

# An Instantiation-Based Approach for Solving Quantified Linear Arithmetic

Andrew Reynolds<sup>1</sup>, Tim King<sup>2</sup>, and Viktor Kuncak<sup>1</sup>

<sup>1</sup> École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

<sup>2</sup> Verimag

**Abstract.** This paper presents a framework to derive instantiation-based decision procedures for satisfiability of quantified formulas in first-order theories, including its correctness, implementation, and evaluation. Using this framework we derive decision procedures for linear real arithmetic (LRA) and linear integer arithmetic (LIA) formulas with one quantifier alternation. Our procedure can be integrated into the solving architecture used by typical SMT solvers. Experimental results on standardized benchmarks from model checking, static analysis, and synthesis show that our implementation of the procedure in the SMT solver CVC4 outperforms existing approaches for quantified linear arithmetic.

## 1 Introduction

Among the biggest challenges in automated reasoning is efficient support for quantifiers in the presence of background theories. Quantifiers enable direct encoding of a number of problems of interest, including synthesis of software fragments from specifications [26, 40, 45], construction of transfer functions for program analysis [32], invariant inference [11, 20], as well as analysis of properties that go beyond safety [8, 9].

The most commonly used complete method for deciding constraints over quantified theories is *quantifier elimination* [22, Section 2.7]. Quantifier elimination algorithms typically solve a more general problem, of transforming arbitrary quantified formula with free variables into a theory-equivalent formula with no quantifiers. However, depending on the particular variant of the language of constraints, performing actual quantifier elimination can have worse complexity than the decision problem [7], in part because it is required to give an answer on any formula, and the smallest formula resulting from quantifier elimination can be very large [53]. When the goal is to decide the satisfiability of quantified constraints, quantifier elimination may be doing unnecessary work. More importantly, procedures based on quantifier elimination often do not handle the underlying ground constraints in the most efficient way. Thus, quantifier elimination tends to be prohibitively expensive in practice.

It is often important to reason about formulas involving multiple theories, each of which may or may not support quantifier elimination. The goal is to obtain a framework for handling quantified formulas that is both complete for formulas belonging to decidable logics, and empirically effective when completeness guarantees are not known.

Recent work involving quantifier elimination [10, 24, 33, 36] has been motivated by avoiding worst-case performance by effectively computing an equisatisfiable set of

ground formulas in a lazy fashion. In the spirit of these approaches, we introduce an approach for establishing the satisfiability of formulas in quantified linear arithmetic based on a new *quantifier instantiation* framework. The use of quantifier instantiation for this task is motivated by the following.

- Procedures based on lazy quantifier instantiation typically establish satisfiability much faster than their theoretical complexity.
- Using quantifier instantiation for decidable fragments enables a uniform integration and composition with existing instantiation-based techniques [13, 14, 41], which are incomplete but work well in practice for undecidable fragments of first-order logic, and are widely used by modern SMT solvers.
- An important class of synthesis problems can be expressed as quantified formulas with one quantifier alternation. As shown in [40], solutions for these problems can be extracted from an unsatisfiable core of quantifier instantiations.

**Contributions** This paper makes the following contributions. First, we define a general class of instantiation-based procedures for establishing the satisfiability of quantified formulas in Section 2, and show that these procedures can be used in part as an approach for solving synthesis problems in Section 2.3. We demonstrate instances of the procedure are sound and complete for formulas over linear real arithmetic (LRA) and linear integer arithmetic (LIA) with one quantifier alternation in Sections 3 and 4, two quantified fragments for which many current SMT solvers do not have efficient support for. We show how our procedure can be integrated into the solving architecture used by SMT solvers in Section 5. We present an evaluation of an implementation of the procedures for LIA and LRA in the SMT solver CVC4, showing that it outperforms state-of-the-art approaches for quantified linear arithmetic used by other SMT solvers and automated theorem provers in Section 6.

**Related Work** Quantifier elimination has been used to, e.g., show decidability and classification of boolean algebras [44, 50], Presburger arithmetic [38], decidability of products [16, 34], [31, Chapter 12], and algebraically closed fields [49]. The original result on decidability of Presburger arithmetic is by Presburger [38]. The space bound for Presburger arithmetic was shown in [17]. The matching lower and upper bounds for Presburger arithmetic were shown in [7], see also [25, Lecture 24]. An analysis parameterized by the number of quantifier alternations is presented in [39]. A mechanically verified quantifier elimination algorithm was developed by Nipkow [35].

An approach for lazy quantifier elimination for linear real arithmetic was developed by Monniaux [33]. Integration of linear quantifier elimination into the solving algorithm used by SMT solvers was developed in [10], though the presented integration is not model driven. Other recent work on lazy quantifier elimination has been developed in the context of SMT-based model checking [24]. The most widely used techniques for quantifier instantiation in SMT were developed in [14], and later in [13, 18], which primarily focused on uninterpreted functions. Our approach for quantified linear arithmetic instantiates quantified formulas based on a lazy stream of candidate models, terminating when either it finds a finite set of instances are unsatisfiable, or discovers that the original formula is satisfiable. Other approaches in this spirit have been used to decide essentially uninterpreted fragment [19], and, more generally, theories having a locality property [3, 23]; these works do not directly apply to quantified linear

arithmetic. A recent approach for quantified formulas with one quantifier alternation has been developed in the SMT solver Yices [15], which does not treat linear integer arithmetic. The present paper builds upon our previous work for solving synthesis conjectures using quantifier instantiation in SMT [40], where an approach for quantified linear arithmetic was described without a specific method for selecting instances and without completeness guarantees. While the present paper focuses on linear arithmetic, where it outperforms existing approaches, we expect the presented framework to be relevant for other quantified theories. Among the examples of further decidable quantified constraints are quantified theories of term algebras [31, Chapter 23], [30, 46] and their extensions [12, 27, 42], feature trees [2, 51], and monadic second-order theories [52].

### 1.1 Preliminaries

We consider formulas in multi-sorted first-order logic. A *signature*  $\Sigma$  consists of a countable set of sort symbols and a set of function symbols. Given a signature  $\Sigma$ , well-sorted terms, atoms, literals, and formulas are defined as usual, and referred to respectively as  $\Sigma$ -terms. We denote by  $FV(t)$  the set of free variables occurring in the term  $t$ , and extend this notion to formulas. A  $\Sigma$ -term or formula is *ground* if it has no free variables. A term written  $t[\mathbf{k}]$  denotes a term whose free variables are in  $\mathbf{k}$ .

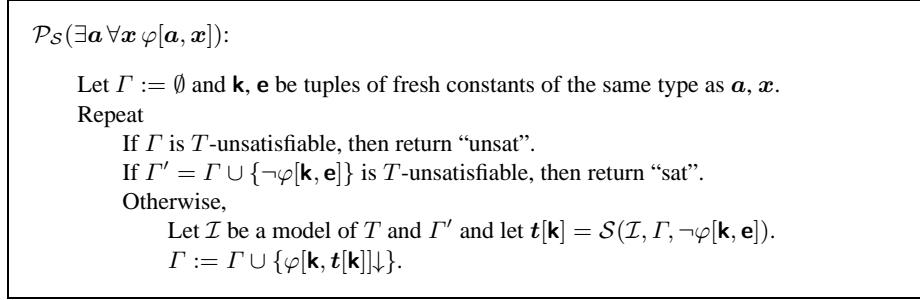
A  $\Sigma$ -interpretation  $\mathcal{I}$  maps

- each sort symbol  $\sigma \in \Sigma$  to a non-empty set  $\sigma^{\mathcal{I}}$ , the *domain* of  $\sigma$  in  $\mathcal{I}$ , and
- each function  $f \in \Sigma$  of sort  $\sigma_1 \times \dots \times \sigma_n \rightarrow \sigma$  to a total function  $f^{\mathcal{I}}$  of sort  $\sigma_1^{\mathcal{I}} \times \dots \times \sigma_n^{\mathcal{I}} \rightarrow \sigma^{\mathcal{I}}$  where  $n > 0$ , and to an element of  $\sigma^{\mathcal{I}}$  when  $n = 0$ .

We write  $t^{\mathcal{I}}$  to denote the interpretation of  $t$  in  $\mathcal{I}$ , defined inductively as usual. A satisfiability relation between  $\Sigma$ -interpretations and  $\Sigma$ -formulas, written  $\mathcal{I} \models \varphi$ , is also defined inductively as usual. In particular, we assume that  $\mathcal{I} \models \neg\varphi$  if and only if it is not the case that  $\mathcal{I} \models \varphi$ . We say that  $\mathcal{I}$  is a *model* of  $\varphi$  if  $\mathcal{I}$  satisfies  $\varphi$ . Formulas  $\varphi_1$  and  $\varphi_2$  are *equivalent (up to  $\mathbf{k}$ )* if they are satisfied by the same set of models (when restricted to the interpretation of variables  $\mathbf{k}$ ).

A *theory* is a pair  $T = (\Sigma, \mathbf{I})$  where  $\Sigma$  is a signature and  $\mathbf{I}$  is a non-empty set of  $\Sigma$ -interpretations, the *models* of  $T$ . Let  $\llbracket F \rrbracket_T$  denote the set of  $T$ -models of  $F$ . Observe that  $\llbracket \neg F \rrbracket_T = \mathbf{I} \setminus \llbracket F \rrbracket_T$ . A  $\Sigma$ -formula  $\varphi[\mathbf{x}]$  is  *$T$ -satisfiable* if it is satisfied by some interpretation in  $\mathbf{I}$  (i.e.  $\llbracket F \rrbracket_T \neq \emptyset$ ). Dually, a  $\Sigma$ -formula  $\varphi[\mathbf{x}]$  is  *$T$ -unsatisfiable* if it is satisfied by no interpretation in  $\mathbf{I}$  (i.e.  $\llbracket F \rrbracket_T = \emptyset$ ). A formula  $\varphi$  is  *$T$ -valid* if every model of  $T$  is a model of  $\varphi$  (i.e.,  $\llbracket F \rrbracket_T = \mathbf{I}$ ). Given a fragment  $\mathbf{L}$  of the language of  $\Sigma$ -formulas, a  $\Sigma$ -theory  $T$  is *satisfaction complete with respect to  $\mathbf{L}$*  if every closed  $T$ -satisfiable formula of  $\mathbf{L}$  is  $T$ -valid. In terms of set of models, satisfaction completeness means that  $\llbracket F \rrbracket_T \neq \emptyset$  implies  $\llbracket F \rrbracket_T = \mathbf{I}$ , or, in other words, for every  $F \in \mathbf{L}$  exactly one of the following two cases hold:  $\llbracket F \rrbracket_T = \emptyset$ , or  $\llbracket F \rrbracket_T = \mathbf{I}$ . If additionally  $\mathbf{L}$  is closed under negation, then, for every  $F \in \mathbf{L}$ , either  $F$  or  $\neg F$  is unsatisfiable.

A set  $\Gamma$  of formulas  *$T$ -entails* a  $\Sigma$ -formula  $\varphi$ , written  $\Gamma \models_T \varphi$ , if every model of  $T$  that satisfies all formulas in  $\Gamma$  satisfies  $\varphi$  as well. A set of literals  $M$  *propositionally entails* a formula  $\varphi$ , written  $M \models_p \varphi$ , if  $M$  entails  $\varphi$  when considering all atomic formulas in  $M \cup \varphi$  as propositional variables; such entailment is one of from propositional logic and is independent of the theory.



**Fig. 1.** An instantiation-based approach  $\mathcal{P}_S$  for determining the  $T$ -satisfiability of  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$  parameterized by selection function  $\mathcal{S}$ .

We write RA (resp. IA) to denote the theory of real (resp. integer) arithmetic. Its signature consists of the sort Real (resp. Int), the binary predicate symbols  $>$  and  $<$ , functions  $+$  and  $\cdot$  denoting addition and multiplication, and the constants of its sort interpreted as usual. We write  $t \leq s$  as shorthand for  $\neg(t > s)$ , and  $t \geq s$  as shorthand for  $\neg(t < s)$ . We write LRA (resp. LIA) to denote the language of linear real (resp. integer) arithmetic formulas, that is, whose literals are of the form  $(\neg)(c_1 \cdot x_1 + \dots + c_n \cdot x_n \bowtie c)$  where  $c_1, \dots, c_n, c$  and  $x_1, \dots, x_n$  are non-zero constants and distinct variables of sort Real (resp. Int) respectively, and  $\bowtie$  is one of  $>$ ,  $<$ , or  $\approx$ . For each literal of this form, there exists an equivalent literal that is in *solved form with respect to*  $x_i$  for each  $i = 1, \dots, n$ . That is, an LRA-literal is in solved form with respect to  $x$  if it is of the form  $(\neg)(x \bowtie t)$ , where  $x \notin FV(t)$ . Similarly, an LIA-literal is in solved form with respect to  $x$  if it is of the form  $(\neg)(c \cdot x \bowtie t)$ , where  $x \notin FV(t)$  and  $c$  is an integer constant greater than zero. For integer constants  $c_1$  and  $c_2$  and non-zero constant  $c$ , we write  $c_1 \equiv_c c_2$  to denote that  $c_1$  and  $c_2$  are congruent modulo  $c$ , that is  $(c_1 \bmod c) = (c_2 \bmod c)$ , and we write  $c \mid c_1$  if  $c$  divides  $c_1$ .

