(F)|} = 0$, $q = 0$, where $p_1, \ldots, p_{|\mathrm{var}(F)|} \in \mathbb{Z}[\alpha_1, \ldots, \alpha_k]$ are the coefficients of the variables in f, and $q \in \mathbb{Z}[\alpha_1, \ldots, \alpha_k]$ is the constant term of f. □

We call $(K \geq 0) \in F$ a *conflicting inequality* with respect to S if $(K \geq 0)/S$ is trivially inconsistent.

Lemma 9 *Consider a state (F, S, L), where $S = \langle x_1 \leftarrow t_1(J_1), \ldots, x_k \leftarrow t_k(J_k)\rangle$, and let $(K \geq 0) \in F$ be a conflicting inequality with respect to S. Compute $\alpha_1, \ldots, \alpha_k \in \mathbb{Q}$ as in Lemma 8. Then*

$$\mathbb{R} \models F \longrightarrow \bigvee_{\substack{i=1 \\ \alpha_i < 0}}^{k} J_i \neq 0.$$

Proof. Assume that F holds. Using the fact that $K \geq 0$ is a conflicting inequality with respect to S and Lemma 8 we can compute $\alpha_1, \ldots, \alpha_k \in \mathbb{Q}$ such that

$$-1 = \operatorname{sgn}(K/S) = \operatorname{sgn}\left(\sum_{i=1}^{k} \alpha_i J_i + K\right) = \operatorname{sgn}\left(\sum_{\substack{i=1 \\ \alpha_i \geq 0}}^{k} \alpha_i J_i + \sum_{\substack{i=1 \\ \alpha_i < 0}}^{k} \alpha_i J_i + K\right).$$

On the other hand, it is clear that F implies non-negative linear combinations of its constraints, i.e., $\sum_{i=1,\alpha_i \geq 0}^{k} \alpha_i J_i + K \geq 0$. Assume for a contradiction that $\bigwedge_{i=1,\alpha_i<0}^{k} J_i = 0$, which implies $\sum_{i=1,\alpha_i<0}^{k} \alpha_i J_i \geq 0$. This leads to the contradiction $\sum_{i=1,\alpha_i<0}^{k} \alpha_i J_i + \sum_{i=1,\alpha_i \geq 0}^{k} \alpha_i J_i + K \geq 0$. □

Given a conflicting inequality $(K \geq 0) \in F$ with respect to some stack S, the following new rule uses a function lincomb computing $\alpha_1, \ldots, \alpha_k \in \mathbb{Q}$ as in Lemma 8 in order to learn the corresponding disjunction from Lemma 9:

Analyze Conflict :

$$\begin{array}{l} (F, S, L) \;\vdash\; \left(F, S, L \cup \left\{\bigvee_{i=1,\alpha_i<0}^{k} J_i \neq 0\right\}\right) \\ \quad \text{where } S = \langle x_1 \leftarrow t_1(J_1), \ldots, x_k \leftarrow t_k(J_k)\rangle \\ \quad \text{and } (\alpha_1, \ldots, \alpha_k) = \operatorname{lincomb}(S, K \geq 0). \end{array}$$

if F contains a conflicting inequality $K \geq 0$ with respect to S.

Our enhanced calculus replaces Leaf Conflict in favor of Analyze Conflict. Since for any choice of $\alpha_1, \ldots, \alpha_k \in \mathbb{Q}$ we have

$$\mathbb{R} \models \bigvee_{\substack{i=1 \\ \alpha_i < 0}}^{k} J_i \neq 0 \longrightarrow \bigvee_{i=1}^{k} J_i \neq 0,$$

the lemmas learned now are at least as strong as the ones learned with the basic calculus.

It is quite clear that one could even learn the stronger lemma $\bigvee_{\alpha_i<0} J_i > 0$. However, computational experiments indicate that this does not perform significantly better than the version described above.

4.1 Soundness

Inspection of Lemma 2 and its proof shows that that lemma remains valid also for Analyze Conflict instead of Leaf Conflict. From Lemma 9 it immediately follows that Lemma 3(i) remains valid. Finally, Lemma 3(ii) remains valid,

because as with LEAF CONFLICT the premise of the assertion is false. On these grounds the Soundness Theorem 4 for the basic calculus remains valid for our enhanced calculus.

4.2 Completeness

For the termination of our enhanced calculus, we construct weights for the states in the same way as in Subsection 3.4. Then Lemma 5 remains valid. For proving this we have to supplement that in the case of an application of our new ANALYZE CONFLICT rule we obtain $n > n'$, $k = k'$, $h = h'$, $l = l'$. Finally, Lemma 6 remains valid. On these grounds, the Completeness Theorem 7 remains valid.

