Hierarchic Superposition Revisited

Peter Baumgartner[1(✉)] and Uwe Waldmann[2(✉)]

[1] Data61/CSIRO, ANU Computer Science and Information Technology (CSIT), Acton, Australia
Peter.Baumgartner@data61.csiro.au
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany
uwe@mpi-inf.mpg.de

Abstract. Many applications of automated deduction require reasoning in first-order logic modulo background theories, in particular some form of integer arithmetic. A major unsolved research challenge is to design theorem provers that are "reasonably complete" even in the presence of free function symbols ranging into a background theory sort. The hierarchic superposition calculus of Bachmair, Ganzinger, and Waldmann already supports such symbols, but, as we demonstrate, not optimally. This paper aims to rectify the situation by introducing a novel form of clause abstraction, a core component in the hierarchic superposition calculus for transforming clauses into a form needed for internal operation. We argue for the benefits of the resulting calculus and provide two new completeness results: one for the fragment where all background-sorted terms are ground and another one for a special case of linear (integer or rational) arithmetic as a background theory.

Keywords: Automated deduction · Superposition calculus · Combinations of theories

1 Introduction

Many applications of automated deduction require reasoning with respect to a combination of a background theory, say integer arithmetic, and a foreground theory that extends the background theory by new sorts such as *list*, new operators, such as $cons : int \times list \to list$ and $length : list \to int$, and first-order axioms. Developing corresponding automated reasoning systems that are also able to deal with quantified formulas has recently been an active area of research. One major line of research is concerned with extending (SMT-based) solvers [24] for the quantifier-free case by instantiation heuristics for quantifiers [17,18, e. g.]. Another line of research is concerned with adding black-box reasoners for specific background theories to first-order automated reasoning methods (resolution [1,5,19], sequent calculi [26], instantiation methods [8,9,16], etc). In both cases, a major unsolved research challenge is to provide reasoning support that is "reasonably complete" in practice, so that the systems can be used more reliably for both proving theorems and finding counterexamples.

C. Lutz et al. (Eds.): Baader Festschrift, LNCS 11560, pp. 15–56, 2019.
https://doi.org/10.1007/978-3-030-22102-7_2

In [5], Bachmair, Ganzinger, and Waldmann introduced the hierarchical superposition calculus as a generalization of the superposition calculus for black-box style theory reasoning. Their calculus works in a framework of hierarchic specifications. It tries to prove the unsatisfiability of a set of clauses with respect to interpretations that extend a background model such as the integers with linear arithmetic conservatively, that is, without identifying distinct elements of old sorts ("confusion") and without adding new elements to old sorts ("junk"). While confusion can be detected by first-order theorem proving techniques, junk can not – in fact, the set of logical consequences of a hierarchic specifications is usually not recursively enumerable. Refutational completeness can therefore only be guaranteed if one restricts oneself to sets of formulas where junk can be excluded a priori. The property introduced by Bachmair, Ganzinger, and Waldmann for this purpose is called "sufficient completeness with respect to simple instances". Given this property, their calculus is refutationally complete for clause sets that are fully abstracted (i. e., where no literal contains both foreground and background symbols). Unfortunately their full abstraction rule may destroy sufficient completeness with respect to simple instances. We show that this problem can be avoided by using a new form of clause abstraction and a suitably modified hierarchical superposition calculus. Since the new calculus is still refutationally complete and the new abstraction rule is guaranteed to preserve sufficient completeness with respect to simple instances, the new combination is strictly more powerful than the old one.

In practice, sufficient completeness is a rather restrictive property. While there are application areas where one knows in advance that every input is sufficiently complete, in most cases this does not hold. As a user of an automated theorem prover, one would like to see a best effort behavior: The prover might for instance try to *make* the input sufficiently complete by adding further theory axioms. In the calculus from [5], this does not help at all: The restriction to a particular kind of instantiations ("simple instances") renders theory axioms essentially unusable in refutations. We show that this can be prevented by introducing two kinds of variables of the background theory sorts, that can be instantiated in different ways, making our calculus significantly "more complete" in practice. We also include a definition rule in the calculus that can be used to establish sufficient completeness by linking foreground terms to background parameters, thus allowing the background prover to reason about these terms.

The following trivial example demonstrates the problem. Consider the clause set $N = \{C\}$ where $C = \mathsf{f}(1) < \mathsf{f}(1)$. Assume that the background theory is integer arithmetic and that f is an integer-sorted operator from the foreground (free) signature. Intuitively, one would expect N to be unsatisfiable. However, N is not sufficiently complete, and it admits models in which $\mathsf{f}(1)$ is interpreted as some junk element $\not\in$, an element of the domain of the integer sort that is not a numeric constant. So both the calculus in [5] and ours are excused to not find a refutation. To fix that, one could add an instance $C' = \neg(\mathsf{f}(1) < \mathsf{f}(1))$ of the irreflexivity axiom $\neg(x < x)$. The resulting set $N' = \{C,\ C'\}$ is (trivially) sufficiently complete as it has no models at all. However, the calculus in [5] is not helped by adding C', since the abstracted version of N' is again

not sufficiently complete and admits a model that interprets $\mathsf{f}(1)$ as $\not\in$. Our abstraction mechanism always preserves sufficient completeness and our calculus will find a refutation.

With this example one could think that replacing the abstraction mechanism in [5] with ours gives all the advantages of our calculus. But this is not the case. Let $N'' = \{C,\ \neg(x < x)\}$ be obtained by adding the more realistic axiom $\neg(x < x)$. The set N'' is still sufficiently complete with our approach thanks to having two kinds of variables at disposal, but it is not sufficiently complete in the sense of [5]. Indeed, in that calculus adding background theory axioms *never* helps to gain sufficient completeness, as variables there have only one kind.