## 2 Quantifier Instantiation for Theories

In this section, we assume a fixed theory  $T$  and a language  $\mathbf{L}$  that is closed under negation and such that the satisfiability of finite sets of  $\mathbf{L}$  formulas modulo  $T$  is decidable. We present a procedure for checking satisfiability of formulas in the language  $\mathcal{Q}(\mathbf{L}) = \{\forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}] \mid \varphi[\mathbf{a}, \mathbf{x}] \in \mathbf{L}\}$ .

### 2.1 An Instantiation Procedure and Its Soundness

Figure 1 presents an instantiation-based approach for determining the satisfiability of a  $T$ -formulas  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$ , where  $\varphi[\mathbf{a}, \mathbf{x}]$  belongs to  $\mathbf{L}$ . The procedure introduces a tuple of distinct fresh constants  $\mathbf{k}$  of the same sort as  $\mathbf{a}$ , and  $\mathbf{e}$  of the same sort as  $\mathbf{x}$ . It maintains a set of formulas  $\Gamma$ , initially empty, and terminates when either  $\Gamma$  or  $\Gamma \cup \{\neg\varphi[\mathbf{k}, \mathbf{e}]\}$  is  $T$ -unsatisfiable. On each iteration, the procedure invokes the subprocedure  $\mathcal{S}$  (over which the procedure is parameterized), which returns a tuple of terms

$t[\mathbf{k}]$  whose free variables are a subset of  $\mathbf{k}$ . We then add to  $\Gamma$  the formula  $\varphi[\mathbf{k}, t[\mathbf{k}]]\downarrow$ , a formula equivalent to  $\varphi[\mathbf{k}, t[\mathbf{k}]]$  up to  $\mathbf{k}^3$ . We call  $\mathcal{S}$  the *selection function* of  $\mathcal{P}_S$ .

**Definition 1** A selection function (for  $\mathbf{L}$ ) takes as arguments an interpretation  $\mathcal{I}$ , a set of formulas  $\Gamma$ , and a formula  $\neg\varphi[\mathbf{k}, \mathbf{e}]$  in  $\mathbf{L}$ , where  $\mathcal{I} \models \Gamma \cup \neg\varphi[\mathbf{k}, \mathbf{e}]$ , and returns a tuple of terms  $t[\mathbf{k}]$  such that  $\varphi[\mathbf{k}, t[\mathbf{k}]]\downarrow$  is also in  $\mathbf{L}$ .

Note that a selection function is only defined if  $\mathcal{I}$  is a model for  $T$ ,  $\Gamma$  and  $\neg\varphi[\mathbf{k}, \mathbf{e}]$ . We first show that the procedure always returns correct results, regardless of the behavior of the selection function, leaving the termination question for the next subsection.

**Lemma 1** If  $\mathcal{P}_S$  terminates with “unsat”, then  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$  is  $T$ -unsatisfiable.

**Proof:** In this case, there exists a set  $\Gamma$  that is equivalent to  $\{\varphi[\mathbf{k}, t_1], \dots, \varphi[\mathbf{k}, t_p]\}$  and is  $T$ -unsatisfiable where  $\mathbf{k}$  are distinct fresh constants. Thus,  $\forall \mathbf{x} \varphi[\mathbf{k}, \mathbf{x}]$  is  $T$ -unsatisfiable. Since  $\mathbf{k}$  are distinct and fresh, we conclude that  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$  is  $T$ -unsatisfiable. ■

**Lemma 2** If  $\mathcal{P}_S$  terminates with “sat”, then  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$  is  $T$ -satisfiable.

**Proof:** In this case, there exists a set  $\Gamma$  equivalent to  $\{\varphi[\mathbf{k}, t_1], \dots, \varphi[\mathbf{k}, t_p]\}$  that is  $T$ -satisfiable, where  $\mathbf{k}$  are distinct fresh constants, and  $\Gamma'$  equivalent to  $\Gamma \cup \{\neg\varphi[\mathbf{k}, \mathbf{e}]\}$  that is  $T$ -unsatisfiable. Since  $\mathbf{e}$  do not occur in  $\Gamma$ , we have that  $\Gamma \cup \{\exists \mathbf{x} \neg\varphi[\mathbf{k}, \mathbf{x}]\}$  is  $T$ -unsatisfiable. Let  $\mathcal{I}$  be a model of  $\Gamma$ . Since  $\mathcal{I}$  is not a model of  $\Gamma'$ , it must be the case that  $\mathcal{I} \not\models \exists \mathbf{x} \neg\varphi[\mathbf{k}, \mathbf{x}]$ , and hence  $\mathcal{I}$  is a model for  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$ . ■

## 2.2 Termination of the Instantiation Procedure

The following properties of selection functions will be of interest.

**Definition 2 (Finite)** A selection function  $\mathcal{S}$  is finite for  $\varphi[\mathbf{k}, \mathbf{e}]$  if there exists a finite set  $\mathcal{S}^*(\varphi[\mathbf{k}, \mathbf{e}])$  such that  $\mathcal{S}(\mathcal{I}, \Gamma, \neg\varphi[\mathbf{k}, \mathbf{e}]) \in \mathcal{S}^*(\varphi[\mathbf{k}, \mathbf{e}])$  for all  $\mathcal{I}, \Gamma$ .

**Definition 3 (Monotonic)** A selection function  $\mathcal{S}$  is monotonic for  $\varphi[\mathbf{k}, \mathbf{e}]$  if whenever  $\Gamma \models \varphi[\mathbf{k}, t]$ , we have that  $\mathcal{S}(\mathcal{I}, \Gamma, \neg\varphi[\mathbf{k}, \mathbf{e}]) \neq t$ .

Observe that, if  $\mathcal{S}$  is a monotonic selection function, then for any finite list of terms  $t_1, \dots, t_n$  we have  $\mathcal{S}(\mathcal{I}, \{\varphi[\mathbf{k}, t_1]\downarrow, \dots, \varphi[\mathbf{k}, t_n]\downarrow\}, \neg\varphi[\mathbf{k}, \mathbf{e}]) \notin \{t_1, \dots, t_n\}$ .

**Definition 4 (Model-Preserving)** A selection function  $\mathcal{S}$  is model-preserving for  $\varphi[\mathbf{k}, \mathbf{e}]$  if whenever  $\mathcal{S}(\mathcal{I}, \Gamma, \neg\varphi[\mathbf{k}, \mathbf{e}]) = t$ , we have that  $\mathcal{I} \models \neg\varphi[\mathbf{k}, t]$ .

**Lemma 3** A selection function that is model-preserving for  $\varphi[\mathbf{k}, \mathbf{e}]$  is also monotonic for  $\varphi[\mathbf{k}, \mathbf{e}]$ .

**Proof:** Assume that  $\mathcal{S}$  is model-preserving for  $\varphi[\mathbf{k}, \mathbf{e}]$  and that  $\mathcal{S}(\mathcal{I}, \Gamma \cup \{\varphi[\mathbf{k}, t]\}, \neg\varphi[\mathbf{k}, \mathbf{e}]) = s$ . By definition of selection function, we have that  $\mathcal{I} \models \varphi[\mathbf{k}, t]$ . By definition of model-preserving, we have that  $\mathcal{I} \models \neg\varphi[\mathbf{k}, s]$ . Thus,  $s \neq t$  and  $\mathcal{S}$  is monotonic for  $\varphi[\mathbf{k}, \mathbf{e}]$ . ■

<sup>3</sup> We further comment on the details of operators  $\downarrow$  in Sections 3 and 4.

**Theorem 1** *If  $\mathcal{S}$  is finite and monotonic for  $\varphi[\mathbf{k}, \mathbf{e}]$  in  $\mathbf{L}$ , then  $\mathcal{P}_{\mathcal{S}}$  is a (terminating) decision procedure for the  $T$ -satisfiability of  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$ .*

**Proof:** Given a monotonic and finite  $\mathcal{S}$ , the procedure  $\mathcal{P}_{\mathcal{S}}$  can only execute a finite number of iterations. Assuming a decision procedure for determining the  $T$ -satisfiability of  $T$ -formulas in  $\mathbf{L}$ ,  $\mathcal{P}_{\mathcal{S}}(\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}])$  must terminate. By Lemmas 1 and 2,  $\mathcal{P}_{\mathcal{S}}$  is a decision procedure for the  $T$ -satisfiability of  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$ . ■

In this paper we will identify selection functions  $\mathcal{S}$  that are finite and monotonic for all  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$  residing in fragments  $\mathbf{L}$ . The fragments we consider are satisfaction complete. We consider satisfaction completeness to be a good guiding principle when choosing candidate logical fragments to which our method can be applied successfully.

### 2.3 Connection to Synthesis

The connection between quantifier elimination and synthesis has been shown fruitful in previous work [26]; it is one of our motivations for further improving quantified reasoning modulo theories. The procedure mentioned in this section can be used to synthesize functions from certain classes of specifications. Consider (second-order)  $T$ -formulas of the form:

$$\exists \mathbf{f} \forall \mathbf{x} \varphi[\mathbf{f}, \mathbf{x}] \quad (1)$$

where  $\varphi$  is a quantifier-free formula,  $\mathbf{x} = (x_1, \dots, x_n)$  is a tuple of variables of sort  $\sigma_i$  for  $i = 1, \dots, n$ , and  $\mathbf{f} = (f_1, \dots, f_m)$  is a tuple of functions of sort  $\sigma_1 \times \dots \times \sigma_n \rightarrow \tau_j$  for  $j = 1, \dots, m$ . We call such formulas *synthesis conjectures*. A synthesis conjecture is *single invocation* (over  $\mathbf{L}$ ) if it is equivalent to:

$$\exists \mathbf{f} \forall \mathbf{x} \psi[\mathbf{x}, \mathbf{f}(\mathbf{x})] \quad (2)$$

where  $\psi[\mathbf{x}, \mathbf{y}] \in \mathbf{L}$ . That is, functions from  $\mathbf{f}$  are applied to the tuple  $\mathbf{x}$  only. The formula (2) is equivalent to the (first-order) formula  $\forall \mathbf{x} \exists \mathbf{y} \psi[\mathbf{x}, \mathbf{y}]$ , whose negation

$$\exists \mathbf{x} \forall \mathbf{y} \neg \psi[\mathbf{x}, \mathbf{y}] \quad (3)$$

is suitable as an input to Figure 1. As observed in [40], solutions for single invocation synthesis conjectures can be extracted from an unsatisfiable core of instantiations when proving the unsatisfiability of (3). In particular, let  $\mathbf{k}$  be a set of distinct fresh variables of the same sort as  $\mathbf{x}$ , and say the set  $\{\neg \psi[\mathbf{k}, \mathbf{t}_1[\mathbf{k}]] \downarrow, \dots, \neg \psi[\mathbf{k}, \mathbf{t}_p[\mathbf{k}]] \downarrow\}$  is  $T$ -unsatisfiable where  $\mathbf{t}_i = (t_i^1[\mathbf{k}], \dots, t_i^m[\mathbf{k}])$  for  $i = 1, \dots, p$ . Then:

$$1 \leq j \leq m : f_j = \lambda \mathbf{x}. \text{ite}(\psi[\mathbf{x}, t_p^j[\mathbf{x}]], t_p^j[\mathbf{x}], (\dots \text{ite}(\psi[\mathbf{x}, t_2^j[\mathbf{x}]], t_2^j[\mathbf{x}], t_1^j[\mathbf{x}])))) \quad (4)$$

is a solution for  $\mathbf{f}$  in (2). We use the instantiation-based procedure in Figure 1 for discharging (3). In contrast to prior work [40], we here devise selection functions  $\mathcal{S}$  for  $\mathbf{L}$  that are finite and monotonic, obtaining a sound and complete method for synthesizing tuples of functions whose specification is a single invocation synthesis conjecture

$$\begin{array}{l}
\mathcal{S}_{SLRA}(\mathcal{I}, \Gamma, \neg\varphi[\mathbf{k}, \mathbf{e}]) \quad \text{where } \varphi[\mathbf{k}, \mathbf{e}] \text{ is } \bigvee_{i=1}^n e < l_i \vee \bigvee_{i=1}^m e > u_i, \text{ for } n > 0 \text{ or } m > 0 \\
\text{Return } \begin{cases} \ell_j, & \text{if } n > 0 \text{ and } \max\{\ell_1^{\mathcal{I}}, \dots, \ell_n^{\mathcal{I}}\} = \ell_j^{\mathcal{I}} \\ u_i, & \text{if } n = 0 \text{ and } \min\{u_1^{\mathcal{I}}, \dots, u_m^{\mathcal{I}}\} = u_i^{\mathcal{I}} \end{cases}
\end{array}$$

**Fig. 2.** A selection function  $\mathcal{S}_{SLRA}$  for a simple fragment of LRA.

over  $\mathbf{L}$ . In the following sections, we show such selection functions both for linear real arithmetic (LRA) and linear real arithmetic (LIA).

