On the Combination of the Bernays–Schönfinkel–Ramsey Fragment with Simple Linear Integer Arithmetic

Matthias Horbach, Marco Voigt, Christoph Weidenbach

To cite this version:

Matthias Horbach, Marco Voigt, Christoph Weidenbach. On the Combination of the Bernays–Schönfinkel–Ramsey Fragment with Simple Linear Integer Arithmetic. CADE 26 - 26th International Conference on Automated Deduction, Aug 2017, Gothenburg, Sweden. pp.77-94, 10.1007/978-3-319-63046-5_6 . hal-01592160

HAL Id: hal-01592160

https://hal.inria.fr/hal-01592160

Submitted on 27 Sep 2017

On the Combination of the Bernays–Schönfinkel–Ramsey Fragment with Simple Linear Integer Arithmetic

Matthias Horbach[1], Marco Voigt[1,2], and Christoph Weidenbach[1]

[1] Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany,
[2] Saarbrücken Graduate School of Computer Science, Saarland Informatics Campus, Saarbrücken, Germany

Abstract. In general, first-order predicate logic extended with linear integer arithmetic is undecidable. We show that the Bernays-Schönfinkel-Ramsey fragment ($\exists^*\forall^*$-sentences) extended with a restricted form of linear integer arithmetic is decidable via finite ground instantiation. The identified ground instances can be employed to restrict the search space of existing automated reasoning procedures considerably, e.g., when reasoning about quantified properties of array data structures formalized in Bradley, Manna, and Sipma's *array property fragment*. Typically, decision procedures for the array property fragment are based on an exhaustive instantiation of universally quantified array indices with all the ground index terms that occur in the formula at hand. Our results reveal that one can get along with significantly fewer instances.

Keywords: Bernays–Schönfinkel–Ramsey fragment · Linear integer arithmetic · Complete instantiation

1 Introduction

The Bernays-Schönfinkel-Ramsey (BSR) fragment comprises exactly the first-order logic prenex sentences with the $\exists^*\forall^*$ quantifier prefix, resulting in a CNF where all occurring function symbols are constants. Formulas may contain equality. Satisfiability of the BSR fragment is decidable and NExpTime-complete [19]. Its extension with linear arithmetic is undecidable [23, 10, 13, 11].

We prove decidability of the restriction to arithmetic constraints of the form $s \lhd t$, $x \lhd t$, where $\lhd$ is one of the standard relations $<, \leq, =, \neq, \geq, >$ and s, t are ground arithmetic terms, and $x \trianglelefteq y$, where $\trianglelefteq$ stands for $\leq$, $=$, or $\geq$. Underlying the result is the observation that similar to the finite model property of BSR, only finitely many instances of universally quantified clauses with arithmetic constraints need to be considered. Our construction is motivated by results from quantifier elimination [20] and hierarchic superposition [4, 3, 18, 11, 5]. In particular, the insights gained from the quantifier elimination side lead to instantiation methods that can result in significantly fewer instances than known, more naive approaches for comparable logic fragments generate, such as the original instantiation approach for the *array property fragment* [8, 6]. For example, consider the following two clauses ($\wedge$ and $\vee$ bind stronger than $\rightarrow$)

$$\begin{array}{rcl} x_2 \neq 5 \wedge R(x_1) & \rightarrow & Q(u_1, x_2) \\ y_1 < 7 \wedge y_2 \leq 2 & \rightarrow & Q(d, y_2) \vee R(y_1) \end{array}$$

where the variable u_1 ranges over a freely selectable domain, x_i, y_i are variables over the integers, and the constant d addresses an element of the same domain that u_1 ranges over. All occurring variables are implicitly universally quantified. Our main result reveals that this clause set is satisfiable if and only if a finite set of ground instances is satisfiable in which (i) u_1 is being instantiated with the constant d, (ii) x_2 and y_2 are being instantiated with the (abstract) integer values $5 + 1$ and $-\infty$, and (iii) x_1 and y_1 are being instantiated with $-\infty$ only. The instantiation does not need to consider the constraints $y_1 < 7$, $y_2 \leq 2$, because it is sufficient to explore the integers either from $-\infty$ upwards—in this case upper bounds on integer variables can be ignored—or from $+\infty$ downwards—ignoring lower bounds—, as is similarly done in linear quantifier elimination over the reals [20]. Moreover, instantiation does not need to consider the value $5 + 1$ for x_1 and y_1, motivated by the fact that the argument x_1 of R is not affected by the constraint $x_2 \neq 5$.

The abstract values $-\infty$ and $+\infty$ are represented by Skolem constants over the integers, together with defining axioms. For the example, we introduce the fresh Skolem constant $c_{-\infty}$ to represent $-\infty$ (a "sufficiently small" value) together with the axiom $c_{-\infty} < 2$, where 2 is the smallest occurring constant. Eventually, we obtain the ground clause set

$$\begin{array}{rcl} 5+1 \neq 5 \wedge R(c_{-\infty}) & \rightarrow & Q(d, 5+1) \\ c_{-\infty} \neq 5 \wedge R(c_{-\infty}) & \rightarrow & Q(d, c_{-\infty}) \\ c_{-\infty} < 7 \wedge 5+1 \leq 2 & \rightarrow & Q(d, 5+1) \vee R(c_{-\infty}) \\ c_{-\infty} < 7 \wedge c_{-\infty} \leq 2 & \rightarrow & Q(d, c_{-\infty}) \vee R(c_{-\infty}) \\ & c_{-\infty} < 2 & \end{array}$$

which has the model $\mathcal{A}$ with $c_{-\infty}^{\mathcal{A}} = 1$, $R^{\mathcal{A}} = \{1\}$, $Q^{\mathcal{A}} = \{(d, 6), (d, 1)\}$.

After developing our instantiation methodology in Section 3, we show in Sections 4 that our instantiation methods are also compatible with uninterpreted functions and additional background theories under certain syntactic restrictions. These results are based on an (un)satisfiability-preserving embedding of uninterpreted functions into BSR clauses. There are interesting known logic fragments that fall into this syntactic category: many-sorted clause sets over *stratified vocabularies* [1, 16], the *array property fragment* [8], and the *finite essentially uninterpreted fragment*, possibly extended with simple integer arithmetic [12]. Consequently, reasoning procedures for these fragments that employ forms of instantiation may benefit from our findings. The paper ends with a discussion in Section 5, where we consider the impact of our results on automated reasoning procedures for our and similar logic fragments and outline possible further improvements.

Due to space limitations, we mostly resort to sketches of proofs. The interested reader is referred to the extended version of the present paper [14].

2 Preliminaries

Hierarchic combinations of first-order logic with background theories build upon sorted logic with equality [4, 5]. We instantiate this framework with the BSR fragment and linear arithmetic over the integers as the *base theory*. The *base sort* $\mathcal{Z}$ shall always be interpreted by the integers $\mathbb{Z}$. For simplicity, we restrict our considerations to a single *free sort* $\mathcal{S}$, which may be freely interpreted as some nonempty domain, as usual.

We denote by $V_{\mathcal{Z}}$ a countably infinite set of base-sort variables. *Linear integer arithmetic (LIA) terms* are build from integer constants $0, 1, -1, 2, -2, \ldots$, the operators $+, -$, and the variables from $V_{\mathcal{Z}}$. We moreover allow base-sort constant symbols whose values have to be determined by an interpretation (*Skolem constants*). They can be conceived as existentially quantified. The LIA constraints we consider are of the form $s \triangleleft t$, where $\triangleleft \in \{<, \leq, =, \neq, \geq, >\}$ and s and t are either LIA variables or ground LIA terms.

In order to hierarchically extend the base theory by the BSR fragment, we introduce the free sort $\mathcal{S}$, a countably infinite set $V_{\mathcal{S}}$ of *free-sort variables*, a finite set Ω of *free (uninterpreted) constant symbols of sort* $\mathcal{S}$ and a finite set Π of *free predicate symbols* equipped with sort information. Note that every predicate symbol in Π has a finite, nonnegative arity and can have a mixed sort over the two sorts $\mathcal{Z}$ and $\mathcal{S}$, e.g. $P : \mathcal{Z} \times \mathcal{S} \times \mathcal{Z}$. We use the symbol $\approx$ to denote the built-in equality predicate on $\mathcal{S}$. To avoid confusion, we tacitly assume that no constant or predicate symbol is overloaded, i.e. they have a unique sort.

Definition 1 (BSR with Simple Linear Integer Constraints–BSR(SLI)). *A* BSR(SLI) clause *has the form $\Lambda \,\|\, \Gamma \to \Delta$, where Λ, Γ, Δ are multisets of atoms satisfying the following conditions.*

(i) Every atom in Λ is a LIA constraint of the form $s \triangleleft t$ or $x \triangleleft t$ or $x \trianglelefteq y$ where s, t are ground, $\triangleleft \in \{<, \leq, =, \neq, \geq, >\}$, and $\trianglelefteq \in \{\leq, =, \geq\}$,
(ii) Every atom in Γ and Δ is either an equation $s \approx s'$ with $s, s' \in \Omega \cup V_{\mathcal{S}}$, or a non-equational atom $P(s_1, \ldots, s_m)$, where every s_i of sort $\mathcal{Z}$ must be a variable $x \in V_{\mathcal{Z}}$, and every s_i of sort $\mathcal{S}$ may be a variable $u \in V_{\mathcal{S}}$ or a constant symbol $c \in \Omega$.