Another alternative to make N sufficiently complete is by adding a clause that forces $\mathsf{f}(1)$ to be equal to some background domain element. For instance, one can add a "definition" for $\mathsf{f}(1)$, that is, a clause $\mathsf{f}(1) \approx \alpha$, where α is a fresh symbolic constant belonging to the background signature (a "parameter"). The set $N''' = \{C, \mathsf{f}(1) \approx \alpha\}$ is sufficiently complete and it admits refutations with both calculi. The definition rule in our calculus mentioned above will generate this definition automatically. Moreover, the set N belongs to a syntactic fragment for which we can guarantee not only sufficient completeness (by means of the definition rule) but also refutational completeness.

We present the new calculus in detail and provide a general completeness result, modulo compactness of the background theory, and two specific completeness results for clause sets that do not require compactness – one for the fragment where all background-sorted terms are ground and another one for a special case of linear (integer or rational) arithmetic as a background theory.

We also report on experiments with a prototypical implementation on the TPTP problem library [27].

Sections 1–7, 9–10, and 12 of this paper are a substantially expanded and revised version of [11]. A preliminary version of Sect. 11 has appeared in [10]. However, we omit from this paper some proofs that are not essential for the understanding of the main ideas. They can be found in a slightly extended version of this paper at http://arxiv.org/abs/1904.03776 [12].

Related Work. The relation with the predecessor calculus in [5] is discussed above and also further below. What we say there also applies to other developments rooted in that calculus, [1, e. g.]. The specialized version of hierarchic superposition in [22] will be discussed in Sect. 9 below. The resolution calculus in [19] has built-in inference rules for linear (rational) arithmetic, but is complete only under restrictions that effectively prevent quantification over rationals. Earlier work on integrating theory reasoning into model evolution [8,9] lacks the treatment of background-sorted foreground function symbols. The same applies to the sequent calculus in [26], which treats linear arithmetic with built-in rules for quantifier elimination. The instantiation method in [16] requires an answer-complete solver for the background theory to enumerate concrete solutions of background constraints, not just a decision procedure. All these approaches have in common that they integrate specialized reasoning for background theories into a general first-order reasoning method. A conceptually different approach consists in using first-order theorem provers as (semi-)decision procedures for

specific theories in DPLL(T)(-like) architectures [2,13,14]. Notice that in this context the theorem provers do not need to reason modulo background theories themselves, and indeed they don't. The calculus and system in [14], for instance, integrates superposition and DPLL(T). From DPLL(T) it inherits splitting of ground non-unit clauses into their unit components, which determines a (backtrackable) model candidate M. The superposition inference rules are applied to elements from M and a current clause set F. The superposition component guarantees refutational completeness for pure first-order clause logic. Beyond that, for clauses containing background-sorted variables, (heuristic) instantiation is needed. Instantiation is done with ground terms that are provably equal w.r.t. the equations in M to some ground term in M in order to advance the derivation. The limits of that method can be illustrated with an (artificial but simple) example. Consider the unsatisfiable clause set $\{i \leq j \vee \mathsf{P}(i+1,x) \vee \mathsf{P}(j+2,x),$ $i \leq j \vee \neg\mathsf{P}(i+3,x) \vee \neg\,\mathsf{P}(j+4,x)\}$ where i and j are integer-sorted variables and x is a foreground-sorted variable. Neither splitting into unit clauses, superposition calculus rules, nor instantiation applies, and so the derivation gets stuck with an inconclusive result. By contrast, the clause set belongs to a fragment that entails sufficient completeness ("no background-sorted foreground function symbols") and hence is refutable by our calculus. On the other hand, heuristic instantiation does have a place in our calculus, but we leave that for future work.

2 Signatures, Clauses, and Interpretations

We work in the context of standard many-sorted logic with first-order signatures comprised of sorts and operator (or function) symbols of given arities over these sorts. A *signature* is a pair $\Sigma = (\Xi, \Omega)$, where Ξ is a set of *sorts* and Ω is a set of *operator symbols* over Ξ. If $\mathcal{X}$ is a set of sorted variables with sorts in Ξ, then the set of well-sorted terms over $\Sigma = (\Xi, \Omega)$ and $\mathcal{X}$ is denoted by $\mathrm{T}_\Sigma(\mathcal{X})$; T_Σ is short for $\mathrm{T}_\Sigma(\emptyset)$. We require that Σ is a *sensible* signature, i. e., that T_Σ has no empty sorts. As usual, we write $t[u]$ to indicate that the term u is a (not necessarily proper) subterm of the term t. The position of u in t is left implicit.

A *Σ-equation* is an unordered pair (s,t), usually written $s \approx t$, where s and t are terms from $\mathrm{T}_\Sigma(\mathcal{X})$ of the same sort. For simplicity, we use equality as the only predicate in our language. Other predicates can always be encoded as a function into a set with one distinguished element, so that a non-equational atom is turned into an equation $P(t_1,\ldots,t_n) \approx true_P$; this is usually abbreviated by $P(t_1,\ldots,t_n)$.[1] A *literal* is an equation $s \approx t$ or a negated equation $\neg(s \approx t)$, also written as $s \not\approx t$. A *clause* is a multiset of literals, usually written as a disjunction; the empty clause, denoted by $\Box$ is a contradiction. If F is a term, equation, literal or clause, we denote by $\mathrm{vars}(F)$ the set of variables that occur in F. We say F is *ground* if $\mathrm{vars}(F) = \emptyset$.