It is important to note that the solution (4) does not necessarily belong to the language  $\mathbf{L}$ , since there is no restriction on the selection functions for  $\mathbf{L}$  that restricts its return value  $t$  to terms in  $\mathbf{L}$ . For example, in our approach for linear real arithmetic,  $t$  may contain a free distinguished constant  $\delta$  representing an infinitesimal positive value, and in our approach for linear integer arithmetic,  $t$  may contain integer division. Additional steps may be necessary in practice for making (4) a computable function.

#### 2.4 Illustration: Instantiation for a Simple Fragment of LRA

We first present a selection function for a restricted class  $\mathbf{L}$  of LRA-formulas  $\exists \mathbf{a} \forall x, \varphi[\mathbf{a}, x]$ , namely whose universal quantifier is over single variable  $x$  of sort Real,  $\mathbf{a}$  are variables of sort Real, and whose (skolemized) body  $\varphi[\mathbf{k}, \mathbf{e}]$  is of the form:

$$(e < \ell_1 \vee \dots \vee e < \ell_n \vee e > u_1 \vee \dots \vee e > u_m) \quad (5)$$

where at least one of  $\{n, m\}$  is greater than zero, and  $e \notin FV(\ell_1, \dots, \ell_n, u_1, \dots, u_m)$ . Figure 2 gives a selection function for  $\mathcal{S}_{SLRA}$ . It considers the interpretation of terms  $\ell_1, \dots, \ell_n$  and  $u_1, \dots, u_m$  in a model  $\mathcal{I}$  of  $\Gamma$ . If  $n > 0$ , then  $\mathcal{S}_{SLRA}$  returns the  $\ell_j$  whose value is maximal in  $\mathcal{I}$ . If  $n = 0$ , then  $m > 0$  and  $\mathcal{S}_{SLRA}$  returns the  $u_i$  whose value is minimal in  $\mathcal{I}$ .

**Lemma 4**  $\mathcal{S}_{SLRA}$  is finite for  $\varphi[\mathbf{k}, \mathbf{e}]$ .

**Proof:** The terms returned by  $\mathcal{S}_{SLRA}$  are in  $\{\ell_1, \dots, \ell_n\}$  when  $n > 0$ , and in  $\{u_1, \dots, u_m\}$  when  $n = 0$ .  $\blacksquare$

**Lemma 5**  $\mathcal{S}_{SLRA}$  is monotonic for  $\varphi[\mathbf{k}, \mathbf{e}]$ .

**Proof:** Let  $\mathcal{I}$  be a model of LRA and  $\Gamma$ , where  $\Gamma \models \varphi[\mathbf{k}, t]$ , and assume by contradiction  $\mathcal{S}_{SLRA}$  returns  $t$ . Consider the case where  $n > 0$  and  $t = \ell_i$  for some  $i \in \{1, \dots, n\}$ . Since  $\mathcal{I} \models \Gamma \cup \neg\varphi[\mathbf{k}, \mathbf{e}]$ , it satisfies:

$$\begin{aligned}
& (\ell_i < \ell_1 \vee \dots \vee \ell_i < \ell_n \vee \ell_i > u_1 \vee \dots \vee \ell_i > u_m) \wedge \\
& e \geq \ell_1 \wedge \dots \wedge e \geq \ell_n \wedge e \leq u_1 \wedge \dots \wedge e \leq u_m
\end{aligned} \quad (6)$$

Assume  $\mathcal{I}$  satisfies  $\ell_i > u_k$  for some  $k \in \{1, \dots, m\}$ . We have  $\mathcal{I}$  also satisfies  $e \geq \ell_i$  and  $e \leq u_k$ , and thus  $e^{\mathcal{I}} \geq \ell_i^{\mathcal{I}} > u_k^{\mathcal{I}} \geq e^{\mathcal{I}}$ . Thus,  $\mathcal{I}$  must satisfy  $\ell_i < \ell_{k'}$  for some  $k' \in$

$\{1, \dots, n\}$ , and thus  $\max\{\ell_1^{\mathcal{I}}, \dots, \ell_n^{\mathcal{I}}\} \neq \ell_i^{\mathcal{I}}$ . Thus,  $\mathcal{S}_{\text{SLRA}}(\Gamma) \neq \ell_i$ . By symmetrical reasoning when  $n = 0, m > 0$  and  $t = u_j$  for some  $j \in \{1, \dots, m\}$ , we have that  $\mathcal{S}_{\text{SLRA}}(\Gamma) \neq u_j$ . Thus,  $\mathcal{S}_{\text{SLRA}}$  does not return  $t$ , and thus is monotonic for  $\varphi[\mathbf{k}, \mathbf{e}]$ . ■

*Example 1.* Consider the formula  $\forall x (x < b \vee x > a)$ . The negated skolemized form ( $\neg\varphi[\mathbf{k}, \mathbf{e}]$  in Figure 1) of this formula is equivalent to the formula  $b \leq e \wedge e \leq a$  where  $e$  is a fresh constant. A possible run of  $\mathcal{P}_{\mathcal{S}_{\text{SLRA}}}$  on this input is as follows, where  $\Gamma$  is initially  $\emptyset$  and on each iteration  $\Gamma' = \Gamma \cup \{b \leq e \wedge e \leq a\}$ .

#	$\Gamma$	$\Gamma'$	$t[\mathbf{k}]$	Add to $\Gamma$
1	sat	sat	$\max\{b^{\mathcal{I}}\} = b^{\mathcal{I}}$	$b < b \vee b > a$
2	sat	unsat		

In step 2, note that  $\Gamma' = \{b < b \vee b > a, b \leq e \wedge e \leq a\}$ , which is unsat. The run establishes that  $\exists ab \forall x (x > a \vee x < b)$  is LRA-satisfiable. ■

*Example 2.* Consider the formula  $\forall x (x < a \vee x < b)$ , whose skolemized negation is equivalent to  $e \geq a \wedge e \geq b$ . A possible run of  $\mathcal{P}_{\mathcal{S}_{\text{SLRA}}}$  on this input is as follows.

#	$\Gamma$	$\Gamma'$	$t[\mathbf{k}]$	Add to $\Gamma$
1	sat	sat	$\max\{a^{\mathcal{I}}, b^{\mathcal{I}}\} = a^{\mathcal{I}}$	$a < a \vee a < b$
2	sat	sat	$\max\{a^{\mathcal{I}}, b^{\mathcal{I}}\} = b^{\mathcal{I}}$	$b < a \vee b < b$
3	unsat			

Note that the formula added in step 1, equivalent to  $a < b$ , ensures that, in the second iteration,  $\max\{a^{\mathcal{I}}, b^{\mathcal{I}}\} \neq a^{\mathcal{I}}$ . The run establishes that  $\exists ab \forall x (x > a \vee x > b)$  is LRA-unsatisfiable, as expected in a linear order without endpoints. ■

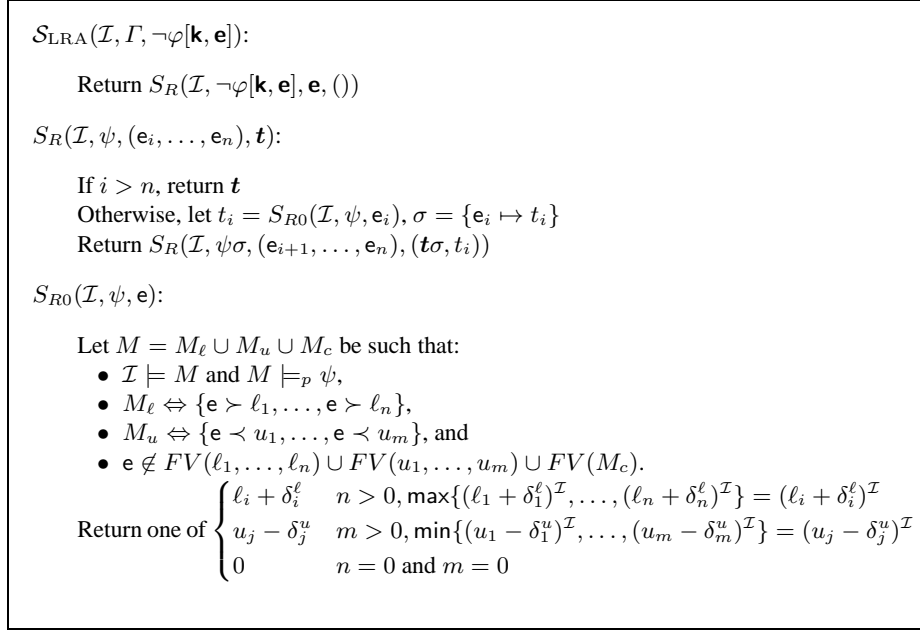
### 3 Instantiation for Quantifier-Free LRA-Formulas

Consider the case where  $\mathbf{a}$  and  $\mathbf{x}$  are vectors of Real variables and  $\mathbf{L}$  is the class of formulas  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$  where  $\varphi[\mathbf{a}, \mathbf{x}]$  is an arbitrary quantifier-free LRA-formula. We assume that equalities are eliminated from  $\varphi$  by the transformation:

$$t \approx 0 \rightsquigarrow 0 \leq t \wedge 0 \geq t$$

Figure 3 gives a selection function  $\mathcal{S}_{\text{LRA}}$  for LRA, which takes an interpretation  $\mathcal{I}$ , a set of formulas  $\Gamma$ , and the formula  $\neg\varphi[\mathbf{k}, \mathbf{e}]$ . It invokes the recursive procedure  $S_R$  which constructs a term corresponding to each variable in  $\mathbf{e}$ . Analogous to existing approaches for linear quantifier elimination [28, 35], our approach makes use of non-standard terms for symbolically representing substitutions. In particular, the terms we consider may involve a free distinguished constant  $\delta$ , representing an infinitesimal positive value. For each variable  $e_i$  from  $\mathbf{e}$ , the procedure  $S_R$  invokes the (non-deterministic) subprocedure  $S_{R0}$ , which chooses a term corresponding to  $e_i$  based on a set of literals  $M$  over the atoms of  $\psi$  which propositionally entail  $\psi$  and are satisfied by  $\mathcal{I}$ , which we call a *propositionally satisfying assignment* for  $\psi$ . We partition  $M$  into three sets





**Fig. 3.** A selection function  $\mathcal{S}_{\text{LRA}}$  for arbitrary quantifier-free LRA-formula  $\varphi[\mathbf{a}, x]$ . Each  $\prec$  is either  $<$  or  $\leq$ ;  $\delta_i^\ell$  is  $\delta$  if the  $i^{\text{th}}$  lower bound for  $\mathbf{e}$  is strict, and 0 otherwise. Similarly, each  $\succ$  is either  $>$  or  $\geq$ ;  $\delta_j^u$  is  $\delta$  if the  $j^{\text{th}}$  upper bound for  $\mathbf{e}$  is strict, and 0 otherwise.

$M_\ell$ ,  $M_u$  and  $M_c$ , where  $M_\ell$  contains literals that correspond to lower bounds for  $\mathbf{e}$ ,  $M_u$  contains literals that correspond to upper bounds for  $\mathbf{e}$ , and  $M_c$  contains the remaining literals. The sets  $M_\ell$  and  $M_u$  are equivalent to sets of literals that are in solved form with respect to  $\mathbf{e}$ . When  $M_\ell$  contains at least one literal, we may return the lower bound whose value is maximal according to  $\mathcal{I}$ , and similarly for  $M_u$ . If both  $M_\ell$  and  $M_u$  are empty, we return the term 0. When  $S_{R0}$  returns the term  $t_i$ , we apply the substitution  $\{\mathbf{e}_i \mapsto t_i\}$  to  $\psi$  and  $\mathbf{t}$ , and append  $t_i$  to  $\mathbf{t}$ . Terms returned by  $S_{R0}$  may involve the constant  $\delta$ . We define a satisfiability relation between models and formulas involving  $\delta$ , as well as the max and min function for terms involving  $\delta$  in the obvious way, such that  $(t_1 + c_1 \cdot \delta)^\mathcal{I} > (t_2 + c_2 \cdot \delta)^\mathcal{I}$  if either  $t_1^\mathcal{I} > t_2^\mathcal{I}$  or both  $t_1^\mathcal{I} = t_2^\mathcal{I}$  and  $c_1 > c_2$ .