We omit the empty multiset left of "$\to$" and denote it by $\Box$ right of "$\to$" (where $\Box$ at the same time stands for *falsity*). The clause notation separates arithmetic constraints from the *free* (also: *uninterpreted*) part. We use the vertical double bar "$\|$" to indicate this separation syntactically. Intuitively, clauses $\Lambda \,\|\, \Gamma \to \Delta$ can be read as $\left(\bigwedge \Lambda \wedge \bigwedge \Gamma\right) \to \bigvee \Delta$, i.e. the multisets Λ, Γ stand for conjunctions of atoms and Δ stands for a disjunction of atoms.

Requiring the free part $\Gamma \to \Delta$ of clauses to not contain any base-sort terms apart from variables does not limit expressiveness. Every base-sort term $t \notin V_{\mathcal{Z}}$ in the free part can safely be replaced by a fresh base-sort variable x_t when an atomic constraint $x_t = t$ is added to the constraint part of the clause (a process known as *purification* or *abstraction* [4, 18]).

A *hierarchic interpretation* is an algebra $\mathcal{A}$ which interprets the base sort $\mathcal{Z}$ as $\mathcal{Z}^{\mathcal{A}} = \mathbb{Z}$, assigns integer values to all occurring base-sort Skolem constants, and interprets all LIA terms and constraints in the standard way. Moreover, $\mathcal{A}$ comprises a nonempty domain $\mathcal{S}^{\mathcal{A}}$, assigns to each free-sort constant symbol c in Ω a domain element $c^{\mathcal{A}} \in \mathcal{S}^{\mathcal{A}}$, and interprets every sorted predicate symbol $P : \xi_1 \times \ldots \times \xi_m$ in Π by a set $P^{\mathcal{A}} \subseteq \xi_1^{\mathcal{A}} \times \ldots \times \xi_m^{\mathcal{A}}$, as usual.

Given a hierarchic interpretation $\mathcal{A}$ and a sort-respecting *variable assignment* $\beta : V_{\mathcal{Z}} \cup V_{\mathcal{S}} \to \mathcal{Z}^{\mathcal{A}} \cup \mathcal{S}^{\mathcal{A}}$, we write $\mathcal{A}(\beta)(s)$ to address the *value of the term s under $\mathcal{A}$ with respect to the variable assignment β*. The variables occurring in clauses are implicitly universally quantified. Therefore, given a clause C, we call $\mathcal{A}$ a *hierarchic model of C*, denoted $\mathcal{A} \models C$, if and only if $\mathcal{A}, \beta \models C$ holds for every variable assignment β. For clause sets N, $\mathcal{A} \models N$ holds if and only if $\mathcal{A} \models C$ holds true for every clause $C \in N$. We call a clause C (a clause set N) *satisfiable* if and only if there exists a hierarchic model $\mathcal{A}$ of C (of N). Two clauses C, D (clause sets N, M) are *equisatisfiable* if and only if C (N) is satisfiable whenever D (M) is satisfiable and vice versa.

Given a BSR(SLI) clause C, $\text{consts}(C)$ denotes the set of all constant symbols occurring in C. The set $\text{bconsts}(N)$ ($\text{fconsts}(N)$) is the restriction of $\text{consts}(N)$ to base-sort (free-sort) constant symbols. By $\text{vars}(C)$ we denote the set of all variables occurring in C. Similar notation is used for other syntactic objects.

We define *substitutions* σ in the standard way as sort-respecting mappings from variables to terms. The *restriction of the domain of a substitution σ to a set V of variables* is denoted by $\sigma|_V$ and is defined such that $v\sigma|_V := v\sigma$ for every $v \in V$ and $v\sigma|_V = v$ for every $v \notin V$. While the application of a substitution σ to terms, atoms and multisets thereof is defined as usual, we need to be more specific for clauses. Consider a BSR(SLI) clause $C := \Lambda \,\|\, \Gamma \to \Delta$ and let $x_1, \ldots, x_k$ denote all base-sort variables occurring in C for which $x_i\sigma \neq x_i$. We then set $C\sigma := \Lambda\sigma, x_1 = x_1\sigma, \ldots, x_k = x_k\sigma \,\|\, \Gamma\sigma|_{V_{\mathcal{S}}} \to \Delta\sigma|_{V_{\mathcal{S}}}$.

A term, atom, etc. is called *ground*, if it does not contain any variables. A BSR(SLI) clause C is called *essentially ground* if it does not contain free-sort variables and for every base-sort variable x occurring in C there is a constraint $x = t$ in C for some ground LIA term t. A clause set N is *essentially ground* if all the clauses it contains are essentially ground.

Definition 2 (Normal Form of BSR(SLI) Clauses). *A BSR(SLI) clause $\Lambda \,\|\, \Gamma \to \Delta$ is in* normal form *if*

(1) all non-ground atoms in Λ have the form $x \trianglelefteq c$ or $x \leq y$ (or their symmetric variants) where c is an integer or Skolem constant and $\trianglelefteq \in \{\leq, =, \geq\}$,
(2) all base-sort variables that occur in Λ also occur in $\Gamma \to \Delta$, and
(3) Γ does not contain any equation of the form $u \approx t$.

A BSR(SLI) clause set N is in normal form *if all clauses in N are in normal form and pairwise variable disjoint. Moreover, we assume that N contains at least one free-sort constant symbol.*

For every BSR(SLI) clause set N there is an equisatisfiable BSR(SLI) clause set N' in normal form. It can be constructed from N by straightforward *purification/abstraction* methods [4, 18] and a simple procedure for eliminating existentially quantified variables in LIA constraints (see [14] for details).

3 Instantiation for BSR(SLI)

In this section, we present and prove our main technical result:

Theorem 3. *Satisfiability of a finite BSR(SLI) clause set N is decidable.*

In essence, one can show that N is equisatisfiable to a finite set of essentially ground clauses (cf. Lemma 12). There are calculi, such as hierarchic superposition [4, 3, 18, 11, 5] or DPLL(T) [21], that can decide satisfiability of ground clause sets. Our decidability result for BSR(SLI) does not come as a surprise, given the similarity to other logic fragments that are known to be decidable, such as the *array property fragment* by Bradley, Manna, and Sipma [8, 7] and Ge and de Moura's *finite essentially uninterpreted fragment* extended with simple integer arithmetic constraints [12].

More important than the obtained decidability result is the instantiation methodology that we employ, in particular for integer-sort variables. Typically, decision procedures for the integer-indexed array property fragment are based on an exhaustive instantiation of universally quantified array indices with all the ground index terms that occur in the formula at hand (cf. the original approach [8, 6] and standard literature [7, 17]). In more sophisticated approaches, only a *relevant portion* of the occurring arithmetic terms is singled out before instantiation [12].

Our methodology will also be based on a concept of relevant terms, determined by connections between the arguments of predicate symbols and instantiation points that are propagated along these connections. This part of our method is not specific for the integers but can be applied to the free part of our language as well. For integer variables, we investigate additional criteria to filter out unnecessary instances, inspired by the Loos–Weispfenning quantifier elimination procedure [20]. We elaborate on this in Sections 3.1 – 3.3.

3.1 Instantiation of Integer Variables

We first summarize the overall approach for the instantiation of integer variables in an intuitive way. To keep the informal exposition simple, we pretend that all LIA terms are constants from $\mathbb{Z}$. We even occasionally refer to the improper values $-\infty$ / $+\infty$ —"sufficiently small/large" integers. A formal treatment with proper definitions will follow.

Given a finite BSR(SLI) clause set N in normal form, we intend to partition $\mathbb{Z}$ into a set $\mathcal{P}$ of finitely many subsets $p \in \mathcal{P}$ such that satisfiability of N necessarily leads to the existence of a *uniform* hierarchic model.

Definition 4 (Uniform Interpretations). *A hierarchic interpretation $\mathcal{A}$ is* uniform *with respect to a partition $\mathcal{P}$ of the integers if and only if for every free predicate symbol Q occurring in N, every part $p \in \mathcal{P}$, and all integers $r_1, r_2 \in p$ we have $\langle \ldots, r_1, \ldots \rangle \in Q^{\mathcal{A}}$ if and only if $\langle \ldots, r_2, \ldots \rangle \in Q^{\mathcal{A}}$.*

As soon as we have found such a finite partition $\mathcal{P}$, we pick one integer value $r_p \in p$ as *representative* from each and every part $p \in \mathcal{P}$. Given a clause C that contains a base-sort variable x, and given constant symbols $d_1, \ldots, d_k$ whose values cover all these representatives, i.e. $\{d_1^{\mathcal{A}}, \ldots, d_k^{\mathcal{A}}\} = \{r_p \mid p \in \mathcal{P}\}$, we observe

$$\mathcal{A} \models C \text{ if and only if } \mathcal{A} \models \big\{C[x/d_i] \bigm| 1 \leq i \leq k\big\} \ .$$

This equivalence claims that we can transform universal quantification over the integer domain into finite conjunction over all representatives of subsets in $\mathcal{P}$. Formulated differently, we can extrapolate a model for a universally quantified clause set, if we can find a model of finitely many instances of this clause set. The formal version of this statement is given in Lemma 12. Uniform hierarchic models play a key role in its proof.