A *substitution* σ is a mapping from variables to terms that is sort respecting, that is, maps each variable $x \in \mathcal{X}$ to a term of the same sort. Substitutions are

[1] Without loss of generality we assume that there exists a distinct sort for every predicate.

homomorphically extended to terms as usual. We write substitution application in postfix form. A term s is an *instance* of a term t if there is a substitution σ such that $t\sigma = s$. All these notions carry over to equations, literals and clauses in the obvious way. The composition $\sigma\tau$ of the substitutions σ and τ is the substitution that maps every variable x to $(x\sigma)\tau$.

The *domain* of a substitution σ is the set $\mathrm{dom}(\sigma) = \{x \mid x \neq x\sigma\}$. We use only substitutions with finite domains, written as $\sigma = [x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$ where $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$. A *ground substitution* is a substitution that maps every variable in its domain to a ground term. A *ground instance* of F is obtained by applying some ground substitution with domain (at least) $\mathrm{vars}(F)$ to it.

A *Σ-interpretation* I consists of a Ξ-sorted family of carrier sets $\{I_\xi\}_{\xi\in\Xi}$ and of a function $I_f : I_{\xi_1} \times \cdots \times I_{\xi_n} \to I_{\xi_0}$ for every $f : \xi_1 \ldots \xi_n \to \xi_0$ in Ω. The *interpretation* t^I of a ground term t is defined recursively by $f(t_1, \ldots, t_n)^I = I_f(t_1^I, \ldots, t_n^I)$ for $n \geq 0$. An interpretation I is called *term-generated*, if every element of an I_ξ is the interpretation of some ground term of sort ξ. An interpretation I is said to *satisfy* a ground equation $s \approx t$, if s and t have the same interpretation in I; it is said to *satisfy* a negated ground equation $s \not\approx t$, if s and t do not have the same interpretation in I. The interpretation I *satisfies* a ground clause C if at least one of the literals of C is satisfied by I. We also say that a ground clause C is *true in* I, if I satisfies C; and that C is *false in* I, otherwise. A term-generated interpretation I is said to *satisfy* a non-ground clause C if it satisfies all ground instances $C\sigma$; it is called a *model* of a set N of clauses, if it satisfies all clauses of N.[2] We abbreviate the fact that I is a model of N by $I \models N$; $I \models C$ is short for $I \models \{C\}$. We say that N *entails* N', and write $N \models N'$, if every model of N is a model of N'; $N \models C$ is short for $N \models \{C\}$. We say that N and N' are *equivalent*, if $N \models N'$ and $N' \models N$.

If $\mathcal{J}$ is a class of Σ-interpretations, a Σ-clause or clause set is called *$\mathcal{J}$-satisfiable* if at least one $I \in \mathcal{J}$ satisfies the clause or clause set; otherwise it is called *$\mathcal{J}$-unsatisfiable*.

A *specification* is a pair $SP = (\Sigma, \mathcal{J})$, where Σ is a signature and $\mathcal{J}$ is a class of term-generated Σ-interpretations called *models* of the specification SP. We assume that $\mathcal{J}$ is closed under isomorphisms.

We say that a class of Σ-interpretations $\mathcal{J}$ or a specification $(\Sigma, \mathcal{J})$ is *compact*, if every infinite set of Σ-clauses that is $\mathcal{J}$-unsatisfiable has a finite subset that is also $\mathcal{J}$-unsatisfiable.

3 Hierarchic Theorem Proving

In hierarchic theorem proving, we consider a scenario in which a general-purpose foreground theorem prover and a specialized background prover cooperate to

[2] This restriction to term-generated interpretations as models is possible since we are only concerned with refutational theorem proving, i. e., with the derivation of a contradiction.

derive a contradiction from a set of clauses. In the sequel, we will usually abbreviate "foreground" and "background" by "FG" and "BG".

The BG prover accepts as input sets of clauses over a *BG signature* $\Sigma_{\mathrm{B}} = (\Xi_{\mathrm{B}}, \Omega_{\mathrm{B}})$. Elements of Ξ_{B} and Ω_{B} are called *BG sorts* and *BG operators*, respectively. We fix an infinite set $\mathcal{X}_{\mathrm{B}}$ of *BG variables* of sorts in Ξ_{B}. Every BG variable has (is labeled with) a *kind*, which is either *"abstraction"* or *"ordinary"*. Terms over Σ_{B} and $\mathcal{X}_{\mathrm{B}}$ are called *BG terms*. A BG term is called *pure*, if it does not contain ordinary variables; otherwise it is *impure*. These notions apply analogously to equations, literals, clauses, and clause sets.

The BG prover decides the satisfiability of Σ_{B}-clause sets with respect to a *BG specification* $(\Sigma_{\mathrm{B}}, \mathcal{B})$, where $\mathcal{B}$ is a class of term-generated Σ_{B}-interpretations called *BG models*. We assume that $\mathcal{B}$ is closed under isomorphisms.

In most applications of hierarchic theorem proving, the set of BG operators Ω_{B} contains a set of distinguished constant symbols $\Omega_{\mathrm{B}}^{D} \subseteq \Omega_{\mathrm{B}}$ that has the property that $d_1^I \neq d_2^I$ for any two distinct $d_1, d_2 \in \Omega_{\mathrm{B}}^{D}$ and every BG model $I \in \mathcal{B}$. We refer to these constant symbols as *(BG) domain elements*.

While we permit arbitrary classes of BG models, in practice the following three cases are most relevant:

(1) $\mathcal{B}$ consists of exactly one Σ_{B}-interpretation (up to isomorphism), say, the integer numbers over a signature containing all integer constants as domain elements and $\leq, <, +, -$ with the expected arities. In this case, $\mathcal{B}$ is trivially compact; in fact, a set N of Σ_{B}-clauses is $\mathcal{B}$-unsatisfiable if and only if some clause of N is $\mathcal{B}$-unsatisfiable.