Overall,  $\mathcal{S}_{\text{LRA}}$  returns a tuple of terms  $\mathbf{t}$ , after which we add the instance  $\varphi[\mathbf{k}, \mathbf{t}] \downarrow$  to  $\Gamma$  in Figure 1. We assume  $\downarrow$  eliminates occurrences of  $\delta$  by the following transformations, which is inspired by virtual term substitution [28]:

$$\delta < t \rightsquigarrow 0 < t \text{ and } \delta > t \rightsquigarrow 0 \geq t \text{ where } \delta \notin FV(t).$$

**Lemma 6**  $\mathcal{S}_{\text{LRA}}$  is finite for  $\varphi[\mathbf{k}, \mathbf{e}]$ .

**Proof:** We first show that only a finite number of terms can be returned by  $S_{R0}(\mathcal{I}, (\neg\varphi[\mathbf{k}, \mathbf{e}])\sigma, \mathbf{e}_i)$  for any  $\mathcal{I}, \sigma$ . Let  $A$  be the set of atoms occurring in  $\varphi[\mathbf{k}, \mathbf{e}_i]\sigma$ .

The literals in satisfying assignments of  $(\neg\varphi[\mathbf{k}, \mathbf{e}])\sigma$  are over these atoms. Let  $\{e_i < \ell_1, \dots, e_i < \ell_n, e_i > u_1, \dots, e_i > u_m\}$  be the set of atoms that are in solved form with respect to  $e_i$  that are equivalent to the atoms of  $A$  containing  $e_i$ , where  $e_i \notin FV(\ell_1, \dots, \ell_n, u_1, \dots, u_m)$ . The terms returned by  $S_{R0}(\mathcal{I}, (\neg\varphi[\mathbf{k}, \mathbf{e}])\sigma, e_i)$  are in  $\{0, \ell_1(+\delta), \dots, \ell_n(+\delta), u_1(-\delta), \dots, u_m(-\delta)\}$ . Since there are only a finite number of recursive calls to  $S_R$  within  $\mathcal{S}_{LRA}$ , and each call appends only a finite number of possible terms to  $\mathbf{t}$ , the set of possible return values of  $\mathcal{S}_{LRA}$  is finite, and thus it is finite for  $\varphi[\mathbf{k}, \mathbf{e}]$ . ■

**Lemma 7** *If  $\mathcal{I}$  is a model for LRA and quantifier-free formula  $\psi$ , then  $\mathcal{I}$  is also a model for  $\psi\{e \mapsto S_{R0}(\mathcal{I}, \psi, e)\}$ .*

**Proof:** Let  $M$  be a set of literals of the form described in the definition of  $S_{R0}$  for  $\mathcal{I}$ ,  $\psi$  and  $e$ . Consider the case where  $S_{R0}(\mathcal{I}, \psi, e) = \ell_i + \delta_i^\ell$  for some  $i$ , where  $n > 0$ . We show that  $\mathcal{I}$  satisfies  $M\{e \mapsto \ell_i + \delta_i^\ell\}$ . First, since  $\max\{(\ell_1 + \delta_1^\ell)^\mathcal{I}, \dots, (\ell_n + \delta_n^\ell)^\mathcal{I}\} = (\ell_i + \delta_i^\ell)^\mathcal{I}$ , we know that  $\mathcal{I}$  satisfies  $M_\ell\{e \mapsto \ell_i + \delta_i^\ell\}$ . In the case that the bound on  $e$  we consider is strict, that is,  $e > \ell_i \in M_\ell$ , then  $\delta_i^\ell$  is  $\delta$ , and  $\ell_i^\mathcal{I} < u_j^\mathcal{I}$  for all  $j \in \{1, \dots, m\}$ . Thus,  $\mathcal{I}$  satisfies  $(\ell_i + \delta < u_j) = (e < u_j)\{e \mapsto \ell_i + \delta\}$ . In the case that the bound on  $e$  we consider is non-strict, that is, if  $e \geq \ell_i \in M_\ell$ , then  $\delta_i^\ell$  is 0, and  $\ell_i^\mathcal{I} \leq u_j^\mathcal{I}$  for all  $j \in \{1, \dots, m\}$ . Thus,  $\mathcal{I}$  satisfies  $(\ell_i < u_j) = (e < u_j)\{e \mapsto \ell_i\}$ . In either case, we have that  $\mathcal{I}$  satisfies each literal in  $M_u\{e \mapsto \ell_i + \delta_i^\ell\}$ . Finally,  $\mathcal{I}$  clearly satisfies  $M_c\{e \mapsto \ell_i + \delta_i^\ell\} = M_c$ . The case when  $m > 0$  is symmetric to the case when  $n > 0$ . In the case where  $n = 0$  and  $m = 0$ , we have that  $\psi$  does not contain  $e$ , and  $\mathcal{I}$  satisfies  $M\{e \mapsto 0\}$ . In each case,  $\mathcal{I}$  satisfies  $M\{e \mapsto S_{R0}(\mathcal{I}, \psi, e)\}$ , which entails  $\psi\{e \mapsto S_{R0}(\mathcal{I}, \psi, e)\}$ , and thus the lemma holds. ■

**Lemma 8**  $\mathcal{S}_{LRA}$  is model-preserving for  $\varphi[\mathbf{k}, \mathbf{e}]$ .

**Proof:** By the definition of  $S_R$  and repeated applications of Lemma 7. ■

**Theorem 2**  $\mathcal{P}_{\mathcal{S}_{LRA}}$  is a sound and complete procedure for determining the LRA-satisfiability of  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$ .

**Proof:** By Theorem 1 and Lemma 3 of our framework as well as LRA-specific Lemma 6 and Lemma 8. ■

We illustrate the procedure through examples.  $S_{R0}$  is non-deterministic; we choose instantiations only based on the lower bounds  $M_\ell$  found in the procedure  $S_{R0}$ , though the procedure is free to base its instantiations on the upper bounds  $M_u$  as well. We underline the literal in  $M_\ell$  corresponding to the bound whose value is maximal in  $\mathcal{I}$ .  $\Gamma$  is initially empty and on each iteration  $\Gamma'$  is the union of  $\Gamma$  and the skolemized negation of the input formula. Each round of  $\mathcal{S}_{LRA}$  computes a tuple  $\mathbf{t}[\mathbf{k}]$ , which is used to instantiate our quantified formula in Figure 1. The last column shows the corresponding instance of the quantified formula after simplification, including the elimination of  $\delta$ .

*Example 3.* To demonstrate how non-strict bounds are handled, consider the formula  $\forall x (x \leq a \vee x \leq b)$ , whose skolemized negation is  $e > a \wedge e > b$ . A possible run of

$\mathcal{P}_{\mathcal{S}_{\text{LRA}}}$  on this input is as follows.

#	$\Gamma$	$\Gamma'$	$S_{R0}(\mathcal{I}, \Gamma, e)$			$t[\mathbf{k}]$	Add to $\Gamma$
			e	$M_\ell$	return		
1	sat	sat	$e$	$\{e > a, e > b\}$	$a + \delta$	$(a + \delta)$	$a < b$
2	sat	sat	$e$	$\{e > a, e \geq b\}$	$b + \delta$	$(b + \delta)$	$b < a$
3	unsat						

This run shows  $\exists ab \forall x (a \geq x \vee x \geq b)$  is LRA-unsatisfiable. The disjuncts of the instance  $a + \delta \leq a \vee a + \delta \leq b$  added to  $\Gamma$  on the first iteration simplify to  $\perp$  and  $a < b$  respectively. We similarly obtain  $b < a$  on the second iteration. ■

*Example 4.* To demonstrate how multiple universally quantified variables are handled, consider the formula  $\forall xy (x + y < a \vee x - y < b)$  whose skolemized negation is  $e_1 + e_2 \geq a \wedge e_1 - e_2 \geq b$ . A possible run of  $\mathcal{P}_{\mathcal{S}_{\text{LRA}}}$  is as follows.

#	$\Gamma$	$\Gamma'$	$S_{R0}(\mathcal{I}, \Gamma, e)$			$t[\mathbf{k}]$	Add to $\Gamma$
			e	$M_\ell$	return		
1	sat	sat	$e_1$	$\{e_1 \geq a - e_2, e_1 \geq b + e_2\}$	$b + e_2$	$(\frac{a+b}{2}, \frac{a-b}{2})$	$\perp$
			$e_2$	$\{e_2 \geq \frac{a-b}{2}\}$	$\frac{a-b}{2}$		
2	unsat						

This run shows  $\exists ab \forall xy (x + y < a \vee x - y < b)$  is LRA-unsatisfiable. The substitution for  $e_2$  is chosen based on  $M_\ell$  after applying the substitution  $\{e_1 \mapsto b + e_2\}$ . ■

*Example 5.* To demonstrate how quantified formulas with Boolean structure are handled, consider the formula  $\forall x ((a < x \wedge x < b) \vee x < a + b)$  whose skolemized negation is  $(a \geq e \vee e \geq b) \wedge e \geq a + b$ . A possible run of  $\mathcal{P}_{\mathcal{S}_{\text{LRA}}}$  is as follows.

#	$\Gamma$	$\Gamma'$	$S_{R0}(\mathcal{I}, \Gamma, e)$			$t[\mathbf{k}]$	Add to $\Gamma$
			e	$M_\ell$	return		
1	sat	sat	$e$	$\{e \geq a + b\}$	$a + b$	$(a + b)$	$0 < b \wedge a < 0$
2	sat	sat	$e$	$\{e \geq b, e \geq a + b\}$	$b$	$(b)$	$0 < a$
3	unsat						

This run shows  $\exists ab \forall x ((a < x \wedge x < b) \vee x < a + b)$  is LRA-unsatisfiable. On the first iteration, we assume that the propositionally satisfying assignment for  $\Gamma'$  included  $a \geq e$ , and hence  $e \geq b$  is not included as a lower bound on that iteration. On the second iteration, the solver must satisfy both  $b > 0$  and  $a < 0$ , which implies the model  $\mathcal{I}$  is such that  $(a + b)^{\mathcal{I}} > a^{\mathcal{I}}$  hence  $e \geq b$  must exist in  $M_\ell$ , and moreover  $b^{\mathcal{I}} > (a + b)^{\mathcal{I}}$  hence  $b$  must be the maximal lower bound for  $e$ . ■

*Example 6.* To demonstrate a case where a variable has no bounds, consider the formula  $\forall xy x \leq y$ , whose skolemized negation is  $e_1 > e_2$ . A possible run of  $\mathcal{P}_{\mathcal{S}_{\text{LRA}}}$  on this input is as follows.

#	$\Gamma$	$\Gamma'$	$S_{R0}(\mathcal{I}, \Gamma, e)$			$t[\mathbf{k}]$	Add to $\Gamma$
			e	$M_\ell$	return		
1	sat	sat	$e_1$	$\{e_1 \geq e_2\}$	$e_2 + \delta$	$(\delta, 0)$	$\perp$
			$e_2$	$\emptyset$	$0$		
2	unsat						

This run shows  $\forall xy x > y$  is LRA-unsatisfiable. Notice that after the substitution  $\{e_1 \mapsto e_2 + \delta\}$ , we have that  $\Gamma'$  contains neither an upper nor a lower bound for  $e_2$ , and hence we choose to return the value 0. ■

*Example 7.* To demonstrate a non-trivial case using the infinitesimal  $\delta$ , consider the formula  $\forall xy (x \leq 0 \vee y - 2 \cdot x \leq 0)$  whose skolemized negation is  $e_1 > 0 \wedge e_2 - 2 \cdot e_1 > 0$ . A possible run of  $\mathcal{P}_{\mathcal{S}_{\text{LRA}}}$  on this input is as follows.