When we extract the partition $\mathcal{P}$ from the given clause set N, we exploit three aspects to increase efficiency:

(E-i) We group argument positions of free predicate symbols in such a way that the instantiation points relevant for these argument positions are identical. This means the variables that are associated to these argument positions, e.g. because they occur in such a place in some clause, need to be instantiated only with terms that are relevant for the respective group of argument positions. This is illustrated in Example 5.

(E-ii) Concerning the *relevant* integer constraints, i.e. the ones that produce instantiation points, one can choose to either stick to lower bounds exclusively, use $-\infty$ as a default (the lowest possible lower bound), and ignore upper bounds. Alternatively, one can focus on upper bounds, use $+\infty$ as default, and ignore lower bounds. This idea goes back to the Loos–Weispfenning quantifier elimination procedure over the reals [20]. Example 8 gives some intuition.

(E-iii) The choice described under (E-ii) can be made independently for every integer variable that is to be instantiated. See Examples 8 and 13.

Example 5. Consider the following clauses:

$$\begin{array}{lrll} C_1 := & 1 \leq x_1, x_2 \leq 0 \parallel & & \to T(x_1),\ Q(x_1, x_2)\ , \\ C_2 := & y_3 \leq 7,\ y_1 \leq y_3 \parallel & Q(y_1, y_2) & \to R(y_3)\ , \\ C_3 := & 6 \leq z_1 \parallel & T(z_1) & \to \Box\ . \end{array}$$

The variables x_1, x_2, y_1, y_2, y_3, and z_1 are affected by the constraints in which they occur explicitly. Technically, it is more suitable to speak of the *argument position* $\langle T, 1\rangle$ instead of variables x_1 and z_1 that occur as the first argument of the predicate symbol T in C_1 and C_3, respectively. Speaking in such terms, argument position $\langle T, 1\rangle$ is directly affected by the constraints $1 \leq x_1$ and $6 \leq z_1$, argument position $\langle Q, 1\rangle$ is directly affected by $1 \leq x_1$ and $y_1 \leq y_3$, $\langle Q, 2\rangle$ is affected by $x_2 \leq 0$, and, finally, $\langle R, 1\rangle$ is affected by $y_3 \leq 7$ and $y_1 \leq y_3$. Besides such direct effects, there are also indirect effects that have to be taken into account. For example, the argument position $\langle Q, 1\rangle$ is indirectly affected by the constraint $6 \leq z_1$, because C_1 establishes a connection between argument positions $\langle T, 1\rangle$ and $\langle Q, 1\rangle$ via the simultaneous occurrence of x_1 in both argument positions and $\langle T, 1\rangle$ is affected by $6 \leq z_1$. This is witnessed by the fact that C_1 and C_3 together logically entail the clause $D := 6 \leq x, y \leq 0 \parallel\ \to Q(x, y)$. D can be obtained by a hierarchic superposition step from C_1 and C_3, for instance. Another entailed clause is $6 \leq z, z \leq 7 \parallel\ \to R(z)$, the (simplified) result of hierarchically resolving D with C_2. Hence, $\langle R, 1\rangle$ is affected by the constraints $6 \leq z$ and $z \leq 7$. Speaking in terms of argument positions, this effect can be described as propagation of the lower bound $6 \leq y_1$ from $\langle Q, 1\rangle$ to $\langle R, 1\rangle$ via the constraint $y_1 \leq y_3$ in C_2. $\square$

One lesson learned from the example is that argument positions can be connected by variable occurrences or constraints of the form $x \leq y$. Such links in a clause set N are expressed by the relation $\rightrightarrows_N$.

Definition 6 (Connections Between Argument Positions and Argument Position Closures). *Let N be a BSR(SLI) clause set in normal form. We define $\rightrightarrows_N$ to be the smallest preorder (i.e. a reflexive and transitive relation) over $\Pi \times \mathbb{N}$ such that $\langle Q, j\rangle \rightrightarrows_N \langle P, i\rangle$ whenever there is a clause $\Lambda \,\|\, \Gamma \to \Delta$ in N containing free atoms $Q(\ldots, u, \ldots)$ and $P(\ldots, v, \ldots)$ in which the variable u occurs at the j-th and the variable v occurs at the i-th argument position and*

(1) either $u = v$,
(2) or $u \neq v$, both are of sort $\mathcal{Z}$ and there are constraints $u = v$ or $u \leq v$ in Λ,
(3) or $u \neq v$, both are of sort $\mathcal{S}$ and there is an atom $u \approx v$ in Γ or in Δ.[3]

$\rightrightarrows_N$ induces downward closed sets $\Downarrow_N\langle P, i\rangle$ of argument positions, called argument position closures*: $\Downarrow_N\langle P, i\rangle := \big\{\langle Q, j\rangle \bigm| \langle Q, j\rangle \rightrightarrows_N \langle P, i\rangle\big\}$.*

Consider a variable v that occurs at the i-th argument position of a free atom $P(\ldots, v, \ldots)$ in N. We denote the argument position closure related to v's argument position in N by $\Downarrow_N(v)$, i.e. $\Downarrow_N(v) := \Downarrow_N\langle P, i\rangle$. If v is a free-sort variable that exclusively occurs in equations, we set $\Downarrow_N(v) := \Downarrow\langle False_v, 1\rangle$ (cf. footnote [3]). To simplify notation a bit, we write $\rightrightarrows$, $\Downarrow\langle P, i\rangle$, and $\Downarrow(v)$ instead of $\rightrightarrows_N$, $\Downarrow_N\langle P, i\rangle$, and $\Downarrow_N(v)$, when the set N is clear from the context.

Notice that $\rightrightarrows$ confined to argument position pairs of the free sort is always symmetric. Asymmetry is only introduced by atomic constraints $x \leq y$.

While the relation $\rightrightarrows$ indicates how instantiation points are propagated between argument positions, the set $\Downarrow\langle P, i\rangle$ comprises all argument positions from which instantiation points are propagated to $\langle P, i\rangle$. For a variable v the set $\Downarrow(v)$ contains all argument positions that may produce instantiation points for v.

Next, we collect the instantiation points that are necessary to eliminate base-sort variables by means of finite instantiation.

Definition 7 (Instantiation Points for Base-Sort Argument Positions). *Let N be a BSR(SLI) clause set in normal form and let $P : \xi_1 \times \ldots \times \xi_m$ be a free predicate symbol occurring in N. For every i with $\xi_i = \mathcal{Z}$ we define $\mathcal{I}_{P,i}$ to be the smallest set satisfying the following condition. We have $d \in \mathcal{I}_{P,i}$ for any constant symbol d for which there exists a clause C in N that contains an atom $P(\ldots, x, \ldots)$ in which x occurs as the i-th argument and that contains a constraint $x = d$ or $x \geq d$.*

The most apparent peculiarity about this definition is that LIA constraints

[3] For any free-sort variable v that occurs in a clause $(\Lambda \,\|\, \Gamma \to \Delta) \in N$ exclusively in equations, we pretend that Δ contains an atom $\mathrm{False}_v(v)$, for a fresh predicate symbol $\mathrm{False}_v : \mathcal{S}$. This is merely a technical assumption. Without it, we would have to treat such variables v as a separate case in all definitions. The atom $\mathrm{False}_v(v)$ is not added "physically" to any clause.

of the form $x \leq d$ are completely ignored when collecting instantiation points for x's argument position. This is one of the aspects that makes this definition interesting from the efficiency point of view, because the number of instances that we have to consider might decrease considerably in this way. The following example may help to develop an intuitive understanding.

Example 8. Consider two clauses $C := \ 3 \leq x,\, x \leq 5 \parallel \rightarrow T(x)$ and $D := \ x \leq 0 \parallel T(x) \rightarrow \Box$. Recall that we are looking for a finite partition $\mathcal{P}$ of $\mathbb{Z}$ such that we can construct a uniform hierarchic model $\mathcal{A}$ of $\{C, D\}$, i.e. for every subset $p \in \mathcal{P}$ and all integers $r_1, r_2 \in p$ we want $r_1 \in T^{\mathcal{A}}$ to hold if and only if $r_2 \in T^{\mathcal{A}}$. A natural candidate for $\mathcal{P}$ is $\{(-\infty, 0], [1, 2], [3, 5], [6, +\infty)\}$, which takes every LIA constraint in C and D into account. Correspondingly, we find the candidate model $\mathcal{A}$ with $T^{\mathcal{A}} = [3, 5]$. Obviously, $\mathcal{A}$ is uniform with respect to $\mathcal{P}$.

But there are other interesting possibilities, for instance, the more coarse-grained partition $\{(-\infty, 2], [3, +\infty)\}$ together with the predicate $T^{\mathcal{A}} = [3, +\infty)$. This latter candidate partition completely ignores the constraints $x \leq 0$ and $x \leq 5$ that constitute upper bounds on x and in this way induces a simpler partition. Dually, we could have concentrated on the upper bounds instead (completely ignoring the lower bounds). This would have led to the partition $\{(-\infty, 0], [1, 5], [6, +\infty)\}$ and the candidate predicate $T^{\mathcal{A}} = [1, 5]$ (or $T^{\mathcal{A}} = [1, +\infty)$). Both ways are possible, but the former yields a coarser partition and is thus more attractive, as it will cause fewer instances in the end. $\Box$

The example reveals quite some freedom in choosing an appropriate partition of the integers. A large number of parts directly corresponds to a large number of instantiation points—one for each interval—, and therefore leads to a large number of instances that need to be considered by a reasoning procedure. Hence, regarding efficiency, it is of great importance to keep the partition $\mathcal{P}$ of $\mathbb{Z}$ coarse.