(2) Σ_{B} is extended by an infinite number of *parameters*, that is, additional constant symbols. While all interpretations in $\mathcal{B}$ share the same carrier sets $\{I_\xi\}_{\xi \in \Xi_{\mathrm{B}}}$ and interpretations of non-parameter symbols, parameters may be interpreted freely by arbitrary elements of the appropriate I_ξ. The class $\mathcal{B}$ obtained in this way is in general not compact; for instance the infinite set of clauses $\{n \leq \beta \mid n \in \mathbb{N}\}$, where β is a parameter, is unsatisfiable in the integers, but every finite subset is satisfiable.

(3) Σ_{B} is again extended by parameters, however, $\mathcal{B}$ is now the class of all interpretations that satisfy some first-order theory, say, the first-order theory of linear integer arithmetic.[3] Since $\mathcal{B}$ corresponds to a first-order theory, compactness is recovered. It should be noted, however, that $\mathcal{B}$ contains non-standard models, so that for instance the clause set $\{n \leq \beta \mid n \in \mathbb{N}\}$ is now satisfiable (e. g., $\mathbb{Q} \times \mathbb{Z}$ with a lexicographic ordering is a model).

The FG theorem prover accepts as inputs clauses over a signature $\Sigma = (\Xi, \Omega)$, where $\Xi_{\mathrm{B}} \subseteq \Xi$ and $\Omega_{\mathrm{B}} \subseteq \Omega$. The sorts in $\Xi_{\mathrm{F}} = \Xi \setminus \Xi_{\mathrm{B}}$ and the operator symbols in $\Omega_{\mathrm{F}} = \Omega \setminus \Omega_{\mathrm{B}}$ are called *FG sorts* and *FG operators*. Again we fix an

[3] To satisfy the technical requirement that all interpretations in $\mathcal{B}$ are term-generated, we assume that in this case Σ_{B} is suitably extended by an infinite set of constants (or by one constant and one unary function symbol) that are not used in any input formula or theory axiom.

infinite set $\mathcal{X}_\mathrm{F}$ of *FG variables* of sorts in Ξ_F. All FG variables have the kind "ordinary". We define $\mathcal{X} = \mathcal{X}_\mathrm{B} \cup \mathcal{X}_\mathrm{F}$.

In examples we will use $\{0, 1, 2, \dots\}$ to denote BG domain elements, $\{+, -, <, \leq\}$ to denote (non-parameter) BG operators, and the possibly subscripted letters $\{x, y\}$, $\{X, Y\}$, $\{\alpha, \beta\}$, and $\{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{f}, \mathsf{g}\}$ to denote ordinary variables, abstraction variables, parameters, and FG operators, respectively. The letter ζ denotes an ordinary variable or an abstraction variable.

We call a term in $\mathrm{T}_\Sigma(\mathcal{X})$ a *FG term*, if it is not a BG term, that is, if it contains at least one FG operator or FG variable (and analogously for literals or clauses). We emphasize that for a FG operator $\mathsf{f} : \xi_1 \dots \xi_n \to \xi_0$ in Ω_F any of the ξ_i may be a BG sort, and that consequently FG terms may have BG sorts.

If I is a Σ-interpretation, the *restriction* of I to Σ_B, written $I|_{\Sigma_\mathrm{B}}$, is the Σ_B-interpretation that is obtained from I by removing all carrier sets I_ξ for $\xi \in \Xi_\mathrm{F}$ and all functions I_f for $f \in \Omega_\mathrm{F}$. Note that $I|_{\Sigma_\mathrm{B}}$ is not necessarily term-generated even if I is term-generated. In hierarchic theorem proving, we are only interested in Σ-interpretations that extend some model in $\mathcal{B}$ and neither collapse any of its sorts nor add new elements to them, that is, in Σ-interpretations I for which $I|_{\Sigma_\mathrm{B}} \in \mathcal{B}$. We call such a Σ-interpretation a *$\mathcal{B}$-interpretation*.

Let N and N' be two sets of Σ-clauses. We say that N *entails* N' *relative to* $\mathcal{B}$ (and write $N \models_\mathcal{B} N'$), if every model of N whose restriction to Σ_B is in $\mathcal{B}$ is a model of N'. Note that $N \models_\mathcal{B} N'$ follows from $N \models N'$. If $N \models_\mathcal{B} \square$, we call N *$\mathcal{B}$-unsatisfiable*; otherwise, we call it *$\mathcal{B}$-satisfiable*.[4]

Our goal in refutational hierarchic theorem proving is to check whether a given set of Σ-clauses N is false in all $\mathcal{B}$-interpretations, or equivalently, whether N is $\mathcal{B}$-unsatisfiable.

We say that a substitution σ is *simple* if $X\sigma$ is a pure BG term for every abstraction variable $X \in \mathrm{dom}(\sigma)$. For example, $[x \mapsto 1 + Y + \alpha]$, $[X \mapsto 1 + Y + \alpha]$ and $[x \mapsto \mathsf{f}(1)]$ all are simple, whereas $[X \mapsto 1 + y + \alpha]$ and $[X \mapsto \mathsf{f}(1)]$ are not. Let F be a clause or (possibly infinite) clause set. By $\mathrm{sgi}(F)$ we denote the set of simple ground instances of F, that is, the set of all ground instances of (all clauses in) F obtained by simple ground substitutions.

For a BG specification $(\Sigma_\mathrm{B}, \mathcal{B})$, we define $\mathrm{GndTh}(\mathcal{B})$ as the set of all ground Σ_B-formulas that are satisfied by every $I \in \mathcal{B}$.