#	$\Gamma$	$\Gamma'$	$S_{R0}(\mathcal{I}, \Gamma, \mathbf{e})$			$\mathbf{t}[\mathbf{k}]$	Add to $\Gamma$
			$\mathbf{e}$	$M_\ell$	return		
1	sat	sat	$e_1$	$\{e_1 > 0\}$	$\delta$	$(\delta, 3 \cdot \delta)$	$\perp$
			$e_2$	$\{e_2 > 2 \cdot \delta\}$	$3 \cdot \delta$		
2	unsat						

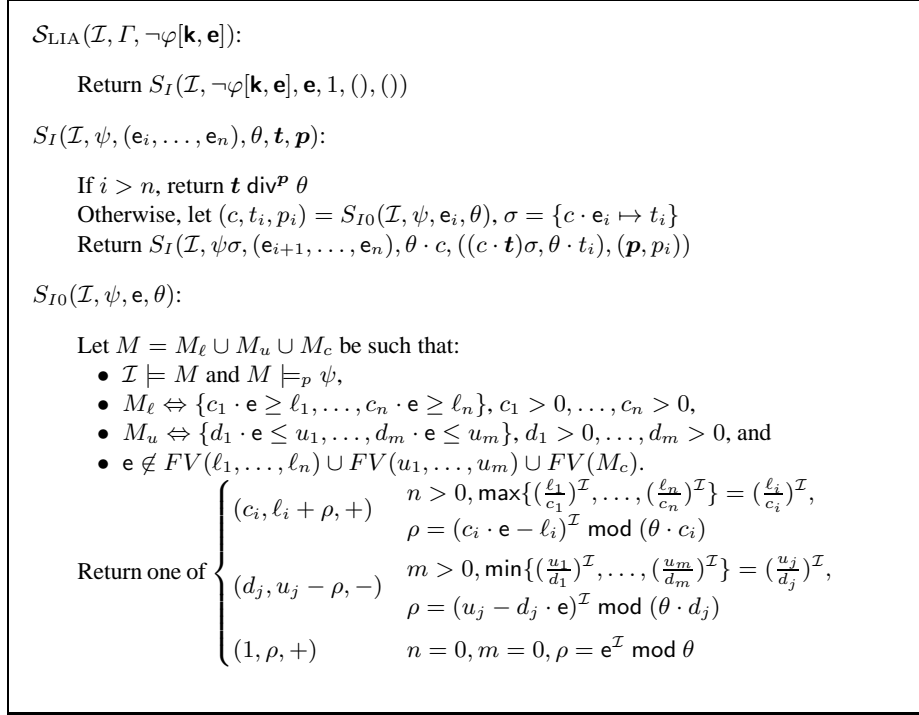
This run shows  $\forall xy x \leq 0 \vee y - 2 \cdot x \leq 0$  is LRA-unsatisfiable. ■

## 4 Instantiation for Quantifier-Free LIA-Formulas

We now turn our attention to the class of arbitrary LIA-formulas  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$ ,  $\mathbf{x}$  is a vector of  $\text{Int}$  variables, and where  $\varphi[\mathbf{a}, \mathbf{x}]$  is quantifier-free. We again assume all equalities are eliminated from  $\varphi$  by replacing them with a conjunction of inequalities.

Figure 4 gives a selection function  $\mathcal{S}_{\text{LIA}}$  for LIA. The procedure invokes the recursive procedure  $S_I$ , which takes as arguments  $\mathcal{I}$ ,  $\neg\varphi[\mathbf{k}, \mathbf{e}]$ , variables  $\mathbf{e}$  that we have yet to incorporate into the substitutions, an integer  $\theta$ , terms  $\mathbf{t}$  found as substitutions for variables from  $\mathbf{e}$  so far, and a tuple of symbols  $\mathbf{p}$  from  $\{+, -\}$  which we refer to as *polarities*. The role of  $\theta$  will be to capture divisibility relationships through the procedure, where  $\theta$  is initially 1. The procedure invokes a call to  $S_{I0}(\mathcal{I}, \psi, \mathbf{e}_i)$  which based on the propositionally satisfying assignment for  $\psi$  returns a tuple of the form  $(c, t_i, p_i)$ , where  $c$  is a constant,  $t_i$  is a term, and  $p_i$  is a polarity. The procedure for constructing the term  $t_i$  in the procedure  $S_{I0}$  is similar to the procedure  $S_{R0}$  in the previous section, where we find the lower bound of the form  $c_i \cdot \mathbf{e} \geq l_i$  such that the (rational) value  $(\frac{l_i}{c_i})^{\mathcal{I}}$  is maximal, and similarly for  $M_u$ . Additionally,  $S_{R0}$  adds a constant  $\rho$  to the maximal lower bound (resp. minimal lower bound). This constant ensures that the returned term  $t_i$  and  $\mathbf{e}$  are congruent modulo  $\theta \cdot c$  in  $\mathcal{I}$ , a fact which in part suffices to show the overall function to be model-preserving. It then constructs a *substitution with coefficients*  $\sigma$  of the form  $\{c \cdot \mathbf{e}_i \mapsto t_i\}$ . A substitution of this form may be applied to integer terms of the form  $c \cdot (d \cdot \mathbf{e}_i + s)$  where  $\mathbf{e}_i \notin FV(s)$ , where  $(c \cdot (d \cdot \mathbf{e}_i + s))\sigma$  is defined as  $d \cdot t_i + c \cdot s$ . Additionally, we define  $(s_1 \bowtie s_2)\sigma$  as  $(c \cdot s_1)\sigma \bowtie (c \cdot s_2)\sigma$  for  $\bowtie \in \{<, >\}$ , and thus we can apply  $\sigma$  to arbitrary LIA-formulas. After constructing  $\sigma$ , the procedure  $S_I$  invokes a recursive call where  $\sigma$  is applied to  $\psi$  and  $(c \cdot \mathbf{t})$ ,  $\theta$  is multiplied by  $c$ , the term  $\theta \cdot t_i$  is appended to  $\mathbf{t}$ , and  $p_i$  is appended to  $\mathbf{p}$ .

Overall,  $S_I$  returns a vector of terms  $(\mathbf{t} \text{ div}^{\mathbf{p}} \theta)$ , that is, integer division applied pairwise to the terms in  $\mathbf{t}$  and the constant  $\theta$ , where  $\mathbf{p}$  determines whether this division rounds up or down. We add the instance  $\varphi[\mathbf{k}, \mathbf{t} \text{ div}^{\mathbf{p}} \theta] \downarrow$  to  $\Gamma$  in Figure 1, where occurrences of integer division are eliminated by defining  $\varphi[\mathbf{k}, \mathbf{t} \text{ div}^{\mathbf{p}} \theta] \downarrow$  as  $\varphi[\mathbf{k}, \mathbf{d}] \wedge \theta \cdot \mathbf{d} \approx \mathbf{t} \pm^{\mathbf{p}} \mathbf{m} \wedge 0 \leq \mathbf{m} < \theta$ , where  $\mathbf{d}$  and  $\mathbf{m}$  are distinct fresh constants, and



**Fig. 4.** A selection function  $\mathcal{S}_{LIA}$  for arbitrary quantifier-free LIA-formula  $\varphi[\mathbf{a}, x]$ .

$\pm^{\mathbf{p}}$  is + if  $\mathbf{p}$  is + and analogously for  $-$ . Note that our selection function chooses  $\mathbf{p}$  such that integer division rounds up for terms coming from lower bounds, and rounds down for terms coming from upper bounds. This choice is not required for correctness, but can reduce the number of instances needed for showing unsatisfiability.

**Lemma 9**  $\mathcal{S}_{LIA}$  is finite for  $\varphi[\mathbf{k}, \mathbf{e}]$ .

**Proof:** First, we show that only a finite number of tuples are returned by  $S_{IO}(\mathcal{I}, (\neg\varphi[\mathbf{k}, \mathbf{e}])\sigma, \mathbf{e}_i, \theta)$  for any  $\mathcal{I}, \sigma, \mathbf{e}_i$  and finite  $\theta$ . Let  $A$  be the set of atoms occurring in  $\varphi[\mathbf{k}, \mathbf{e}_i]\sigma$ . The literals in satisfying assignments of  $(\neg\varphi[\mathbf{k}, \mathbf{e}])\sigma$  are over these atoms. Let  $\{c_1 \cdot \mathbf{e}_i < \ell_1, \dots, c_n \cdot \mathbf{e}_i < \ell_n, d_1 \cdot \mathbf{e}_i > u_1, \dots, d_m \cdot \mathbf{e}_i > u_m\}$  be the set of atoms that are in solved form with respect to  $\mathbf{e}_i$  that are equivalent to the atoms of  $A$  containing  $\mathbf{e}_i$ , where  $\mathbf{e}_i \notin FV(\ell_1, \dots, \ell_n, u_1, \dots, u_m)$  and  $c_1 > 0, \dots, c_n > 0, d_1 > 0, \dots, d_m > 0$ . The tuples returned by  $S_{IO}(\mathcal{I}, (\neg\varphi[\mathbf{k}, \mathbf{e}])\sigma, \mathbf{e}_i)$  are in the finite set:

$$\begin{aligned} & \{(c_i, \ell_i + \rho, +) \mid 1 \leq i \leq n\} \cup \{(d_j, u_j + 1 + \rho, -) \mid 1 \leq j \leq m\} \cup \\ & \{(d_j, u_j - \rho, -) \mid 1 \leq j \leq m\} \cup \{(c_i, \ell_i - 1 - \rho, +) \mid 1 \leq i \leq n\} \cup \{(1, \rho, +)\} \end{aligned}$$

and where  $0 \leq \rho < (\theta \cdot c)$ . Since  $c$  and  $\theta$  are finite, there are a finite number of tuples of this form. Since there are only a finite number of recursive calls to  $S_I$  within  $\mathcal{S}_{LIA}$ , and

each call modifies  $t$  based a finite number of possible tuples coming from the set above, the set of possible return values of  $S_{LIA}$  is finite, and thus it is finite for  $\varphi[\mathbf{k}, \mathbf{e}]$ . ■

**Lemma 10** *If  $\mathcal{I}$  is a model for LIA and a quantifier-free formula  $\psi$ ,  $\theta \geq 1$ , and  $S_{I0}(\mathcal{I}, \psi, \mathbf{e}, \theta) = (c, t, p)$ , then:*

1.  $(c \cdot \mathbf{e})^{\mathcal{I}} \equiv_{\theta \cdot c} t^{\mathcal{I}}$ , and
2.  $\mathcal{I} \models \psi\{c \cdot \mathbf{e} \mapsto t\}$ .

**Proof:** We first show part 1. In the case that  $n > 0$  and  $S_{I0}(\mathcal{I}, \psi, \mathbf{e}, \theta) = (c_i, \ell_i + \rho, +)$ , and

$$(\ell_i + \rho)^{\mathcal{I}} \equiv_{\theta \cdot c_i} (\ell_i + (c_i \cdot \mathbf{e} - \ell_i)^{\mathcal{I}} \bmod (\theta \cdot c_i))^{\mathcal{I}} \equiv_{\theta \cdot c_i} (c_i \cdot \mathbf{e})^{\mathcal{I}}.$$

In the case that  $m > 0$  and  $S_{I0}(\mathcal{I}, \psi, \mathbf{e}, \theta) = (d_j, u_j - \rho, -)$ , we have

$$(u_j - \rho)^{\mathcal{I}} \equiv_{\theta \cdot d_j} (u_j - (u_j - d_j \cdot \mathbf{e})^{\mathcal{I}} \bmod (\theta \cdot d_j))^{\mathcal{I}} \equiv_{\theta \cdot d_j} (d_j \cdot \mathbf{e})^{\mathcal{I}}.$$

In the case that  $n = 0, m = 0$ , and  $S_{I0}(\mathcal{I}, \psi, \mathbf{e}, \theta) = (1, \rho, +)$ , we have that  $\rho^{\mathcal{I}} \equiv_{\theta \cdot 1} (\mathbf{e}^{\mathcal{I}} \bmod \theta)^{\mathcal{I}} \equiv_{\theta \cdot 1} (1 \cdot \mathbf{e})^{\mathcal{I}}$ .