4.3 Complexity

It is clear that the single exponential upper worst-case bound on the time complexity discussed for the basic calculus in Subsection 3.5 holds also for the enhanced calculus. It is open whether there is a polynomial upper bound, but we do not believe so.

5 Computational Experiments and Conclusions

We have implemented both calculi in the Reduce package Redlog [4]. Both Redlog and Reduce are freely available on SourceForge.[3]

In our implementation we use the selection of either upper or lower bounds, which we have briefly discussed after the example in Section 2. This is supplemented with the virtual substitution of either ∞ or $-\infty$, respectively. From a learning point of view, these substitutions can be essentially ignored. Thus, on the practical side, we are already a bit closer to the *virtual* substitution than in our theory developed here.

For getting an impression of the performance of our enhanced calculus in contrast to the basic calculus we have computed some of the smaller examples from Netlib:[4] afiro (28 constraints, 32 variables), blend (75 constrains, 83 variables), kb2 (44 constrains, 41 variables), sc50a (51 constrains, 48 variables), sc50b (51 constrains, 48 variables), sc105 (106 constrains, 103 variables). In addition, we have computed kmc, a Klee–Minty cube of dimension 50 [8].

The results are collected in Table 1. The columns "basic" and "enhanced" refer to our respective calculi. The columns "rlqe" refer to Redlog's implementation of quantifier elimination by virtual substitution, which accepts way more general input than our examples considered here. For each example we give the number of performed substitutions and the computation times for the various methods.

All our examples are originally linear optimization problems, while in their present form our calculi are limited to feasibility. Under "original" in Table 1

[3] `http://reduce-algebra.sourceforge.net/`
[4] `http://www.netlib.org/lp/data/`

	original			feasible			infeasible		
	basic	enhanced	rlqe	basic	enhanced	rlqe	basic	enhanced	rlqe
afiro	333,355	136	267	909,315	183	546	34,382,742	183	574
	14 s	52 ms	42 ms	37 s	70 ms	86 ms	23 min	70 ms	92 ms
blend	83	83	1,592	n/a	n/a	n/a	n/a	n/a	n/a
	103 ms	103 ms	706 ms	>8 h	>8 h	>8 h	>8 h	>8 h	>8 h
kb2	43	43	6,965	n/a	n/a	n/a	n/a	n/a	n/a
	17 ms	19 ms	7 s	>8 h	>8 h	>8 h	>8 h	>8 h	>8 h
kmc	50	50	50	50	50	50	50	50	48
	18 ms	19 ms	18 ms	18 ms	18 ms	18 ms	19 ms	20 ms	18 ms
sc50a	86	70	7,535	166,894	600	24,700	15,064,009	568	56,668
	16 ms	49 ms	2 s	7 s	673 ms	7 s	10 min	594 ms	17 s
sc50b	48	48	1,561	49	49	602	216,952	49	2,223
	14 ms	14 ms	353 ms	14 ms	16 ms	161 ms	13 s	18 ms	545 ms
sc105	n/a	4,450	1,785,226	n/a	6,701	n/a	n/a	5,427	n/a
	>8 h	39 s	21 min	>8 h	1 min	>8 h	>8 h	44 s	>8 h

Table 1. Netlib examples computed by our implementation. For each example the first line gives the number of performed substitutions, and the second line gives the CPU running times. All computations have been performed on a 2.4 GHz Intel Xeon E5-4640 running Debian Linux 64 bit.

we collect data for the original constraint sets ignoring the target functions. All these examples are in fact feasible so that there is a certain risk that we accidentally find feasible points very early.

This observation motivates the additional computation of two variants for each of these examples: As a first variant, we add a constraint $c \leq \lceil m \rceil$, where c is the objective function and $m \in \mathbb{Q} \setminus \mathbb{Z}$ is its known minimum subject to the constraints. The idea is that this makes the problem "just feasible." The corresponding results are collected in Table 1 under "feasible." Similarly, we add $c \leq \lfloor m \rfloor$ to render the problem infeasible but "almost feasible." The corresponding results are collected in Table 1 under "infeasible."

We observe that the number of substitutions performed with the enhanced calculus is in many cases orders of magnitude smaller than the number of substitutions performed by the basic calculus. The computation times with the enhanced calculus are often dramatically shorter and never significantly longer. The few slightly longer computation times can be explained by an overhead caused by solving systems of linear equations for finding the α_i in Analyze Conflict.

Although our calculus can not directly compete with simplex-based methods for solving systems of linear inequalities, it gives significant improvements over basic virtual substitution. Since virtual substitution can handle parameters and quantifier alternations [22, 11], we believe our calculus has a great potential in such extensions.

Concerning the general approach and the role of possible parameters, virtual substitution is somewhat similar to the Fourier–Motzkin method. However, recall from Subsection 3.5 that the worst case time complexity for our approach is one exponential step better than that of the Fourier–Motzkin method.