Definition 3.1 (Sufficient completeness). *A Σ-clause set N is called* sufficiently complete w.r.t. simple instances *if for every Σ-model J of* $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})$[5] *and every ground BG-sorted FG term s there is a ground BG term t such that $J \models s \approx t$.*[6] ◻

[4] If $\Sigma = \Sigma_\mathrm{B}$, this definition coincides with the definition of satisfiability w.r.t. a class of interpretations that was given in Sect. 2. A set N of BG clauses is $\mathcal{B}$-satisfiable if and only if some interpretation of $\mathcal{B}$ is a model of N.

[5] In contrast to [5], we include $\mathrm{GndTh}(\mathcal{B})$ in the definition of sufficient completeness. (This is independent of the abstraction method; it would also have been useful in [5].).

[6] Note that J need *not* be a $\mathcal{B}$-interpretation.

For brevity, we will from now on omit the phrase "w.r.t. simple instances" and speak only of "sufficient completeness". It should be noted, though, that our definition differs from the classical definition of sufficient completeness in the literature on algebraic specifications.

4 Orderings

A *hierarchic reduction ordering* is a strict, well-founded ordering on terms that is compatible with contexts, i. e., $s \succ t$ implies $u[s] \succ u[t]$, and stable under simple substitutions, i. e., $s \succ t$ implies $s\sigma \succ t\sigma$ for every simple σ. In the rest of this paper we assume such a hierarchic reduction ordering $\succ$ that satisfies all of the following: (i) $\succ$ is total on ground terms, (ii) $s \succ d$ for every domain element d and every ground term s that is not a domain element, and (iii) $s \succ t$ for every ground FG term s and every ground BG term t. These conditions are easily satisfied by an LPO with an operator precedence in which FG operators are larger than BG operators and domain elements are minimal with, for example, $\cdots \succ -2 \succ 2 \succ -1 \succ 1 \succ 0$ to achieve well-foundedness.

Condition (iii) and stability under *simple* substitutions together justify to always order $s \succ X$ where s is a non-variable FG term and X is an abstraction variable. By contrast, $s \succ x$ can only hold if $x \in \mathrm{vars}(s)$. Intuitively, the combination of hierarchic reduction orderings and abstraction variables affords ordering more terms.

The ordering $\succ$ is extended to literals over terms by identifying a positive literal $s \approx t$ with the multiset $\{s, t\}$, a negative literal $s \not\approx t$ with $\{s, s, t, t\}$, and using the multiset extension of $\succ$. Clauses are compared by the multiset extension of $\succ$, also denoted by $\succ$.

The non-strict orderings $\succeq$ are defined as $s \succeq t$ if and only if $s \succ t$ or $s = t$ (the latter is multiset equality in case of literals and clauses). A literal L is *maximal* (*strictly maximal*) in a clause $L \vee C$ if there is no $K \in C$ with $K \succ L$ ($K \succeq L$).

5 Weak Abstraction

To refute an input set of Σ-clauses, hierarchic superposition calculi derive BG clauses from them and pass the latter to a BG prover. In order to do this, some separation of the FG and BG vocabulary in a clause is necessary. The technique used for this separation is known as *abstraction*: One (repeatedly) replaces some term q in a clause by a new variable and adds a disequations to the clause, so that $C[q]$ is converted into the equivalent clause $\zeta \not\approx q \vee C[\zeta]$, where ζ is a new (abstraction or ordinary) variable.

The calculus by Bachmair, Ganzinger, and Waldmann [5] works on "fully abstracted" clauses: Background terms occurring below a FG operator or in

an equation between a BG and a FG term or vice versa are abstracted out until one arrives at a clause in which no literal contains both FG and BG operators.

A problematic aspect of any kind of abstraction is that it tends to increase the number of incomparable terms in a clause, which leads to an undesirable growth of the search space of a theorem prover. For instance, if we abstract out the subterms t and t' in a ground clause $\mathsf{f}(t) \approx \mathsf{g}(t')$, we get $x \not\approx t \vee y \not\approx t' \vee \mathsf{f}(x) \approx \mathsf{g}(y)$, and the two new terms $\mathsf{f}(x)$ and $\mathsf{g}(y)$ are incomparable in any reduction ordering. In [5] this problem is mitigated by considering only instances where BG-sorted variables are mapped to BG terms: In the terminology of the current paper, all BG-sorted variables in [5] have the kind "abstraction". This means that, in the example above, we obtain the two terms $\mathsf{f}(X)$ and $\mathsf{g}(Y)$. If we use an LPO with a precedence in which f is larger than g and g is larger than every BG operator, then for every simple ground substitution τ, $\mathsf{f}(X)\tau$ is strictly larger that $\mathsf{g}(Y)\tau$, so we can still consider $\mathsf{f}(X)$ as the only maximal term in the literal.

The advantage of full abstraction is that this clause structure is preserved by all inference rules. There is a serious drawback, however: Consider the clause set $N = \{\, 1 + \mathsf{c} \not\approx 1 + \mathsf{c} \,\}$. Since N is ground, we have $\mathrm{sgi}(N) = N$, and since $\mathrm{sgi}(N)$ is unsatisfiable, N is trivially sufficiently complete. Full abstraction turns N into $N' = \{\, X \not\approx \mathsf{c} \vee 1 + X \not\approx 1 + X \,\}$. In the simple ground instances of N', X is mapped to all pure BG terms. However, there are Σ-interpretations of $\mathrm{sgi}(N')$ in which c is interpreted differently from any pure BG term, so $\mathrm{sgi}(N') \cup \mathrm{GndTh}(\mathcal{B})$ does have a Σ-model and N' is no longer sufficiently complete. In other words, the calculus of [5] is refutationally complete for clause sets that are fully abstracted and sufficiently complete, but full abstraction may destroy sufficient completeness. (In fact, the calculus is not able to refute N'.)