To show part 2, we first focus on the case where  $n > 0$  and  $S_{I0}(\mathcal{I}, \psi, \mathbf{e}, \theta) = (c_i, \ell_i + \rho, +)$ . We have that  $\rho = (c_i \cdot \mathbf{e} - \ell_i)^{\mathcal{I}} \bmod (\theta \cdot c_i)$ . Let  $M$  be a set of literals of the form described in the body of  $S_{I0}(\mathcal{I}, \psi, \mathbf{e})$ . We show that  $\mathcal{I}$  satisfies each literal in  $M\sigma$ , where  $\sigma = \{c_i \cdot \mathbf{e} \mapsto \ell_i + \rho\}$ . First, consider an atom in  $M_\ell\sigma$  that is equivalent to  $(c_j \cdot \mathbf{e} \geq \ell_j)\sigma$  for some  $j \in \{1, \dots, n\}$ . This is equivalent to  $(c_j \cdot c_i \cdot \mathbf{e} \geq c_i \cdot \ell_j)\sigma$ , which is equivalent to  $\frac{c_j \cdot c_i}{c_i} \cdot (\ell_i + \rho) \geq \frac{c_j \cdot c_i}{c_j} \cdot \ell_j$ , which is satisfied by  $\mathcal{I}$  since  $(\frac{\ell_i}{c_i})^{\mathcal{I}} \geq (\frac{\ell_j}{c_j})^{\mathcal{I}}$  by our selection of  $(c_i, \ell_i + \rho)$  and since  $\rho \geq 0$ . Second, consider the atom in  $M_u\sigma$  that is equivalent to  $(d_j \cdot \mathbf{e} \leq u_j)\sigma$  for some  $j \in \{1, \dots, m\}$ . Let  $\rho' = (c_i \cdot \mathbf{e} - \ell_i)^{\mathcal{I}}$ , which is greater than 0 since  $\mathcal{I}$  satisfies  $(c_i \cdot \mathbf{e} \geq \ell_i)$ . Since  $(c_i \cdot \mathbf{e})^{\mathcal{I}} = (\ell_i + \rho')^{\mathcal{I}}$ , we have that  $\mathcal{I}$  satisfies  $(d_j \cdot \mathbf{e} \leq u_j)\{c_i \cdot \mathbf{e} \mapsto \ell_i + \rho'\}$ , which is equivalent to  $(d_j \cdot (\ell_i + \rho') \leq c_i \cdot u_j)$ . Since  $\rho = \rho' \bmod (\theta \cdot c) \leq \rho'$ , we have that  $\mathcal{I}$  also satisfies  $(d_j \cdot (\ell_i + \rho) \leq c_i \cdot u_j)$ , which is  $(d_j \cdot \mathbf{e} \leq u_j)\sigma$ . Finally,  $\mathcal{I}$  satisfies  $M_c\sigma$  as  $M_c\sigma = M_c$  and  $\mathcal{I} \models M_c$ . Thus,  $\mathcal{I}$  satisfies  $M\sigma$ , which entails  $\psi\sigma$ . The case for when  $m > 0$  and  $S_{I0}(\mathcal{I}, \psi, \mathbf{e}, \theta) = (d_j, u_j - \rho, -)$  is symmetric. When  $n = 0, m = 0$ , and  $S_{I0}(\mathcal{I}, \psi, \mathbf{e}) = (1, \rho, +)$ , the assignment  $M$  does not contain  $\mathbf{e}$ , and thus  $\mathcal{I}$  satisfies  $M\{c \cdot \mathbf{e} \mapsto \rho\} = M$  and  $\psi\{c \cdot \mathbf{e} \mapsto \rho\}$ . ■

**Lemma 11** *Each recursive call to  $S_I(\mathcal{I}, \psi, (\mathbf{e}_1, \dots, \mathbf{e}_n), \theta, (t_1, \dots, t_{i-1}), \mathbf{p})$  within  $\mathcal{S}_{LIA}(\mathcal{I}, \Gamma, (\mathbf{e}_1, \dots, \mathbf{e}_n))$  is such that:*

1.  $\theta \mid t_j^{\mathcal{I}}$  for each  $1 \leq j < i$ , and
2.  $\mathcal{I} \models \psi$  and  $\psi$  is equivalent to  $\neg\varphi[\mathbf{k}, \mathbf{e}]\{\theta \cdot \mathbf{e}_1 \mapsto t_1\} \cdot \dots \cdot \{\theta \cdot \mathbf{e}_{i-1} \mapsto t_{i-1}\}$ .

**Proof:** Both statements clearly hold for the initial call to  $S_I$  in the body of  $\mathcal{S}_{LIA}$ . Now, assume both statements hold for some call to  $S_I(\mathcal{I}, \psi, (\mathbf{e}_i, \mathbf{e}'), \theta, (t_1, \dots, t_{i-1}), \mathbf{p})$ , and assume  $(c, t_i, p_i) = S_{I0}(\mathcal{I}, \psi, \mathbf{e}_i, \theta)$ . We show that both statements hold for the call to  $S_I(\mathcal{I}, \psi\sigma, \mathbf{e}', \theta \cdot c, ((c \cdot t_1)\sigma, \dots, (c \cdot t_{i-1})\sigma, \theta \cdot t_i), (\mathbf{p}, p_i))$ , where  $\sigma = \{c \cdot \mathbf{e}_i \mapsto t_i\}$ .

To show part 1, we have from Lemma 10 part 1 that:

$$(c \cdot \mathbf{e}_i)^{\mathcal{I}} \equiv_{\theta \cdot c} (t_i + \rho)^{\mathcal{I}} \tag{7}$$

Consider a  $t_j$  where  $1 \leq j < i$ , where by our assumption is such that  $\theta \mid t_j^{\mathcal{I}}$ , and thus  $\theta \cdot c \mid (c \cdot t_j)^{\mathcal{I}}$ . By (7), we have that  $\theta \cdot c \mid ((c \cdot t_j)\sigma)^{\mathcal{I}}$ . Also by (7), we have that  $c \mid (t_i + \rho)^{\mathcal{I}}$ , and thus  $\theta \cdot c \mid (\theta \cdot (t_i + \rho))^{\mathcal{I}}$ .

To show part 2, by our assumption,  $\mathcal{I} \models \psi$  and thus by Lemma 10 part 2 we have that  $\mathcal{I} \models \psi\sigma$ . By our assumption,  $\psi$  is equivalent to  $\neg\varphi[\mathbf{k}, \mathbf{e}]\{\theta \cdot \mathbf{e}_1 \mapsto t_1\} \cdot \dots \cdot \{\theta \cdot \mathbf{e}_{i-1} \mapsto t_{i-1}\}$ . Thus,  $\psi\sigma$  is equivalent to  $\neg\varphi[\mathbf{k}, \mathbf{e}]\{(\theta \cdot c) \cdot \mathbf{e}_1 \mapsto (c \cdot t_1)\sigma\} \cdot \dots \cdot \{(\theta \cdot c) \cdot \mathbf{e}_{i-1} \mapsto (c \cdot t_{i-1})\sigma\} \cdot \{(\theta \cdot c) \cdot \mathbf{e}_i \mapsto \theta \cdot (t_i + \rho)\}$ . Thus, the lemma holds. ■

**Lemma 12**  $\mathcal{S}_{\text{LIA}}$  is model-preserving for  $\varphi[\mathbf{k}, \mathbf{e}]$ .

**Proof:** Assume that  $\mathcal{S}_{\text{LIA}}(\mathcal{I}, \Gamma, \neg\varphi[\mathbf{k}, \mathbf{e}]) = \mathbf{t}$ , where  $\mathbf{e} = (e_1, \dots, e_n)$ , and  $\mathbf{t} = (t_1, \dots, t_n)$ . By Lemma 11 and the definition of  $\mathcal{S}_{\text{LIA}}$ , there is a  $\theta$  such that for each  $i = 1, \dots, n$ , term  $t_i$  is of the form  $s_i \text{ div}^p \theta$  where  $\theta \mid s_i^{\mathcal{I}}$ , and  $\mathcal{I} \models (\neg\varphi[\mathbf{k}, \mathbf{e}])\{\theta \cdot \mathbf{e}_1 \mapsto s_1\} \cdot \dots \cdot \{\theta \cdot \mathbf{e}_n \mapsto s_n\}$ . Thus,  $\mathcal{I}$  satisfies  $(\neg\varphi[\mathbf{k}, \mathbf{e}])\{\mathbf{e} \mapsto \mathbf{t}\} = \neg\varphi[\mathbf{k}, \mathbf{t}]$ , and thus  $\mathcal{S}_{\text{LIA}}$  is model-preserving for  $\varphi[\mathbf{k}, \mathbf{e}]$ . ■

**Theorem 3**  $\mathcal{P}_{\mathcal{S}_{\text{LIA}}}$  is a sound and complete procedure for determining the LIA-satisfiability of  $\exists \mathbf{a} \forall \mathbf{x} \varphi[\mathbf{a}, \mathbf{x}]$ .

**Proof:** By Theorem 1, Lemma 3, Lemma 9 and Lemma 12. ■

*Example 8.* To demonstrate a case involving a substitution with coefficients, consider the formula  $\forall xy (2 \cdot x < a \vee x + 3 \cdot y < b)$  whose negation is  $2 \cdot e_1 \geq a \wedge e_1 + 3 \cdot e_2 \geq b$ . A possible run of  $\mathcal{P}_{\mathcal{S}_{\text{LIA}}}$  on this input is as follows.

#	$\Gamma$	$\Gamma'$	$S_{I0}(\mathcal{I}, \Gamma, \mathbf{e}, \theta)$			$\mathbf{t}[\mathbf{k}]$	Add to $\Gamma$	
			$\mathbf{e}$	$\theta$	$M_\ell$			return
1	sat	sat	$e_1$	1	$\{2 \cdot e_1 \geq a, \dots\}$	$(2, a, +)$	$(6 \cdot a, 4 \cdot b - 2 \cdot a) \text{ div}^+ 12$	$\psi_1$
			$e_2$	2	$\{6 \cdot e_2 \geq 2 \cdot b - a\}$	$(6, 2 \cdot b - a, +)$		
2	unsat							

Thus,  $\exists ab \forall xy (2 \cdot x < a \vee x + 3 \cdot y < b)$  is LIA-unsatisfiable. We assume  $\rho = 0$  for all calls to  $S_{I0}$  in this run. Applying the substitution  $\{2 \cdot e_1 \mapsto a\}$  to  $e_1 + 3 \cdot e_2 \geq b$  results in the bound  $6 \cdot e_2 \geq 2 \cdot b - a$  for  $e_2$ . We add to  $\Gamma$  the instance  $\psi_1$ , which is equivalent to  $2 \cdot ((6 \cdot a) \text{ div}^+ 12) < a \vee (6 \cdot a) \text{ div}^+ 12 + 3 \cdot ((4 \cdot b - 2 \cdot a) \text{ div}^+ 12) < b$ . Applying normalization  $\downarrow$  to this formula results in a one that is LIA-unsatisfiable. ■

*Example 9.* To demonstrate a case necessarily involving a non-zero value of  $\rho$ , consider the formula  $\forall xy (3 \cdot x + y \not\approx a \vee 0 > y \vee y > 2)$  whose negation is  $3 \cdot e_1 + e_2 \approx a \wedge 0 \leq e_2 \wedge e_2 \leq 2$ , where  $\approx$  denotes the conjunction of non-strict upper and lower bounds. A possible run of  $\mathcal{P}_{\mathcal{S}_{\text{LIA}}}$  on this input is as follows.

#	$\Gamma$	$\Gamma'$	$S_{I0}(\mathcal{I}, \Gamma, \mathbf{e}, \theta)$			$\mathbf{t}[\mathbf{k}]$	Add to $\Gamma$	
			$\mathbf{e}$	$\theta$	$M_\ell$			return
1	sat	sat	$e_1$	1	$\{3 \cdot e_1 \geq a - e_2\}$	$(3, a - e_2, +)$	$(a, 0) \text{ div}^+ 3$	$\psi_1$
			$e_2$	3	$\{e_2 \geq 0\}$	$(1, 0, +)$		
2	sat	sat	$e_1$	1	$\{3 \cdot e_1 \geq a - e_2\}$	$(3, a - e_2, +)$	$(a - 1, 1) \text{ div}^+ 3$	$\psi_2$
			$e_2$	3	$\{e_2 \geq 0\}$	$(1, 1, +)$		
3	sat	sat	$e_1$	1	$\{3 \cdot e_1 \geq a - e_2\}$	$(3, a - e_2, +)$	$(a - 2, 2) \text{ div}^+ 3$	$\psi_3$
			$e_2$	3	$\{e_2 \geq 0\}$	$(1, 2, +)$		
4	unsat							

SMTQI( $T, \Gamma$ ):

If  $\Gamma$  is  $T$ -unsatisfiable, then return “unsat”.

Otherwise,

Let  $M$  be such that  $\mathcal{I} \models M$  and  $M \models_p \Gamma$  for some model  $\mathcal{I}$  of  $T$  and  $\Gamma$ .

$\Gamma' := \Gamma$ .

For each  $(\neg)A_i \in M$  where  $A_i \Leftrightarrow \forall \mathbf{x}.\varphi_i[\mathbf{x}]$ ,

$\Gamma' := \Gamma' \cup (A_i \vee B_i) \cup (B_i \Rightarrow \neg\varphi_i[\mathbf{e}_i])$ .

If  $A_i \in M$  and  $B_i \in M$ ,

$\Gamma' := \Gamma' \cup (A_i \Rightarrow \varphi_i[\mathcal{S}_i(\mathcal{I}, \Gamma, \neg\varphi_i[\mathbf{e}_i])])$

If  $\Gamma' = \Gamma$ , then return “sat”. Otherwise, return SMTQI( $T, \Gamma'$ ).