It remains to address the question of why it is sufficient to consider lower bounds only. At this point, we content ourselves with an informal explanation. Let $\varphi(x)$ be a satisfiable $\wedge$-$\vee$-combination of upper and lower bounds on some integer variable x. For the sake of simplicity, we assume that every atom in φ is of the form $c \leq x$ or $x \leq c$ with $c \in \mathbb{Z}$. When we look for some value of x that satisfies φ, we start from some "sufficiently small value" $-\infty$. If $-\infty$ yields a solution for φ, we are done. If $[x {\mapsto} -\infty] \not\models \varphi$, there must be some lower bound in φ that prevents $-\infty$ from being a solution. In order to find a solution, we successively increase the value of x until a solution is found. Interesting test points $r \in \mathbb{Z}$ for x are those where $r - 1$ violates some lower bound $c \leq x$ in φ and r satisfies the bound, i.e. $r = c$. Consider two lower bounds $c_1 \leq x$ and $c_2 \leq x$ in φ such that $c_1 < c_2$ and φ contains no further bound $d \leq x$ with $c_1 < d < c_2$. Any assignment $[x {\mapsto} r]$ with $c_1 < r < c_2$ satisfies exactly the same lower bounds as the assignment $[x {\mapsto} c_1]$ does. Moreover, any such assignment satisfies *at most* the upper bounds that $[x {\mapsto} c_1]$ satisfies. In fact, it may violate some of them. Consequently, if neither $[x {\mapsto} c_1]$ nor $[x {\mapsto} c_2]$ satisfy φ, then $[x {\mapsto} r]$ with $c_1 < r < c_2$ cannot satisfy φ either. In other words, it suffices to test only values induced by lower bounds. The abstract value $-\infty$ serves as the default value, which corresponds to the implicit lower bound $-\infty < x$.

Definition 9 (Instantiation Points for Base-Sort Argument Position Closures and Induced Partition). *Let N be a BSR(SLI) clause set in normal form and let $\mathcal{A}$ be a hierarchic interpretation. For every base-sort argument position closure $\Downarrow\langle P, i\rangle$ induced by $\rightrightarrows$ we define the following:*

The set $\mathcal{I}_{\Downarrow\langle P,i\rangle}$ of instantiation points *for $\Downarrow\langle P, i\rangle$ is defined by $\mathcal{I}_{\Downarrow\langle P,i\rangle} := \{c_{-\infty}\} \cup \bigcup_{\langle Q,j\rangle \in \Downarrow\langle P,i\rangle} \mathcal{I}_{Q,j}$, where we assume $c_{-\infty}$ to be a distinguished base-sort constant symbol that may occur in N.*

Let the sequence $r_1, \ldots, r_k$ comprise all integers in the set $\big\{c^{\mathcal{A}} \bigm| c \in \mathcal{I}_{\Downarrow\langle P,i\rangle} \setminus \{c_{-\infty}\}\big\}$ ordered so that $r_1 < \ldots < r_k$. The partition $\mathcal{P}^{\mathcal{A}}_{\Downarrow\langle P,i\rangle}$ of the integers into finitely many intervals is defined by

$$\mathcal{P}^{\mathcal{A}}_{\Downarrow\langle P,i\rangle} := \big\{(-\infty, r_1 - 1], [r_1, r_2 - 1], \ldots, [r_{k-1}, r_k - 1], [r_k, +\infty)\big\}.$$

Please note that partitions as described in the definition do always exist, and do not contain empty parts.

Lemma 10. *Let N be a BSR(SLI) clause set in normal form and let $\mathcal{A}$ be a hierarchic interpretation. Consider two argument position pairs $\langle Q, j\rangle, \langle P, i\rangle$ for which $\langle Q, j\rangle \rightrightarrows \langle P, i\rangle$ holds in N. Then $\mathcal{I}_{\Downarrow\langle Q,j\rangle} \subseteq \mathcal{I}_{\Downarrow\langle P,i\rangle}$. Moreover, $\mathcal{P}^{\mathcal{A}}_{\Downarrow\langle P,i\rangle}$ is a refinement of $\mathcal{P}^{\mathcal{A}}_{\Downarrow\langle Q,j\rangle}$, i.e. for every $p \in \mathcal{P}^{\mathcal{A}}_{\Downarrow\langle P,i\rangle}$ there is some $p' \in \mathcal{P}^{\mathcal{A}}_{\Downarrow\langle Q,j\rangle}$ such that $p \subseteq p'$.*

Lemma 11. *Let N be a BSR(SLI) clause set in normal form and let $\mathcal{A}$ be a hierarchic interpretation. For every part $p \in \mathcal{P}^{\mathcal{A}}_{\Downarrow\langle P,i\rangle}$ of the form $p = [r_\ell, r_u]$ or $p = [r_\ell, +\infty)$ we find some constant symbol $c_{\Downarrow\langle P,i\rangle,p} \in \mathcal{I}_{\Downarrow\langle P,i\rangle}$ with $c^{\mathcal{A}}_{\Downarrow\langle P,i\rangle,p} = r_\ell$.*

Note that the lemma did not say anything about the part $(-\infty, r_u]$ which also belongs to every $\mathcal{P}^{\mathcal{A}}_{\Downarrow\langle P,i\rangle}$. Our intention is that the constant symbol $c_{-\infty}$ shall be interpreted by a value from this interval. Hence, we add the set of clauses $\Psi^{-\infty}_N := \big\{(c_{-\infty} \geq c \,\|\, \to \Box) \bigm| c \in \text{bconsts}(N) \setminus \{c_{-\infty}\}\big\}$ whenever necessary. Note that if $\mathcal{A}$ is a hierarchic model of a given BSR(SLI) clause set N, then $\mathcal{A}$ can be turned into a model of $\Psi^{-\infty}_N$ just by changing the interpretation of $c_{-\infty}$. After this modification $\mathcal{A}$ is still a model of N, if $c_{-\infty}$ does not occur in N.

The next lemma shows that we can eliminate base-sort variables x from clauses C in a finite BSR(SLI) clause set N by replacing C with finitely many instances in which x is substituted with the instantiation points that we computed for x. In addition, the axioms that stipulate the meaning of $c_{-\infty}$ need to be added. Iterating this instantiation step for every base-sort variable in N eventually leads to a clause set that is essentially ground with respect to the constraint parts of the clauses it contains (free-sort variables need to be treated separately, of course, see Section 3.3).

Lemma 12 (Finite Integer-Variable Elimination). *Let N be a finite BSR(SLI) clause set in normal form such that, if the constant symbol $c_{-\infty}$ occurs in N, then $\Psi^{-\infty}_N \subseteq N$. Suppose there is a clause C in N which contains a base-sort variable x. Let $\widehat{N}_x$ be the clause set $\widehat{N}_x := \big(N \setminus \{C\}\big) \cup \big\{C[x/c] \bigm| c \in \mathcal{I}_{\Downarrow_N(x)}\big\} \cup \Psi^{-\infty}_N$. N is satisfiable if and only if $\widehat{N}_x$ is satisfiable.*

Proof sketch. The "only if"-part is trivial.

The "if"-part requires a more sophisticated argument. In what follows, the notations $\rightrightarrows$ and $\Downarrow$ always refer to the original clause set N. Let $\mathcal{A}$ be a hierarchic model of $\widehat{N}_x$. We use $\mathcal{A}$ to construct the hierarchic model $\mathcal{B} \models N$ as follows. For the domain $\mathcal{S}^{\mathcal{B}}$ we reuse $\mathcal{A}$'s free domain $\mathcal{S}^{\mathcal{A}}$. For every base-sort or free-sort constant symbol $c \in \mathrm{consts}(N)$ we set $c^{\mathcal{B}} := c^{\mathcal{A}}$. For every predicate symbol $P : \xi_1 \times \ldots \times \xi_m$ that occurs in N, for every argument position i, $1 \leq i \leq m$, with $\xi_i = \mathcal{Z}$, and for every interval $p \in \mathcal{P}^{\mathcal{A}}_{\Downarrow\langle P,i\rangle}$ Lemma 11 and the extra clauses in $\Psi_N^{-\infty}$ guarantee the existence of a base-sort constant symbol $c_{\Downarrow\langle P,i\rangle,p} \in \mathcal{I}_{\Downarrow(x)}$, such that $c^{\mathcal{A}}_{\Downarrow\langle P,i\rangle,p} \in p$. Based on this observation, we define the family of projection functions $\pi_{\Downarrow\langle P,i\rangle} : \mathbb{Z} \cup \mathcal{S}^{\mathcal{B}} \to \mathbb{Z} \cup \mathcal{S}^{\mathcal{A}}$ by

$$\pi_{\Downarrow\langle P,i\rangle}(\mathfrak{a}) := \begin{cases} c^{\mathcal{A}}_{\Downarrow\langle P,i\rangle,p} & \text{if } \xi_i = \mathcal{Z} \text{ and } p \in \mathcal{P}^{\mathcal{A}}_{\Downarrow\langle P,i\rangle} \\ & \text{is the interval } \mathfrak{a} \text{ lies in,} \\ \mathfrak{a} & \text{if } \xi_i = \mathcal{S}. \end{cases}$$

Using the projection functions $\pi_{\Downarrow\langle P,i\rangle}$, we define the sets $P^{\mathcal{B}}$ in such a way that for all domain elements $\mathfrak{a}_1, \ldots, \mathfrak{a}_m$ of appropriate sorts

$\big\langle \mathfrak{a}_1, \ldots, \mathfrak{a}_m \big\rangle \in P^{\mathcal{B}}$ if and only if $\big\langle \pi_{\Downarrow\langle P,1\rangle}(\mathfrak{a}_1), \ldots, \pi_{\Downarrow\langle P,m\rangle}(\mathfrak{a}_m) \big\rangle \in P^{\mathcal{A}}$.