The problem that we have seen is caused by the fact that full abstraction replaces FG terms by abstraction variables, which may not be mapped to FG terms later on. The obvious fix would be to use ordinary variables instead of abstraction variables whenever the term to be abstracted out is not a pure BG term, but as we have seen above, this would increase the number of incomparable terms and it would therefore be detrimental to the performance of the prover.

Full abstraction is a property that is stronger than actually necessary for the completeness proof of [5]. In fact, it was claimed in a footnote in [5] that the calculus could be optimized by abstracting out only non-variable BG terms that occur below a FG operator. This is incorrect, however: Using this abstraction rule, neither our calculus nor the calculus of [5] would be able to refute $\{\, 1 + 1 \approx 2,\ (1 + 1) + \mathsf{c} \not\approx 2 + \mathsf{c} \,\}$, even though this set is unsatisfiable and trivially sufficiently complete. We need a slightly different abstraction rule to avoid this problem:

Definition 5.1. *A BG term q is a* target term *in a clause C if q is neither a domain element nor a variable and if C has the form $C[f(s_1, \ldots, q, \ldots, s_n)]$, where f is a FG operator or at least one of the s_i is a FG or impure BG term.*[7]

A clause is called weakly abstracted *if it does not have any target terms.*

The weakly abstracted version of a clause *is the clause that is obtained by exhaustively replacing $C[q]$ by*

- *$C[X] \vee X \not\approx q$, where X is a new abstraction variable, if q is a pure target term in C,*
- *$C[y] \vee y \not\approx q$, where y is a new ordinary variable, if q is an impure target term in C.*

The weakly abstracted version of a clause C is denoted by $\mathrm{abstr}(C)$*; if N is a set of clauses then* $\mathrm{abstr}(N) = \{\, \mathrm{abstr}(C) \mid C \in N \,\}$. □

For example, weak abstraction of the clause $\mathsf{g}(1, \alpha, \mathsf{f}(1) + (\alpha + 1), z) \approx \beta$ yields $\mathsf{g}(1, X, \mathsf{f}(1) + Y, z) \approx \beta \vee X \not\approx \alpha \vee Y \not\approx \alpha + 1$. Note that the terms 1, $\mathsf{f}(1) + (\alpha + 1)$, z, and β are not abstracted out: 1 is a domain element; $\mathsf{f}(1) + (\alpha + 1)$ has a BG sort, but it is not a BG term; z is a variable; and β is not a proper subterm of any other term. The clause $\mathsf{write}(\mathsf{a}, 2, \mathsf{read}(\mathsf{a}, 1) + 1) \approx \mathsf{b}$ is already weakly abstracted. Every pure BG clause is trivially weakly abstracted.

Nested abstraction is only necessary for certain impure BG terms. For instance, the clause $\mathsf{f}(z + \alpha) \approx 1$ has two target terms, namely α (since z is an impure BG term) and $z + \alpha$ (since f is a FG operator). If we abstract out α, we obtain $\mathsf{f}(z + X) \approx 1 \vee X \not\approx \alpha$. The new term $z + X$ is still a target term, so one more abstraction step yields $\mathsf{f}(y) \approx 1 \vee X \not\approx \alpha \vee y \not\approx z + X$. (Alternatively, we can first abstract out $z + \alpha$, yielding $\mathsf{f}(y) \approx 1 \vee y \not\approx z + \alpha$, and then α. The final result is the same.)

It is easy to see that the abstraction process described in Definition 5.1 terminates by comparing the multisets of the numbers of non-variable occurrences in the left and right-hand sides of all literals before and after an abstraction step.

Proposition 5.2. *If N is a set of clauses and N' is obtained from N by replacing one or more clauses by their weakly abstracted versions, then* $\mathrm{sgi}(N)$ *and* $\mathrm{sgi}(N')$ *are equivalent and N' is sufficiently complete whenever N is.*

Proof. Let us first consider the case of a single abstraction step applied to a single clause. Let $C[q]$ be a clause with a target term q and let $D = C[\zeta] \vee \zeta \not\approx q$ be the result of abstracting out q (where ζ is a new abstraction variable, if q is pure, and a new ordinary variable, if q is impure). We will show that $\mathrm{sgi}(C)$ and $\mathrm{sgi}(D)$ have the same models.

[7] Target terms are terms that need to be abstracted out; so for efficiency reasons, it is advantageous to keep the number of target terms as small as possible. We will show in Sect. 7 why domain elements may be treated differently from other non-variable terms. On the other hand, all the results in the following sections continue to hold if the restriction that q is not a domain element is dropped (i. e., if domain elements are abstracted out as well). We will make use of this fact in Sect. 11.

In one direction let I be an arbitrary model of $\mathrm{sgi}(C)$. We have to show that I is also a model of every simple ground instance $D\tau$ of D. If I satisfies the disequation $\zeta\tau \not\approx q\tau$ then this is trivial. Otherwise, $\zeta\tau$ and $q\tau$ have the same interpretation in I. Since $\mathrm{dom}(\tau) \supseteq \mathrm{vars}(D) = \mathrm{vars}(C) \cup \{\zeta\}$, $C\tau$ is a simple ground instance of C, so I is a model of $C\tau = C\tau[q\tau]$. By congruence, we conclude that I is also a model of $C\tau[\zeta\tau]$, hence it is a model of $D\tau = C\tau[\zeta\tau] \vee \zeta\tau \not\approx q\tau$.