**Fig. 5.** Procedure SMTQI for SMT solving with quantifier instantiation, which determines the  $T$ -satisfiability of a ground set of  $T$ -formulas  $\Gamma$  in purified form with respect to quantified formulas.

This run shows  $\exists a \forall xy (2 \cdot x < a \vee x + 3 \cdot y < b)$  is LIA-unsatisfiable. On the first iteration, we assume that the model, call it  $\mathcal{I}_1$ , interprets all variables as 0, and hence the values chosen for  $e_1$  and  $e_2$  correspond to their maximal lower bounds in  $\mathcal{I}_1$ ,  $a - e_2$  and 0 respectively, where in each call to  $S_{I_0}$  we have  $\rho = 0$ . The instance  $\psi_1$  added to  $\Gamma$  on this iteration is equivalent to  $3 \cdot (a \operatorname{div}^+ 3) \not\approx a$  and implies that  $a^{\mathcal{I}} \not\equiv_3 0$  in all subsequent models  $\mathcal{I}$ . Thus, all models  $\mathcal{I}$  satisfying  $3 \cdot e_1 + e_2 \approx a$  must be such that  $e_2^{\mathcal{I}} \not\equiv_3 0$  as well. On the next iteration, we again find a model, call it  $\mathcal{I}_2$ , where the maximal lower bound for  $e_2$  is 0. By the above reasoning and since  $\mathcal{I}_2$  satisfies  $3 \cdot e_1 + e_2 \approx a \in \Gamma'$ , it must be that  $\rho = ((e_2 - 0)^{\mathcal{I}_2} \bmod 3) \neq 0$ . Assuming  $(e_2 - 0)^{\mathcal{I}_2} \equiv_3 1$ , the instance  $\psi_2$  is equivalent to  $3 \cdot ((a - 1) \operatorname{div}^+ 3) + 1 \not\approx a$ , which implies that  $a^{\mathcal{I}} \not\equiv_3 1$  in all subsequent models  $\mathcal{I}$ , and hence  $e_2^{\mathcal{I}} \not\equiv_3 1$ . The instance  $\psi_3$  is equivalent to  $3 \cdot ((a - 2) \operatorname{div}^+ 3) + 2 \not\approx a$  and implies that  $a^{\mathcal{I}} \not\equiv_3 2$ , which together with the two previous instances are  $T$ -unsatisfiable. ■

The number of possible return values of  $S_{I_0}$  is proportional to the size of  $\theta$ , which means the worst-case performance of our overall decision procedure is proportional to this size. While not shown here, our implementation takes steps to reduce the size of  $\theta$  by factoring out common divisors in  $\theta$  and the coefficients returned by  $S_{I_0}$ .

## 5 Integration in an SMT Solver

This section gives an overview of how an instantiation-based procedure for quantified formulas as described in Section 2 can be used in part for determining the satisfiability of inputs  $\Gamma$  having arbitrary Boolean structure that contain any number of quantified  $T$ -formulas and quantifier alternations. We describe the high-level details of the approach only, and comment informally on its properties here.

Figure 5 defines a procedure SMTQI which takes as input a theory  $T$  and a set of ground  $T$ -formulas  $\Gamma$  in *purified form* with respect to quantified formulas, that is, a



formula where all quantified formulas  $\forall \mathbf{x}.\varphi_i[\mathbf{x}]$  have been replaced by (uniquely associated) boolean variables  $A_i$ . We call such a variable  $A_i$  the *positive guard* of  $\forall \mathbf{x}.\varphi_i[\mathbf{x}]$  and write  $A_i \Leftrightarrow \forall \mathbf{x}.\varphi_i[\mathbf{x}]$  to denote  $A_i$  is the positive guard of  $\forall \mathbf{x}.\varphi_i[\mathbf{x}]$ . In each call to SMTQL, if  $\Gamma$  is  $T$ -unsatisfiable, then the procedure returns “unsat”. Otherwise, we find a model  $\mathcal{I}$  of  $T$  and  $\Gamma$  and a corresponding propositionally satisfying assignment  $M$ . We then build a new set of formulas  $\Gamma'$ , initially containing  $\Gamma$ , as follows. For each quantified formula  $\forall \mathbf{x}.\varphi_i[\mathbf{x}]$  whose positive guard  $A_i$  is in  $M$ , we add the formulas  $(A_i \vee B_i)$  and  $(B_i \Rightarrow \neg \varphi_i[e_i])$  to  $\Gamma'$  if we have not done so already, where  $B_i$  is a fresh Boolean variable, which we call the *negative guard* of  $\forall \mathbf{x}.\varphi_i[\mathbf{x}]$ . If both the positive and negative guards of  $\forall \mathbf{x}.\varphi_i[\mathbf{x}]$  have positive polarity in  $M$  (we will say such a quantified formula is *active in  $M$* ), we consider an instance of this quantified formula to  $\Gamma'$  based on its associated selection function  $\mathcal{S}_i$ . If  $\varphi_i$  has nested quantification, this instance will contain quantified formulas, which we purify in the same manner described above. If no new formulas are added to  $\Gamma'$  in this process, then the procedure returns “sat”. Otherwise, we call SMTQL on  $\Gamma'$ .

At a high level, the procedure in Figure 5 adds guarded instances of quantified formulas to an evolving set of formulas  $\Gamma$  until  $\Gamma$  is  $T$ -unsatisfiable, or a fixed point is reached. In this respect, the algorithm is similar to existing instantiation-based approaches used by SMT solvers for quantified formulas [13, 18]. However, the procedure differs from these approaches in the following ways. Firstly, when a universally quantified formula occurs with negative polarity in  $M$ , typical approaches add the clause  $(\neg A_i \Rightarrow \neg \varphi_i[e_i])$  to  $\Gamma$ . Here, we choose to add the clauses  $(A_i \vee B_i)$  and  $(B_i \Rightarrow \neg \varphi_i[e_i])$  instead. This allow us to consider both the positive and negative versions of quantified formulas simultaneously. As such, the algorithm only adds instances of quantified formulas where *both* the positive and negative guards have positive polarity in  $M$ . To ensure the model soundness of the approach (that is, the algorithm answers “sat” only if the input is indeed  $T$ -satisfiable), we require the following property of the set of literals  $M$  in the body of SMTQL:

If no quantified formula is active in  $M$ , then  $\Gamma \cup \{B_i\}$  is  $T$ -unsat for  $i = 1, \dots, n$ ,  
 where  $\{A_1, \dots, A_n\}$  is the set of the positive guards with positive polarity in  $M$ .

In other words, the set of literals  $M$  are chosen such that, if possible, at least one of  $B_1, \dots, B_n$  is true in  $M$ . In practice, this requirement can be met in a DPLL(T)-based SMT solver by instructing its underlying SAT solver, when it chooses a decision literal to add to  $M$ , to choose one of the unassigned negative guards of quantified formulas in  $\Gamma$ , if one exists, and add it with positive polarity.

We refer to the treatment of quantified formulas in Figure 5 as *counterexample-guided quantifier instantiation* [40]. A closely related approach is that of model-based quantifier instantiation [19], which like the approach described here, adds instances of quantified formulas based on models for their negations. This approach differs in its scope, in that it primarily targets quantified formulas having uninterpreted functions, whereas the approach described in Figure 5 targets quantified formulas having no uninterpreted functions. It also differs in that it uses a separate copy of the SMT solver as an oracle for checking the satisfiability of the negation of each quantified formula it

instantiates, whereas the approach described in Figure 5 uses a single instance of the SMT solver for doing these tasks simultaneously in its main solving loop.

### 5.1 Arbitrary Quantifier Alternation

The algorithm in Figure 5 can be used as a basis for handling quantified formulas with arbitrary quantifier alternations. Assume we rewrite formulas added to  $\Gamma'$  in the body SMTQI so they are in purified form with respect to quantified formulas, that is, we replace each quantified formula  $\forall x.\varphi_j[x]$  occurring in formulas  $B_i \Rightarrow \neg\varphi_i[e_i]$  and  $A_i \Rightarrow \varphi_i[\mathcal{S}_i(\mathcal{I}, \Gamma, e_i)]$  with its corresponding positive guard  $A_j$ . We give an intuition of how the procedure SMTQI handles such formulas in the following. A more comprehensive description is the subject of future work.

Consider the LIA-formula  $\forall x.\neg(\forall y.x > y)$ , call it  $\varphi_1$ , whose positive guard is  $A_1$ . Given the input  $\Gamma = \{A_1\}$ , the procedure SMTQI adds the formulas  $A_1 \vee B_1$  and  $B_1 \Rightarrow A_2$  to  $\Gamma$ , where  $A_2$  is the positive guard for  $(\forall y.e_1 > y)$  (call this formula  $\varphi_2$ ) where  $e_1$  is a fresh constant. On the second call to SMTQI, the satisfying assignment includes  $A_2$  and the procedure similarly adds the formulas  $A_2 \vee B_2$  and  $B_2 \Rightarrow e_1 \leq e_2$  to  $\Gamma$  where  $e_2$  is a fresh constant. On the third call to SMTQI, we have that  $\Gamma = \{A_1, B_1 \Rightarrow A_2, B_2 \Rightarrow e_1 \leq e_2, \dots\}$  is  $T$ -satisfiable, and we may choose a satisfying assignment  $M = \{A_1, B_1, A_2, B_2, e_1 \leq e_2, \dots\}$ . Both  $\varphi_1$  and  $\varphi_2$  are active in  $M$ . The literal  $e_1 \leq e_2$  is over the atoms of  $\varphi_2[e_2/y]$ , and we may add the formula  $A_2 \Rightarrow e_1 > e_2$  to  $\Gamma$  on this iteration, assuming our selection function for  $\varphi_2$  chose to return the maximal lower bound  $e_1$  for  $e_2$ . On the fourth call to SMTQI, we have that  $\Gamma = \{A_1, B_1 \Rightarrow A_2, A_2 \Rightarrow e_1 > e_2, \dots\}$ , and hence  $B_1$  or  $A_2$  cannot occur with positive polarity in a satisfying assignment  $M$  for this set. Hence, neither  $\varphi_1$  nor  $\varphi_2$  is active in  $M$  and the procedure SMTQI adds no instances to  $\Gamma$ , indicating that our input is satisfiable.

## 6 Experimental Evaluation

We have implemented the procedure in the SMT solver CVC4 [4] (version 1.5 pre-release). This section presents an evaluation of this implementation compared against other SMT solvers and first-order theorem provers. We considered all quantified benchmarks over 6 classes in the LRA and LIA logics of the SMT library [5]. The class **keymarea** are verification conditions coming from the Keymaera verification tool [37], **scholl** were used for simplification of non-convex polyhedra in [43], **psyco** were used for weakest precondition synthesis for compiler optimizations in [29], **uauto** correspond to verification conditions processed by the tool in [21], and the **tptp** classes correspond to simple arithmetic conjectures coming from the TPTP library [47]. We also considered a class of benchmarks **sygus** corresponding to first-order formulations of the 71 single-invocation synthesis conjectures taken from the conditional linear integer track of the 2015 edition of the syntax-guided synthesis competition [1]. All benchmarks are in the SMT version 2 format. For comparisons with automated theorem provers, they were converted to the TPTP format by the SMTtoTPTP conversion

	keymaera (222)		scholl (371)		tptp (25)		Total (621)	
	#	time	#	time	#	time	#	time
<b>CVC4</b>	<b>222</b>	1.5	<b>348</b>	1401.1	<b>25</b>	0.1	<b>595</b>	1402.7
<b>Z3</b>	<b>222</b>	1.6	327	360.1	<b>25</b>	0.3	574	362.0
<b>VampireZ3</b>	220	51.2	57	393.2	<b>25</b>	2.3	302	446.7
<b>Beagle</b>	<b>222</b>	377.9	53	577.9	<b>25</b>	29.7	300	985.5
<b>Vampire</b>	218	57.8	43	31.1	<b>25</b>	1.3	286	90.1
<b>Yices</b>	<b>222</b>	0.4	–	0.0	<b>25</b>	0.04	247	0.5
<b>ZenonArith</b>	205	13.8	25	452.9	14	0.9	244	467.7
<b>Princess</b>	202	1136.2	0	0.0	<b>25</b>	67.4	227	1203.6

**Fig. 6.** Results for LRA benchmarks, showing times (in seconds) and benchmarks solved by each solver and configuration over 3 benchmark classes with a 300s timeout. Yices (version 2.4.1) does not support nested quantification, hence it was not applicable for the scholl class.

	psyco (189)		tptp (46)		uauto (155)		sygus (71)		Total (461)	
	#	time	#	time	#	time	#	time	#	time
<b>CVC4</b>	<b>189</b>	89.1	<b>46</b>	0.3	<b>155</b>	1.6	<b>71</b>	22.0	<b>461</b>	112.9
<b>Z3</b>	183	31.6	<b>46</b>	0.6	<b>155</b>	1.2	<b>71</b>	18.3	455	51.7
<b>Beagle</b>	28	900.0	<b>46</b>	48.4	153	343.6	57	617.7	284	1909.7
<b>Princess</b>	13	513.4	<b>46</b>	48.0	<b>155</b>	201.9	68	418.8	282	1182.1
<b>VampireZ3</b>	4	3.1	36	4.7	<b>155</b>	106.3	55	151.8	250	265.9
<b>Vampire</b>	6	196.0	36	2.0	<b>155</b>	378.0	46	262.8	243	838.7
<b>ZenonArith</b>	0	0.0	30	1.9	154	15.0	28	1374.6	212	1391.5

**Fig. 7.** Results for LIA benchmarks, showing times (in seconds) and benchmarks solved by each solver and configuration over 4 benchmark classes with a 300s timeout.

tool [6]. We remark that all benchmarks consist purely of quantified formulas over linear arithmetic with very little, and in a majority of cases, no quantifier-free content.<sup>4</sup>

The results for the linear real and integer benchmarks are in Figures 6 and 7 respectively. Of the 7 benchmark classes, only one (the **scholl** class from LRA) had quantified formulas with nested quantification. As alluded to in Section 5, our implementation is capable of handling such problems by applying quantifier instantiation to quantified formulas as they become introduced by clauses added to  $\Gamma$  in Figure 5.