We next show $\mathcal{B} \models N$. Consider any clause $C' := \Lambda' \,\|\, \Gamma' \to \Delta'$ in N and let $\beta : V_{\mathcal{Z}} \cup V_{\mathcal{S}} \to \mathbb{Z} \cup \mathcal{S}^{\mathcal{B}}$ be some variable assignment. From β we derive a special variable assignment β_π for which we shall infer $\mathcal{A}, \beta_\pi \models C'$ as an intermediate step: $\beta_\pi(v) := \pi_{\Downarrow(v)}(\beta(v))$ for every variable v. If $C' \neq C$, then $\widehat{N}_x$ already contains C', and thus $\mathcal{A}, \beta_\pi \models C'$ must hold. In case of $C' = C$, let p_* be the interval in $\mathcal{P}^{\mathcal{A}}_{\Downarrow(x)}$ containing the value $\beta(x)$, and let c_* be an abbreviation for $c_{\Downarrow(x),p_*}$. Due to $\beta_\pi(x) = c_*^{\mathcal{A}}$ and since $\mathcal{A}$ is a model of the clause $C\big[x/c_*\big]$ in $\widehat{N}_x$, we conclude $\mathcal{A}, \beta_\pi \models C$. Hence, in any case we can deduce $\mathcal{A}, \beta_\pi \models C'$. By case distinction on why $\mathcal{A}, \beta_\pi \models C'$ holds, we may use this result to infer $\mathcal{B}, \beta \models C'$. It follows that $\mathcal{B} \models N$. □

3.2 Independent Bound Selection

By now we have mainly focused on lower bounds as sources for instantiation points. However, as we have already pointed out (cf. (E-ii) and (E-iii) in Section 3.1 and Example 8), there is also a dual approach in which upper bounds on integer variables play the central role. It turns out that the choice between the two approaches can be made independently for every variable that is to be instantiated. In the interest of efficiency, it makes sense to always choose the approach that results in fewer non-redundant instances or, more abstractly speaking, a set of instances whose satisfiability is easier to decide. Example 13 illustrates the overall approach.

Given a clause set N in normal form, the relation $\rightrightarrows_N$ is defined as before. Dually to the sets $\Downarrow_N\langle P,i\rangle$, we define the sets $\Uparrow_N\langle P,i\rangle := \big\{\langle Q,j\rangle \;\big|\; \langle P,i\rangle \rightrightarrows_N \langle Q,j\rangle\big\}$, which constitute *upwards closed* sets with respect to $\rightrightarrows_N$ rather than *downwards closed* sets. Regarding instantiation points, only LIA constraints $x = d$ and $x \leq d$ lead to $d \in \mathcal{I}_{\Uparrow_N(x)}$. In addition, $c_{+\infty}$ is by default added to every set $\mathcal{I}_{\Uparrow_N \langle P,i\rangle}$. In order to fix the meaning of $c_{+\infty}$, we introduce the set of axioms $\Psi_N^{+\infty} := \big\{(c_{+\infty} \leq c \,\|\, \rightarrow \Box) \;\big|\; c \in \mathrm{bconsts}(N) \setminus \{c_{+\infty}\}\big\}$. The dual versions of Definitions 7 and 9 and Lemma 12 can be found in [14].

In both, Lemma 12 and its dual version, the equisatisfiable instantiation can be applied to the respective variable independently of the instantiation steps that have already been done or are still to be done in the future. This means, we can choose independently, whether to stick to the lower or upper bounds for instantiation. This choice can, for example, be made depending on the number of non-redundant instances that have to be generated.

Example 13. Consider the following BSR(SLI) clause set N:

$$\begin{aligned}
1 \leq x_1, x_2 \leq 0 \;\|\; & \rightarrow T(x_1),\; Q(x_1,x_2)\;,\\
y_3 \leq 7,\; y_1 \leq y_3 \;\|\; Q(y_1,y_2) & \rightarrow R(y_3)\;,\\
6 \leq z_1,\; z_1 \leq 9 \;\|\; T(z_1) & \rightarrow \Box\;.
\end{aligned}$$

We intend to instantiate the variables y_3, y_1, x_1, z_1 in this order. For y_3 we can choose between $\mathcal{I}_{\Downarrow_N (y_3)} = \{c_{-\infty}, 1, 6\}$ and $\mathcal{I}_{\Uparrow_N (y_3)} = \{7, c_{+\infty}\}$. Using the latter option, we obtain the instances

$$\begin{aligned}
7 \leq 7, y_1 \leq 7, y_3 = 7 \;\|\; Q(y_1,y_2) \rightarrow R(y_3)\\
c_{+\infty} \leq 7, y_1 \leq c_{+\infty}, y_3 = c_{+\infty} \;\|\; Q(y_1,y_2) \rightarrow R(y_3)
\end{aligned}$$

plus the clauses in $\Psi_N^{+\infty}$. The constraint $7 \leq 7$ can be removed, as it is redundant. The second instance can be dropped immediately, since the constraint $c_{+\infty} \leq 7$ is false in any model satisfying $\Psi_N^{+\infty}$. Dual simplifications can be applied to constraints with $c_{-\infty}$. Let N' contain the clauses in $\Psi_N^{+\infty}$ and the clauses

$$\begin{aligned}
1 \leq x_1, x_2 \leq 0 \;\|\; & \rightarrow T(x_1),\; Q(x_1,x_2)\;,\\
y_1 \leq 7, y_3 = 7 \;\|\; Q(y_1,y_2) & \rightarrow R(y_3)\;,\\
6 \leq z_1, z_1 \leq 9 \;\|\; T(z_1) & \rightarrow \Box\;.
\end{aligned}$$

For y_1 we use $\mathcal{I}_{\Downarrow_{N'} (y_1)} = \{c_{-\infty}, 1, 6\}$ rather than $\mathcal{I}_{\Uparrow_{N'} (y_1)} = \{7, 9, c_{+\infty}\}$ for instantiation and obtain N'' (after simplification):

$$\begin{aligned}
1 \leq x_1, x_2 \leq 0 \;\|\; & \rightarrow T(x_1),\; Q(x_1,x_2)\;,\\
y_3 = 7, y_1 = c_{-\infty} \;\|\; Q(y_1,y_2) & \rightarrow R(y_3)\;,\\
y_3 = 7, y_1 = 1 \;\|\; Q(y_1,y_2) & \rightarrow R(y_3)\;,\\
y_3 = 7, y_1 = 6 \;\|\; Q(y_1,y_2) & \rightarrow R(y_3)\;,\\
6 \leq z_1,\; z_1 \leq 9 \;\|\; T(z_1) & \rightarrow \Box\;,
\end{aligned}$$

plus the clauses in $\Psi_N^{-\infty}$ and $\Psi_N^{+\infty}$ and plus the clause $c_{-\infty} \geq c_{+\infty} \| \rightarrow \Box$. The sets of instantiation points for x_1 in N'' are $\mathcal{I}_{\Downarrow_{N''} (x_1)} = \{c_{-\infty}, 1, 6\}$ and $\mathcal{I}_{\Uparrow_{N''} (x_1)} = \{c_{-\infty}, 1, 6, 9, c_{+\infty}\}$. The latter set nicely illustrates how instantiation sets for particular variables can evolve during the incremental process of instantiation. We take the set with fewer instantiation points and obtain N''':

$$\begin{aligned}
x_2 \leq 0, x_1 = 1 \;\|\; & \rightarrow T(x_1),\; Q(x_1,x_2)\;,\\
x_2 \leq 0, x_1 = 6 \;\|\; & \rightarrow T(x_1),\; Q(x_1,x_2)\;,\\
y_3 = 7, y_1 = c_{-\infty} \;\|\; Q(y_1,y_2) & \rightarrow R(y_3)\;,\\
y_3 = 7, y_1 = 1 \;\|\; Q(y_1,y_2) & \rightarrow R(y_3)\;,\\
y_3 = 7, y_1 = 6 \;\|\; Q(y_1,y_2) & \rightarrow R(y_3)\;,\\
6 \leq z_1,\; z_1 \leq 9 \;\|\; T(z_1) & \rightarrow \Box\;,
\end{aligned}$$

plus $\Psi_N^{-\infty} \cup \Psi_N^{+\infty} \cup \{c_{-\infty} \geq c_{+\infty} \| \rightarrow \Box\}$. We instantiate z_1 using the set $\mathcal{I}_{\Downarrow_{N'''} (z_1)} = \{c_{-\infty}, 1, 6\}$ and not $\mathcal{I}_{\Uparrow_{N'''} (z_1)} = \{c_{-\infty}, 1, 6, 9, c_{+\infty}\}$:

$$\begin{aligned}
x_2 \leq 0, x_1 = 1 \;\|\; & \rightarrow T(x_1), Q(x_1,x_2)\;,\\
x_2 \leq 0, x_1 = 6 \;\|\; & \rightarrow T(x_1), Q(x_1,x_2)\;,
\end{aligned}$$

$$\begin{aligned} y_3 = 7, y_1 = c_{-\infty} \parallel Q(y_1, y_2) &\rightarrow R(y_3)\ , \\ y_3 = 7, y_1 = 1 \parallel Q(y_1, y_2) &\rightarrow R(y_3)\ , \\ y_3 = 7, y_1 = 6 \parallel Q(y_1, y_2) &\rightarrow R(y_3)\ , \\ z_1 = 6 \parallel T(z_1) \quad &\rightarrow \Box\ , \end{aligned}$$

plus $\Psi_N^{-\infty} \cup \Psi_N^{+\infty} \cup \{c_{-\infty} \geq c_{+\infty} \| \rightarrow \Box\}$. Until now, we have introduced 6 non-redundant instances. A completely naive instantiation approach where x_1, y_1, y_3, z_1 are instantiated with all occurring constant symbols $0, 1, 6, 7, 9$ leads to 17 non-redundant instances. This corresponds to the originally proposed method for the *array property fragment*, cf. [8]. A more sophisticated instantiation approach where x_1, y_1, y_3, z_1 are instantiated with $1, 6, 7, 9$ (as there is no connection from 0 to x_1, y_1, y_3, z_1) leads to 13 non-redundant instances. For instance, the methods described in [12] produce this set of instances. $\Box$

3.3 Instantiation of Free-Sort Variables

We can also follow an instantiation approach for free-sort variables. In a nutshell, we collect only *relevant* instantiation points for a given argument position (cf. (E-i)). A similar approach is taken in [12]. Consult [14] for details.

4 Stratified Clause Sets

In this section we treat certain clause sets with uninterpreted non-constant function symbols. By a transformation into an equisatisfiable set of BSR clauses, we show that our instantiation methods are also applicable in such settings.

Definition 14. *Let N be a finite set of variable-disjoint first-order clauses in which also non-constant function symbols occur. By Π_N and Ω_N we denote the set of occurring predicate symbols and function symbols (including constants), respectively. N is considered to be* stratified *if we can define a mapping $lvl_N : (\Pi_N \cup \Omega_N) \times \mathbb{N} \rightarrow \mathbb{N}$ that maps argument position pairs (of predicate and function symbols) to nonnegative integers such that the following conditions are satisfied.*

(a) For every function symbol $f : \xi_1 \times \ldots \times \xi_m \rightarrow \xi_{m+1}$ and every $i \leq m$ we have $lvl_N\langle f, i\rangle > lvl_N\langle f, m+1\rangle$.
(b) For every (sub)term $g(s_1, \ldots, s_{k-1}, f(t_1, \ldots, t_m), s_{k+1}, \ldots, s_{m'})$ occurring in N we have $lvl_N\langle f, m+1\rangle = lvl_N\langle g, k\rangle$. This includes the case where f is a constant symbol and $m = 0$. Moreover, this also includes the case where g is replaced with a predicate symbol P.
(c) For every variable v that occurs in two (sub)terms $f(s_1, \ldots, s_{k-1}, v, s_{k+1}, \ldots, s_m)$ and $g(t_1, \ldots, t_{k'-1}, v, t_{k'+1}, \ldots, t_{m'})$ in N we have $lvl_N\langle f, k\rangle = lvl_N\langle g, k'\rangle$. The same applies, if f or g or both are replaced with predicate symbols.
(d) For every equation $f(s_1, \ldots, s_m) \approx g(t_1, \ldots, t_{m'})$ we have $lvl_N\langle f, m+1\rangle = lvl_N\langle g, m'+1\rangle$. This includes the cases where f or g or both are constant symbols (with $m = 0$ or $m' = 0$ or both, respectively).

Several known logic fragments fall into this syntactic category: many-sorted clauses over *stratified vocabularies* as described in [1, 16], and clauses belonging to the *finite essentially uninterpreted fragment* (cf. Proposition 2 in [12]).

Lemma 15. *Let $C = \Gamma \to \Delta$ be a first-order clause and let $f_1, \ldots, f_n$ be a list of all uninterpreted non-constant function symbols occurring in C. Let $R_1, \ldots, R_n$ be distinct predicate symbols that do not occur in C and that have the sort $R_i : \xi_1 \times \ldots \times \xi_m \times \xi_{m+1}$, if and only if f_i has the sort $\xi_1 \times \ldots \times \xi_m \to \xi_{m+1}$. Let Φ_1 and Φ_2 be the following sets of sentences:*
$\Phi_1 := \big\{\forall x_1 \ldots x_m uv.\ R_i(x_1, \ldots, x_m, u) \wedge R_i(x_1, \ldots, x_m, v) \to u \approx v \;\big|\; 1 \leq i \leq n\big\}$ and $\Phi_2 := \big\{\forall x_1 \ldots x_m \exists v.\ R_i(x_1, \ldots, x_m, v) \;\big|\; 1 \leq i \leq n\big\}$. There is a clause D that does not contain non-constant function symbols and for which the set $\{D\} \cup \Phi_1 \cup \Phi_2$ is equisatisfiable to C.

Proof sketch. We apply the following flattening rules. v stands for a fresh variable that has not occurred yet. P ranges over predicate symbols different from $\approx$. $\bar{s}$ and $\bar{t}$ stand for tuples of arguments.

$$\frac{\Gamma, f_i(\bar{s}) \approx f_j(\bar{t}) \to \Delta}{\Gamma, R_i(\bar{s}, v), R_j(\bar{t}, v) \to \Delta}\,(\text{fun-fun left}) \qquad \frac{\Gamma \to \Delta, f_i(\bar{s}) \approx f_j(\bar{t})}{\Gamma, R_i(\bar{s}, v) \to \Delta, R_j(\bar{t}, v)}\,(\text{fun-fun right})$$

$$\frac{\Gamma, f_i(\bar{s}) \approx c \to \Delta}{\Gamma, R_i(\bar{s}, c) \to \Delta}\,(\text{fun-const left}) \qquad \frac{\Gamma \to \Delta, f_i(\bar{s}) \approx c}{\Gamma \to \Delta, R_i(\bar{s}, c)}\,(\text{fun-const right})$$

$$\frac{\Gamma, f_i(\bar{s}) \approx x \to \Delta}{\Gamma, R_i(\bar{s}, x) \to \Delta}\,(\text{fun-var left}) \qquad \frac{\Gamma \to \Delta, f_i(\bar{s}) \approx x}{\Gamma \to \Delta, R_i(\bar{s}, x)}\,(\text{fun-var right})$$

$$\frac{\Gamma, P(\ldots, f_i(\bar{s}), \ldots) \to \Delta}{\Gamma, R_i(\bar{s}, v), P(\ldots, v, \ldots) \to \Delta}\,(\text{fun left}) \quad \frac{\Gamma \to \Delta, P(\ldots, f_i(\bar{s}), \ldots)}{\Gamma, R_i(\bar{s}, v) \to \Delta, P(\ldots, v, \ldots)}\,(\text{fun right})$$

□

Given a BSR clause $\Gamma \to \Delta$, we consider an atom $R_j(\bar{t}, v)$ in Δ to be *guarded*, if there is also an atom $R_i(\bar{s}, v)$ in Γ. With the exception of the rule `(fun-var right)` the flattening rules presented in the proof of Lemma 15 preserve guardedness of atoms in Δ and introduce atoms $R_j(\bar{t}, v)$ on the right-hand side of a clause only if at the same time a corresponding guard is introduced on the left-hand side of the clause.

Hence, if we are given a stratified clause set in which the atoms $x \approx t$ in the consequents of implications are subject to certain restrictions (e.g. $t \neq f(\ldots)$ and guardedness of atoms $u \approx c$ and $u \approx v$), then the above flattening rules yield clauses that belong to the following class of BSR(SLI) clauses—after necessary purification and normalization steps. In the definition we mark certain predicate symbols that are intended to represent uninterpreted functions. By adding suitable axioms later on, these will be equipped with the properties of function graphs.