In the other direction let I be an arbitrary model of $\mathrm{sgi}(D)$. We have to show that I is also a model of every simple ground instance $C\tau$ of C. Without loss of generality assume that $\zeta \notin \mathrm{dom}(\tau)$. If ζ is an abstraction variable, then q is a pure BG term, and since τ is a simple substitution, $q\tau$ is a pure BG term as well. Consequently, the substitutions $[\zeta \mapsto q\tau]$ and $\tau' = \tau[\zeta \mapsto q\tau]$ are again simple substitutions and $D\tau'$ is a simple ground instance of D. This implies that I is a model of $D\tau'$. The clause $D\tau'$ has the form $D\tau' = C\tau'[\zeta\tau'] \vee \zeta\tau' \not\approx q\tau'$; since $\zeta\tau' = q\tau$, $C\tau' = C\tau$ and $q\tau' = q\tau$, this is equal to $C\tau[q\tau] \vee q\tau \not\approx q\tau$. Obviously, the literal $q\tau \not\approx q\tau$ must be false in I, so I must be a model of $C\tau[q\tau] = C[q]\tau = C\tau$.

By induction over the number of abstraction steps we conclude that for any clause C, $\mathrm{sgi}(C)$ and $\mathrm{sgi}(\mathrm{abstr}(C))$ are equivalent. The extension to clause sets N and N' follows then from the fact that I is a model of $\mathrm{sgi}(N)$ if and only if it is a model of $\mathrm{sgi}(C)$ for all $C \in N$. Moreover, the equivalence of $\mathrm{sgi}(N)$ and $\mathrm{sgi}(N')$ implies obviously that N' is sufficiently complete whenever N is. □

In contrast to full abstraction, the weak abstraction rule does not require abstraction of FG terms (which can destroy sufficient completeness if done using abstraction variables, and which is detrimental to the performance of a prover if done using ordinary variables). BG terms are usually abstracted out using abstraction variables. The exception are BG terms that are impure, i. e., that contain ordinary variables themselves. In this case, we cannot avoid to use ordinary variables for abstraction, otherwise, we might again destroy sufficient completeness. For example, the clause set $\{\mathsf{P}(1+y),\ \neg\mathsf{P}(1+\mathsf{c})\}$ is sufficiently complete. If we used an abstraction variable instead of an ordinary variable to abstract out the impure subterm $1+y$, we would get $\{\mathsf{P}(X) \vee X \not\approx 1+y,\ \neg\mathsf{P}(1+\mathsf{c})\}$, which is no longer sufficiently complete.

In input clauses (that is, before abstraction), BG-sorted variables may be declared as "ordinary" or "abstraction". As we have seen above, using abstraction variables can reduce the search space; on the other hand, abstraction variables may be detrimental to sufficient completeness. Consider the following example: The set of clauses $N = \{\neg\mathsf{f}(x) > \mathsf{g}(x) \vee \mathsf{h}(x) \approx 1,\ \neg\mathsf{f}(x) \leq \mathsf{g}(x) \vee \mathsf{h}(x) \approx 2,\ \neg\mathsf{h}(x) > 0\}$ is unsatisfiable w.r.t. linear integer arithmetic, but since it is not sufficiently complete, the hierarchic superposition calculus does not detect the unsatisfiability. Adding the clause $X > Y \vee X \leq Y$ to N does not help: Since the abstraction variables X and Y may not be mapped to the FG terms $\mathsf{f}(x)$ and $\mathsf{g}(x)$ in a simple ground instance, the resulting set is still not sufficiently complete. However, if we add the clause $x > y \vee x \leq y$, the

set of clauses becomes (vacuously) sufficiently complete and its unsatisfiability is detected.

One might wonder whether it is also possible to gain anything if the abstraction process is performed using ordinary variables instead of abstraction variables. The following proposition shows that this is not the case:

Proposition 5.3. *Let N be a set of clauses, let N' be the result of weak abstraction of N as defined above, and let N'' be the result of weak abstraction of N where all newly introduced variables are ordinary variables. Then* $\mathrm{sgi}(N')$ *and* $\mathrm{sgi}(N'')$ *are equivalent and* $\mathrm{sgi}(N')$ *is sufficiently complete if and only if* $\mathrm{sgi}(N'')$ *is.*

Proof. By Proposition 5.2, we know already that $\mathrm{sgi}(N)$ and $\mathrm{sgi}(N')$ are equivalent. Moreover, it is easy to check the proof of Proposition 5.2 is still valid if we assume that the newly introduced variable ζ is always an ordinary variable. (Note that the proof requires that abstraction variables are mapped only to pure BG terms, but it does not require that a variable that is mapped to a pure BG term must be an abstraction variable.) So we can conclude in the same way that $\mathrm{sgi}(N)$ and $\mathrm{sgi}(N'')$ are equivalent, and hence, that $\mathrm{sgi}(N')$ and $\mathrm{sgi}(N'')$ are equivalent. From this, we can conclude that N' is sufficiently complete whenever N'' is. □

6 Base Inference System

An *inference system* $\mathcal{I}$ is a set of inference rules. By an $\mathcal{I}$ *inference* we mean an instance of an inference rule from $\mathcal{I}$ such that all conditions are satisfied.