We are confident that generalization of our calculi to strict inequalities and negated equations is quite straightforward. We think that the treatment of conjunctive systems of polynomial inequalities of arbitrary degrees is a challenging but realistic next step. From a theoretical point of view, generalization of virtual substitution to arbitrary degrees is well-understood [24]. From a practical point of view, there are robust implementations in Redlog for degree two, which have been successfully applied to numerous problems from science and engineering during the past twenty years [5, 18, 19, 17, 20, 21, 16].

Acknowledgments

This research was supported in part by the German Transregional Collaborative Research Center SFB/TR 14 AVACS and by the ANR/DFG project SMArT. The project arose out of discussions at Dagstuhl Seminar 13411, *Deduction and Arithmetic*, held in October 2013.

References

1. Abraham, E., Loup, U., Corzilius, F., Sturm, T.: A lazy SMT-solver for a non-linear subset of real algebra. In: Proceedings of the SMT 2010. (2010)

2. Corzilius, F., Abraham, E.: Virtual substitution for SMT-solving. In: Fundamentals of Computation Theory. Volume 6914 of LNCS. (2011) 360–371
3. Dantzig, G.B.: Linear Programming and Extensions. Princeton University Press (1963)
4. Dolzmann, A., Sturm, T.: Redlog: Computer algebra meets computer logic. ACM SIGSAM Bulletin **31**(2) (1997) 2–9
5. Dolzmann, A., Sturm, T., Weispfenning, V.: Real quantifier elimination in practice. In: Algorithmic Algebra and Number Theory. Springer (1998) 221–247
6. Fränzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex Boolean structure. JSAT **1** (2007) 209–236
7. Jovanovic, D., de Moura, L.: Solving non-linear arithmetic. In: IJCAR 2012. Volume 7364 of LNAI. (2012) 339–354
8. Klee, V., Minty, G.: How good is the simplex algorithm? In: Proceedings of the Third Symposium on Inequalities. Academic Press (1972) 159–175
9. Korovin, K., Tsiskaridze, N., Voronkov, A.: Conflict resolution. In: Proceedings of the CP'09. Volume 5732 of LNCS. (2009) 509–523
10. Korovin, K., Voronkov, A.: Solving systems of linear inequalities by bound propagation. In: Proceedings of the CADE-23. Volume 6803 of LNCS. (2011) 369–383
11. Loos, R., Weispfenning, V.: Applying linear quantifier elimination. THE Computer Journal **36**(5) (1993) 450–462
12. Loup, U., Scheibler, K., Corzilius, F., Ábrahám, E., Becker, B.: A symbiosis of interval constraint propagation and cylindrical algebraic decomposition. In: Proceedings of the CADE-24. Volume 7898 of LNCS. (2013) 193–207
13. McMillan, K.L., Kuehlmann, A., Sagiv, M.: Generalizing DPLL to richer logics. In: Proceedings of the CAV'09. Volume 5643 of LNCS. (2009) 462–476
14. Motzkin, T.S.: Beiträge zur Theorie der linearen Ungleichungen. Doctoral dissertation, Universität Zürich (1936)
15. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). Journal of the ACM **53**(6) (2006) 937–977
16. Platzer, A., Quesel, J.D., Rümmer, P.: Real world verification. In: Proceedings of the CADE-22. Volume 5663 of LNCS. (2009) 485–501
17. Sofronie-Stokkermans, V.: Hierarchical and modular reasoning in complex theories: The case of local theory extensions. In: FroCoS 2007. Volume 4720 of LNCS. (2007) 47–71
18. Sturm, T.: Real Quantifier Elimination in Geometry. Doctoral dissertation, Universität Passau, Germany (1999)
19. Sturm, T., Tiwari, A.: Verification and synthesis using real quantifier elimination. In: Proceedings of the ISSAC 2011. ACM Press (2011) 329–336
20. Sturm, T., Weber, A., Abdel-Rahman, E.O., El Kahoui, M.: Investigating algebraic and logical algorithms to solve Hopf bifurcation problems in algebraic biology. Math. Comput. Sci. **2**(3) (2009) 493–515
21. Weber, A., Sturm, T., Abdel-Rahman, E.O.: Algorithmic global criteria for excluding oscillations. Bull. Math. Biol. **73**(4) (2011) 899–916
22. Weispfenning, V.: The complexity of linear problems in fields. J. Symb. Comput. **5**(1&2) (1988) 3–27
23. Weispfenning, V.: Parametric linear and quadratic optimization by elimination. Technical Report MIP-9404, Universität Passau, Germany (1994)
24. Weispfenning, V.: Quantifier elimination for real algebra—the quadratic case and beyond. Appl. Algebr. Eng. Comm. **8**(2) (1997) 85–101