Definition 16 (Stratified and Guarded BSR(SLI)). *Consider a BSR(SLI) clause set N in normal form. Let $R_1, \ldots, R_n$ be a list of predicate symbols that we consider to be* marked *in N. We call N* stratified and guarded *with respect to $R_1, \ldots, R_n$, if and only if the following conditions are met.*

(a) There is some function $lvl_N : \Pi \times \mathbb{N} \to \mathbb{N}$ that assigns to each argument position pair $\langle P, i\rangle$ a nonnegative integer $lvl_N\langle P, i\rangle$ such that

(a.1) $\langle P, i\rangle \rightrightarrows_N \langle Q, j\rangle$ *entails* $lvl_N\langle P, i\rangle = lvl_N\langle Q, j\rangle$*, and*

(a.2) for every marked predicate symbol $R_j : \xi_1 \times \ldots \times \xi_m \times \xi_{m+1}$ *we have* $lvl_N\langle R_j, i\rangle > lvl_N\langle R_j, m+1\rangle$ *for every* $i \leq m$*.*

(b) In every clause $\Lambda \,\|\, \Gamma \to \Delta$ *in* N *any occurrence of an atom* $R_j(s_1, \ldots, s_m, v)$ *in* Δ *entails that* Γ *contains some atom* $R_\ell(t_1, \ldots, t_{m'}, v)$*.*

(c) For every atom $u \approx t$ *in* N*, where* t *is either a free-sort variable* v *or a free-sort constant symbol, at least one of two cases applies:*

(c.1) $u \approx t$*, which must occur in the consequent of a clause, is guarded by some atom* $R_j(t_1, \ldots, t_m, u)$ *occurring in the antecedent of the same clause.*

(c.2) For every marked predicate symbol $R_j : \xi_1 \times \ldots \times \xi_m \times \xi_{m+1}$ *and every argument position closure* $\Downarrow_N\langle R_j, i\rangle$ *with* $1 \leq i \leq m$ *we have* $\Downarrow_N\langle R_j, i\rangle \cap \Downarrow_N(u) = \emptyset$*. If* $t = v$*, we in addition have* $\Downarrow_N\langle R_j, i\rangle \cap \Downarrow_N(v) = \emptyset$*.*

Notice that any atom $u \approx v$ over distinct variables requires two guards $R(\bar{s}, u)$ and $R(\bar{t}, v)$ in order to be guarded in accordance with Condition (c.1).

Let N be a finite BSR(SLI) clause set in normal form that is stratified and guarded with respect to $R_1, \ldots, R_n$. Let $R_i : \xi_1 \times \ldots \times \xi_m \times \xi_{m+1}$ be marked in N and let $P : \zeta_1 \times \ldots \times \zeta_{m'}$ be any predicate symbol occurring in N (be it marked or not). We write $R_i \succeq P$ if and only if $\mathrm{lvl}_N\langle R_i, m+1\rangle \geq \min_{1\leq \ell \leq m'}\big(\mathrm{lvl}_N\langle P, \ell\rangle\big)$. Without loss of generality, we assume $R_1 \succeq_N \ldots \succeq_N R_n$. Let $\Phi_1 := \{\forall x_1 \ldots x_m u u'.(R_i(x_1, \ldots, x_m, u) \wedge R_i(x_1, \ldots, x_m, u')) \to u \simeq u' \mid R_i \text{ has arity } m+1\}$ and $\Phi_2 := \{\forall x_1 \ldots x_m \exists u.\, R_i(x_1, \ldots, x_m, u) \mid R_i \text{ has arity } m+1\}$, where "$\simeq$" is a placeholder for "$\approx$" in free-sort equations and for "$=$" in base-sort equations.

Given a set M of BSR(SLI) clauses and an $(m+1)$-ary predicate symbol R that is marked in M, we define the set $\Phi(R, M) :=$

$$\big\{R(c_1, \ldots, c_m, d_{Rc_1\ldots c_m}) \;\big|\; \langle c_1, \ldots, c_m\rangle \in \mathcal{I}^{[m]}_{\Downarrow_M\langle R,\cdot\rangle}\big\}$$
$$\cup\big\{\forall x_1 \ldots x_m.\, \textstyle\bigvee_{\langle c_1,\ldots,c_m\rangle \in \mathcal{I}^{[m]}_{\Downarrow_M\langle R,\cdot\rangle}} R(x_1, \ldots, x_m, d_{Rc_1\ldots c_m})\big\}$$
$$\cup\big\{\forall x_1 \ldots x_m u.\, R(x_1, \ldots, x_m, u) \to \textstyle\bigvee_{\langle c_1,\ldots,c_m\rangle \in \mathcal{I}^{[m]}_{\Downarrow_M\langle R,\cdot\rangle}} u \simeq d_{Rc_1\ldots c_m}\big\}$$
$$\cup\big\{\forall x_1 \ldots x_m.\, R(x_1, \ldots, x_m, d_{Rc_1\ldots c_m}), R(x_1, \ldots, x_m, d_{Rc'_1\ldots c'_m}) \to d_{Rc_1\ldots c_m} \simeq d_{Rc'_1\ldots c'_m} \;\big|\; \langle c_1, \ldots, c_m\rangle, \langle c'_1, \ldots, c'_m\rangle \in \mathcal{I}^{[m]}_{\Downarrow_M\langle R,\cdot\rangle}\big\}$$

where $\mathcal{I}^{[m]}_{\Downarrow_M\langle R,\cdot\rangle}$ is used as an abbreviation for $\mathcal{I}_{\Downarrow_M\langle R,1\rangle} \times \ldots \times \mathcal{I}_{\Downarrow_M\langle R,m\rangle}$ and the $d_{Rc_1\ldots c_m}$ are assumed to be fresh constant symbols. It is worth noticing that the clauses corresponding to $\Phi(R, M)$ are stratified and guarded BSR(SLI) clauses.

We construct the sequence $M_0, M_1, \ldots, M_n$ of finite clause sets as follows: $M_0 := N$, every $M_{\ell+1}$ with $\ell \geq 0$ is an extension of M_ℓ by the BSR(SLI) clauses that correspond to the sentences in $\Phi(R_{\ell+1}, M_\ell)$.

Lemma 17. *The set* $N \cup \Phi_1 \cup \Phi_2$ *is satisfiable if and only if* M_n *is satisfiable.*

This lemma entails that all the instantiation methods developed in Section 3 can be used to decide satisfiability of stratified and guarded BSR(SLI) clause sets.

We can add another background theory to the stratified and guarded fragment of BSR(SLI) while preserving compatibility with our instantiation approach. Let

$\Pi_{\mathcal{T}}$ and $\Omega_{\mathcal{T}}$ be finite sets of sorted predicate symbols and sorted function symbols, respectively, and let $\mathcal{T}$ be some theory over $\Pi_{\mathcal{T}}$ and $\Omega_{\mathcal{T}}$. We assume that $\Pi_{\mathcal{T}}$ is disjoint from the set Π of uninterpreted predicate symbols. For any set X of variables, let $\mathbb{T}_{\mathcal{T}}(X)$ be the set of all well-sorted terms constructed from the variables in X and the function and constant symbols in $\Omega_{\mathcal{T}}$.

Definition 18 (BSR(SLI+$\mathcal{T}$)). *A clause set N belongs to* BSR(SLI+$\mathcal{T}$) *if it complies with the syntax of a BSR(SLI) clause set that is stratified and guarded with respect to certain predicate symbols $R_1, \ldots, R_n$ with the following exceptions. Let $C := \Lambda \,\|\, \Gamma \to \Delta$ be a clause in N. We allow atoms $P(s_1, \ldots, s_m)$ with $P \in \Pi_{\mathcal{T}}$ and $s_1, \ldots, s_m \in \mathbb{T}_{\mathcal{T}}(V_{\mathcal{Z}} \cup V_{\mathcal{S}})$—including equations $s_1 \approx s_2$—, if for every variable u occurring in any of the s_i there is either a LIA guard of the form $u = t$ in Λ with t being ground, or there is a guard $R_j(t_1, \ldots, t_{m'}, u)$ in Γ.*

The instantiation methods presented in Section 3 are also applicable to BSR(SLI+$\mathcal{T}$), since Lemma 17 can be extended to cover finite BSR(SLI+$\mathcal{T}$) clause sets. When computing instantiation points for BSR(SLI+$\mathcal{T}$) clause sets, we ignore $\mathcal{T}$-atoms. For example, a clause $\| R(t,u), P(s,c) \to P(s',u), Q(u)$ where $P(s,c)$ and $P(s',u)$ are $\mathcal{T}$-atoms, *does not* lead to an instantiation point c for $\Downarrow\langle Q, 1\rangle$. If we stick to this approach, the proof of Lemma 17 can easily be adapted to handle additional $\mathcal{T}$-atoms. The involved model construction remains unchanged. $\mathcal{T}$-atoms are basically treated like guarded free-sort atoms $u \approx d$.

Proposition 19. *BSR(SLI+$\mathcal{T}$) allows an (un)satisfiability-preserving embedding of the* array property fragment *with integer-indexed arrays and element theory $\mathcal{T}$ (cf. [8]) and of the* finite essentially uninterpreted fragment *extended with simple integer arithmetic literals (cf. [12]) into BSR(SLI+$\mathcal{T}$).*

Example 20. The following formula φ belongs to the array property fragment with integer indices and the theory of bit vectors as the element theory $\mathcal{T}$. The operator $\sim$ stands for bitwise negation of bit vectors and the relations $\preceq$ and $\approx$ are used as the "at most" and the equality predicate on bit vectors, respectively. Moreover, $a[i]$ denotes a read operation on the array a at index i.

$$\begin{array}{rllrll}
\varphi := & c \geq 1 & \wedge & \forall ij. \quad 0 \leq i \leq j & \to & a[i] \preceq a[j] \\
 & & \wedge & \forall i.\ 0 \leq i \leq c-1 & \to & a[i] \preceq {\sim} a[0] \\
 & & \wedge & & \to & a[c] \approx {\sim} a[0] \\
 & & \wedge & \forall i. \quad i \geq c+1 & \to & a[i] \succeq {\sim} a[0]
\end{array}$$

Translating φ into BSR(SLI+$\mathcal{T}$) yields the following clause set N, in which we consider P_a to be marked.

$$\begin{array}{rcl|rcll}
c < 1 & \| & \to \Box & 0 \leq i, i \leq j & \| & P_a(i,u), P_a(j,v) & \to u \preceq v \\
e \neq c-1 & \| & \to \Box & 0 \leq i, i \leq e, y = 0 & \| & P_a(i,u), P_a(y,v) & \to u \preceq {\sim} v \\
f \neq c+1 & \| & \to \Box & x = c, y = 0 & \| & P_a(x,u), P_a(y,v) & \to u \approx {\sim} v \\
 & & & i \geq f, y = 0 & \| & P_a(i,u), P_a(y,v) & \to u \succeq {\sim} v
\end{array}$$

In order to preserve (un)satisfiability, functional axioms have to be added for P_a (cf. the sets Φ_1 and Φ_2 that we used earlier). Doing so, we leave BSR(SLI+$\mathcal{T}$).