For both LRA and LIA, CVC4 solves the most benchmarks overall (595 and 461 respectively), and did not give a conflicting response with any of the other solvers. Moreover, we note that CVC4 solves *every* benchmark that does not involve nested quantification, giving confirming evidence that our approach and implementation for solving linear arithmetic with one quantifier alternation is indeed sound and complete. Although we do not claim completeness for formulas with nested quantification, CVC4 solves more benchmarks from the **scholl** class than any other solver.

The SMT solver Z3 (version 4.3.2) solves the next most benchmarks overall, solving 574 and 455 total for the LRA and LIA sets respectively. A technique [15] in the SMT solver Yices (version 2.4.1) is able to solve all benchmarks from the **keymaera** and

<sup>4</sup> Details can be found at <http://lara.epfl.ch/~reynolds/InstLA>.

**tptp** classes of LRA, both does not handle quantified formulas in LIA or with nested quantification. We also considered the entrants of the first-order typed theorems division (TFA) of CASC 25, the most recent competition for automated theorem provers [48].<sup>5</sup> For both benchmarks over LIA and LRA, the automated theorem provers trail the performance of CVC4 (and Z3) significantly. The best LRA automated theorem prover, VampireZ3, solves only 302 benchmarks, compared to 595 solved by CVC4. One of the best LIA automated theorem provers, Princess, solves 284 benchmarks, also notably less than the 461 solved by CVC4. Interestingly, Princess solver, implemented in the high-level Scala programming language (running on top of Java Virtual Machine), while notably trailing CVC4, still performs better than VampireZ3, a generally well-performing combination of two heavily optimized solvers independently specialized for ground and quantified constraints. This illustrates the non-triviality of the interaction between theory reasoning and quantified reasoning.

## 7 Conclusion

We have presented a class of instantiation-based procedures that are at the same time complete for quantified linear arithmetic and highly efficient in practice, outperforming existing implementations. Thanks to our framework we also obtain a simple and modular correctness arguments for soundness and completeness on formulas with one quantifier alternation.

For future work, we would like to show that the approach is complete for quantified linear arithmetic with arbitrary quantifier alternations, and develop heuristics for avoiding worst case performance for quantified integer arithmetic involving large coefficients. We also plan to develop selection functions for other theories, in particular, algebraic datatypes and fixed-width bitvectors, as well as for combined theories.

*Acknowledgements* We would like to thank Peter Baumgartner for his help with converting the benchmarks used in the evaluation to the TPTP format.

## References

1. R. Alur, R. Bodik, E. Dallal, D. Fisman, P. Garg, G. Juniwal, H. Kress-Gazit, P. Madhusudan, M. M. K. Martin, M. Raghothaman, S. Saha, S. A. Seshia, R. Singh, A. Solar-Lezama, E. Torlak, and A. Udupa. Syntax-guided synthesis. To Appear in Marktoberdorf NATO proceedings, 2014.
2. R. Backofen. A complete axiomatization of a theory with feature and arity constraints. *Journal of Logic Programming*, 24:37–72, 1995.
3. K. Bansal, A. Reynolds, T. King, C. Barrett, and T. Wies. Deciding local theory extensions via e-matching. In *Computer Aided Verification (CAV)*. Springer, 2015.
4. C. Barrett, C. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, and C. Tinelli. CVC4. In *Computer Aided Verification (CAV)*. Springer, 2011.

<sup>5</sup> We omit SPASS+T, which did not handle some classes of benchmarks due to restrictions on its input format, was comparable to the other automated theorem provers for the others. We show results for a bug fixed version of Beagle (version 0.9.30).

5. C. Barrett, A. Stump, and C. Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). [www.SMT-LIB.org](http://www.SMT-LIB.org), 2010.
6. P. Baumgartner. Smttotptp a converter for theorem proving formats. In *CADE-25*, volume 9195 of *Lecture Notes in Computer Science*. Springer, 2015.
7. L. Berman. The complexity of logical theories. *Theoretical Computer Science*, 11(1), 1980.
8. T. A. Beyene, S. Chaudhuri, C. Popeea, and A. Rybalchenko. A constraint-based approach to solving games on infinite graphs. In *POPL*, pages 221–234, 2014.
9. T. A. Beyene, C. Popeea, and A. Rybalchenko. Solving existentially quantified Horn clauses. In *CAV*, pages 869–882, 2013.
10. N. Bjørner. Linear quantifier elimination as an abstract decision procedure. In J. Giesl and R. Hähnle, editors, *IJCAR*, volume 6173 of *LNCS*, pages 316–330. Springer, 2010.
11. N. Bjørner, K. L. McMillan, and A. Rybalchenko. Program verification as satisfiability modulo theories. In *SMT@IJCAR*, pages 3–11, 2012.
12. H. Comon and C. Delor. Equational formulae with membership constraints. *Information and Computation*, 112(2):167–216, 1994.
13. L. M. de Moura and N. Bjørner. Efficient e-matching for SMT solvers. In F. Pfenning, editor, *CADE*, volume 4603 of *LNCS*, pages 183–198. Springer, 2007.
14. D. Detlefs, G. Nelson, and J. B. Saxe. Simplify: A theorem prover for program checking. Technical report, J. ACM, 2003.
15. B. Dutertre. Solving exists/forall problems with yices. In *Workshop on Satisfiability Modulo Theories*, 2015.
16. S. Feferman and R. L. Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.
17. J. Ferrante and C. W. Rackoff. *The Computational Complexity of Logical Theories*, volume 718 of *Lecture Notes in Mathematics*. Springer, 1979.
18. Y. Ge, C. Barrett, and C. Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In *CADE*, volume 4603 of *LNCS*. Springer, 2007.
19. Y. Ge and L. de Moura. Complete instantiation for quantified formulas in satisfiability modulo theories. In *Proceedings of CAV'09*, volume 5643 of *LNCS*. Springer, 2009.
20. S. Grebenshchikov, N. P. Lopes, C. Popeea, and A. Rybalchenko. Synthesizing software verifiers from proof rules. In *PLDI*, pages 405–416, 2012.
21. M. Heizmann, D. Dietsch, J. Leike, B. Musa, and A. Podelski. Ultimate automizer with array interpolation. In *TACAS*, 2015.
22. W. Hodges. *Model Theory*, volume 42 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1993.
23. S. Jacobs. Incremental instance generation in local reasoning. In *CAV '09*, pages 368–382, Berlin, Heidelberg, 2009. Springer-Verlag.
24. A. Komuravelli, A. Gurfinkel, and S. Chaki. SMT-based model checking for recursive programs. In *Computer Aided Verification*. Springer International Publishing, 2014.
25. D. Kozen. *Theory of Computation*. Springer, 2006.
26. V. Kuncak, M. Mayer, R. Piskac, and P. Suter. Complete functional synthesis. In B. G. Zorn and A. Aiken, editors, *PLDI*, pages 316–329. ACM, 2010.
27. V. Kuncak and M. Rinard. Structural subtyping of non-recursive types is decidable. In *Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2003.
28. R. Loos and V. Weispfenning. Applying linear quantifier elimination, 1993.
29. N. P. Lopes and J. Monteiro. Weakest precondition synthesis for compiler optimizations. In *VMCAI 2014*, pages 203–221, 2014.
30. M. J. Maher. Complete axiomatizations of the algebras of the finite, rational, and infinite trees. *IEEE Symposium on Logic in Computer Science*, 1988.
31. A. I. Mal'cev. *The Metamathematics of Algebraic Systems*, volume 66 of *Studies in Logic and The Foundations of Mathematics*. North-Holland, 1971.

32. D. Monniaux. Automatic modular abstractions for linear constraints. In *POPL 2009*, pages 140–151, 2009.
33. D. Monniaux. Quantifier elimination by lazy model enumeration. In T. Touili, B. Cook, and P. Jackson, editors, *CAV*, volume 6174 of *LNCS*, pages 585–599. Springer, 2010.
34. A. Mostowski. On direct products of theories. *Journal of Symbolic Logic*, 17(1), 1952.
35. T. Nipkow. Linear quantifier elimination. *Automated Reasoning*, pages 18–33, 2008.
36. A. Phan, N. Bjørner, and D. Monniaux. Anatomy of alternating quantifier satisfiability (work in progress). In *SMT 2012*, 2012.
37. A. Platzer, J.-D. Quesel, and P. Rümmer. Real world verification. In *Automated Deduction—CADE-22*, pages 485–501. Springer Berlin Heidelberg, 2009.
38. M. Presburger. über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt. In *Comptes Rendus du premier Congrès des Mathématiciens des Pays slaves, Warsawa*, pages 92–101, 1929.
39. C. R. Reddy and D. W. Loveland. Presburger arithmetic with bounded quantifier alternation. In *ACM STOC*, pages 320–325. ACM Press, 1978.
40. A. Reynolds, M. Deters, V. Kuncak, C. W. Barrett, and C. Tinelli. Counterexample guided quantifier instantiation for synthesis in CVC4. In *CAV*. Springer, 2015.
41. A. Reynolds, C. Tinelli, and L. D. Moura. Finding conflicting instances of quantified formulas in SMT. In *Formal Methods in Computer-Aided Design (FMCAD)*, 2014.
42. T. Rybina and A. Voronkov. A decision procedure for term algebras with queues. *ACM Transactions on Computational Logic (TOCL)*, 2(2):155–181, 2001.
43. C. Scholl, S. Disch, F. Pigorsch, and S. Kupferschmid. Using an smt solver and craig interpolation to detect and remove redundant linear constraints in representations of non-convex polyhedra. In *SMT*, pages 18–26. ACM, 2008.
44. T. Skolem. Untersuchungen über die Axiome des Klassenkalküls and über “Produktions- und Summationsprobleme”, welche gewisse Klassen von Aussagen betreffen. Skrifter utgit av Videnskapsselskapet i Kristiania, I. klasse, no. 3, Oslo, 1919.
45. T. Sturm and A. Tiwari. Verification and synthesis using real quantifier elimination. In *ISSAC 2011*, pages 329–336, 2011.
46. T. Sturm and V. Weispfenning. Quantifier elimination in term algebras: The case of finite languages. In *Computer Algebra in Scientific Computing (CASC)*, TUM Muenchen, 2002.
47. G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.
48. G. Sutcliffe. The 7th ijcar automated theorem proving system competition—casc-j7. *AI Communications*, pages 1–10, 2015.
49. A. Tarski. Arithmetical classes and types of algebraically closed and real-closed fields. *Bull. Amer. Math. Soc.*, 55(1), 1949.
50. A. Tarski. Arithmetical classes and types of boolean algebras. *Bull. Amer. Math. Soc.*, 55, 64, 1192, 1949.
51. R. Treinen. Feature trees over arbitrary structures. In P. Blackburn and M. de Rijke, editors, *Specifying Syntactic Structures*, chapter 7. CSLI Publications and FoLLI, 1997.
52. I. Walukiewicz. Monadic second-order logic on tree-like structures. *Theoretical Comput. Sci.*, 275(1–2):311–346, Mar. 2002.
53. V. Weispfenning. Complexity and uniformity of elimination in presburger arithmetic. In *ISSAC '97*, pages 48–53, New York, NY, USA, 1997. ACM.