The clause set N induces the set $\mathcal{I}_{\Downarrow\langle P_a,1\rangle} = \{c_{-\infty}, 0, c, f\}$ of instantiation points for the index of the array. An adaptation of Lemma 17 for BSR(SLI+$\mathcal{T}$)

entails that adding the clause set N' corresponding to the following set of sentences yields a BSR(SLI+$\mathcal{T}$) clause set $N \cup N'$ that is equisatisfiable to φ.

$$\begin{aligned}
&\big\{P_a(c', d_{P_a c'}) \;\big|\; c' \in \{c_{-\infty}, 0, c, f\}\big\} \\
&\cup \big\{\forall i.\, \textstyle\bigvee_{c' \in \{c_{-\infty}, 0, c, f\}} P_a(i, d_{P_a c'})\big\} \\
&\cup \big\{\forall iu.\; P_a(i, u) \to \textstyle\bigvee_{c' \in \{c_{-\infty}, 0, c, f\}} u \approx d_{P_a c'}\big\} \\
&\cup \big\{\forall i.\; P_a(i, d_{P_a c'}), P_a(i, d_{P_a c''}) \to d_{P_a c'} \approx d_{P_a c''} \;\big|\; c', c'' \in \{c_{-\infty}, 0, c, f\}\big\}
\end{aligned}$$

Using the instantiation methods that we have developed in Sections 3.1 – 3.3, the set $N \cup N'$ can be turned into an equisatisfiable quantifier-free clause set. One possible (uniform) model $\mathcal{A} \models N \cup N'$ assigns $c_{-\infty}^{\mathcal{A}} = -1$, $e^{\mathcal{A}} = 2$, $c^{\mathcal{A}} = 3$, $f^{\mathcal{A}} = 4$, $d_{P_a c_{-\infty}}^{\mathcal{A}} = 00$, $d_{P_a 0}^{\mathcal{A}} = 01$, $d_{P_a e}^{\mathcal{A}} = 01$, $d_{P_a c}^{\mathcal{A}} = 10$, $d_{P_a f}^{\mathcal{A}} = 11$, and yields the array $\langle 01, 01, 01, 10, 11, 11, 11, \ldots\rangle$. $\square$

5 Discussion

We have demonstrated how universally quantified variables in BSR(SLI) clause sets can be instantiated economically. In certain cases our methods lead to exponentially fewer instances than a naive instantiation with all occurring integer terms would generate. Moreover, we have sketched how defining suitable finite-domain sort predicates instead of explicitly instantiating variables can avoid immediate blow-ups caused by explicit instantiation. It is then left to the theorem prover to actually instantiate variables as needed.

We have shown that our methods are compatible with uninterpreted, non-constant functions under certain restrictions. Even another background theory $\mathcal{T}$ may be added, leading to BSR(SLI+$\mathcal{T}$). This entails applicability of our instantiation approach to known logic fragments, such as the *array property fragment* [8], the *finite essentially uninterpreted fragment with arithmetic literals* [12], and many-sorted first-order formulas over *stratified vocabularies* [1, 16].

The instantiation methodology that we have described specifically for integer variables can also be adapted to work for universally quantified variables ranging over the reals [24]. Our computation of instantiation points considers all argument positions in predicate atoms independently. This can be further refined by considering dependencies between argument positions and clauses. For example, this refinement idea was successfully applied in first-order logic [9, 16].

Once all the integer variables are grounded by successive instantiation, we are left with a clause set where for every integer variable x in any clause there is a defining equation $x = c$ for some constant c. Thus, the clause set can actually be turned into a standard first-order BSR clause set by replacing the integer constants with respective fresh uninterpreted constants. Then, as an alternative to further grounding the free-sort variables, any state-of-the-art BSR decision procedure can be applied to test satisfiability [22, 15, 2]. It is even sufficient to know the instantiation sets for the base sort variables. Then, instead of explicit grounding, by defining respective finite-domain sort predicates for the sets, the worst-case exponential blow-up of grounding can be prevented.

References

1. Aharon Abadi, Alexander Rabinovich, and Mooly Sagiv. Decidable Fragments of Many-Sorted Logic. *Journal of Symbolic Computation*, 45(2):153–172, 2010.
2. Gábor Alagi and Christoph Weidenbach. NRCL – A Model Building Approach to the Bernays–Schönfinkel Fragment. In *Frontiers of Combining Systems (FroCoS'15)*, pages 69–84, 2015.
3. Ernst Althaus, Evgeny Kruglov, and Christoph Weidenbach. Superposition Modulo Linear Arithmetic SUP(LA). In *Frontiers of Combining Systems (FroCoS'09)*, pages 84–99, 2009.
4. Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Refutational Theorem Proving for Hierarchic First-Order Theories. *Applicable Algebra in Engineering, Communication and Computing*, 5:193–212, 1994.
5. Peter Baumgartner and Uwe Waldmann. Hierarchic Superposition with Weak Abstraction. In *Automated Deduction (CADE-24)*, pages 39–57, 2013.
6. Aaron R. Bradley. *Safety Analysis of Systems.* PhD thesis, 2007.
7. Aaron R. Bradley and Zohar Manna. *The Calculus of Computation – Decision Procedures with Applications to Verification.* Springer, 2007.
8. Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. What's Decidable About Arrays? In *Verification, Model Checking, and Abstract Interpretation (VMCAI'06)*, pages 427–442, 2006.
9. Koen Claessen, Ann Lillieström, and Nicholas Smallbone. Sort It Out with Monotonicity – Translating between Many-Sorted and Unsorted First-Order Logic. In *Automated Deduction (CADE-23)*, pages 207–221, 2011.
10. Peter J. Downey. Undecidability of Presburger Arithmetic with a Single Monadic Predicate Letter. Technical report, Center for Research in Computer Technology, Harvard University, 1972.
11. Arnaud Fietzke and Christoph Weidenbach. Superposition as a Decision Procedure for Timed Automata. *Mathematics in Computer Science*, 6(4):409–425, 2012.
12. Yeting Ge and Leonardo Mendonça de Moura. Complete Instantiation for Quantified Formulas in Satisfiabiliby Modulo Theories. In *Computer Aided Verification (CAV'09)*, pages 306–320, 2009.
13. Joseph Y. Halpern. Presburger Arithmetic with Unary Predicates is Π^1_1 Complete. *Journal of Symbolic Logic*, 56(2):637–642, 1991.
14. Matthias Horbach, Marco Voigt, and Christoph Weidenbach. On the Combination of the Bernays–Schönfinkel–Ramsey Fragment with Simple Linear Integer Arithmetic. *ArXiv preprint*, arXiv:1705.08792 [cs.LO], 2017.
15. Konstantin Korovin. Inst-Gen – A Modular Approach to Instantiation-Based Automated Reasoning. In Andrei Voronkov and Christoph Weidenbach, editors, *Programming Logics – Essays in Memory of Harald Ganzinger*, volume 7797 of *Lecture Notes in Computer Science*, pages 239–270. Springer, 2013.
16. Konstantin Korovin. Non-cyclic Sorts for First-Order Satisfiability. In *Frontiers of Combining Systems (FroCoS'13)*, pages 214–228, 2013.
17. Daniel Kroening and Ofer Strichman. *Decision Procedures.* Texts in Theoretical Computer Science. An EATCS Series. Springer, second edition, 2016.
18. Evgeny Kruglov and Christoph Weidenbach. Superposition Decides the First-Order Logic Fragment Over Ground Theories. *Mathematics in Computer Science*, 6(4):427–456, 2012.
19. Harry R. Lewis. Complexity Results for Classes of Quantificational Formulas. *Journal of Computer and System Sciences*, 21(3):317–353, 1980.

20. Rüdiger Loos and Volker Weispfenning. Applying Linear Quantifier Elimination. *The Computer Journal*, 36(5):450–462, 1993.
21. Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM*, 53:937–977, 2006.
22. Ruzica Piskac, Leonardo Mendonça de Moura, and Nikolaj Bjørner. Deciding Effectively Propositional Logic Using DPLL and Substitution Sets. *Journal of Automated Reasoning*, 44(4):401–424, 2010.
23. Hilary Putnam. Decidability and Essential Undecidability. *Journal of Symbolic Logic*, 22(1):39–54, 1957.
24. Marco Voigt and Christoph Weidenbach. Bernays-Schönfinkel-Ramsey with Simple Bounds is NEXPTIME-complete. *ArXiv preprint*, arXiv:1501.07209 [cs.LO], 2015.