The *base inference system* $\mathrm{HSP}_{\mathrm{Base}}$ of the hierarchic superposition calculus consists of the inference rules Equality resolution, Negative superposition, Positive superposition, Equality factoring, and Close defined below. The calculus is parameterized by a hierarchic reduction ordering $\succ$ and by a "selection function" that assigns to every clause a (possibly empty) subset of its negative FG literals. All inference rules are applicable only to weakly abstracted premise clauses.

$$\text{Equality resolution}\quad \frac{s \not\approx t \vee C}{\mathrm{abstr}(C\sigma)}$$

if (i) σ is a simple mgu of s and t, (ii) $s\sigma$ is not a pure BG term, and (iii) if the premise has selected literals, then $s \not\approx t$ is selected in the premise, otherwise $(s \not\approx t)\sigma$ is maximal in $(s \not\approx t \vee C)\sigma$.[8]

For example, Equality resolution is applicable to $1 + \mathsf{c} \not\approx 1 + x$ with the simple mgu $[x \mapsto \mathsf{c}]$, but it is not applicable to $1 + \alpha \not\approx 1 + x$, since $1 + \alpha$ is a pure BG term.

[8] As in [5], it is possible to strengthen the maximality condition by requiring that there exists some simple ground substitution ψ such that $(s \not\approx t)\sigma\psi$ is maximal in $(s \not\approx t \vee C)\sigma\psi$ (and analogously for the other inference rules).

$$\textsf{Negative superposition}\ \frac{l \approx r \vee C \qquad s[u] \not\approx t \vee D}{\mathrm{abstr}((s[r] \not\approx t \vee C \vee D)\sigma)}$$

if (i) u is not a variable, (ii) σ is a simple mgu of l and u, (iii) $l\sigma$ is not a pure BG term, (iv) $r\sigma \not\succeq l\sigma$, (v) $(l \approx r)\sigma$ is strictly maximal in $(l \approx r \vee C)\sigma$, (vi) the first premise does not have selected literals, (vii) $t\sigma \not\succeq s\sigma$, and (viii) if the second premise has selected literals, then $s \not\approx t$ is selected in the second premise, otherwise $(s \not\approx t)\sigma$ is maximal in $(s \not\approx t \vee D)\sigma$.

$$\textsf{Positive superposition}\ \frac{l \approx r \vee C \qquad s[u] \approx t \vee D}{\mathrm{abstr}((s[r] \approx t \vee C \vee D)\sigma)}$$

if (i) u is not a variable, (ii) σ is a simple mgu of l and u, (iii) $l\sigma$ is not a pure BG term, (iv) $r\sigma \not\succeq l\sigma$, (v) $(l \approx r)\sigma$ is strictly maximal in $(l \approx r \vee C)\sigma$, (vi) $t\sigma \not\succeq s\sigma$, (vii) $(s \not\approx t)\sigma$ is strictly maximal in $(s \approx t \vee D)\sigma$, and (viii) none of the premises has selected literals.

$$\textsf{Equality factoring}\ \frac{s \approx t \vee l \approx r \vee C}{\mathrm{abstr}((l \approx r \vee t \not\approx r \vee C)\sigma)}$$

where (i) σ is a simple mgu of s and l, (ii) $s\sigma$ is not a pure BG term, (iii) $(s \approx t)\sigma$ is maximal in $(s \approx t \vee l \approx r \vee C)\sigma$, (iv) $t\sigma \not\succeq s\sigma$, (v) $l\sigma \not\succeq r\sigma$, and (vi) the premise does not have selected literals.

$$\textsf{Close}\ \frac{C_1 \quad \cdots \quad C_n}{\square}$$

if $C_1, \ldots, C_n$ are BG clauses and $\{C_1, \ldots, C_n\}$ is $\mathcal{B}$-unsatisfiable, i. e., no interpretation in $\mathcal{B}$ is a Σ_{B}-model of $\{C_1, \ldots, C_n\}$.

Notice that Close is not restricted to take *pure* BG clauses only. The reason is that also impure BG clauses admit simple ground instances that are pure.

Theorem 6.1. *The inference rules of* $\mathrm{HSP}_{\mathrm{Base}}$ *are sound w.r.t.* $\models_{\mathcal{B}}$*, i. e., for every inference with premises in* N *and conclusion* C*, we have* $N \models_{\mathcal{B}} C$*.*

Proof. Equality resolution, Negative superposition, Positive superposition, and Equality factoring are clearly sound w.r.t. $\models$, and therefore also sound w.r.t. $\models_{\mathcal{B}}$. For Close, soundness w.r.t. $\models_{\mathcal{B}}$ follows immediately from the definition. □

All inference rules of $\mathrm{HSP}_{\mathrm{Base}}$ involve (simple) mgus. Because of the two kinds of variables, abstraction and ordinary ones, the practical question arises if standard unification algorithms can be used without or only little modification. For example, the terms Z and $(x + y)$ admit a simple mgu $\sigma = [x \mapsto X, y \mapsto Y, Z \mapsto X + Y]$. This prompts for the use of weakening substitutions as in many-sorted logics with subsorts [28]. A closer inspection of the inference rules

shows, however, that such substitutions never need to be considered: All unifiers computed in the inference rules have the property that abstraction variables are only mapped to abstraction variables or domain elements; apart from this additional restriction, we can use a standard unification algorithm.

In contrast to [5], the inference rules above include an explicit weak abstraction in their conclusion. Without it, conclusions would not be weakly abstracted in general. For example Positive superposition applied to the weakly abstracted clauses $\mathsf{f}(X) \approx 1 \vee X \not\approx \alpha$ and $\mathsf{P}(\mathsf{f}(1)+1)$ would then yield $\mathsf{P}(1+1) \vee 1 \not\approx \alpha$, whose P-literal is not weakly abstracted. Additionally, the side conditions of our rules differ somewhat from the corresponding rules of [5], this is due on the one hand to the presence of impure BG terms (which must sometimes be treated like FG terms), and on the other hand to the fact that, after weak abstraction, literals may still contain both FG and BG operators.

The inference rules are supplemented by a redundancy criterion, that is, a mapping $\mathcal{R}_{\mathrm{Cl}}$ from sets of formulae to sets of formulae and a mapping $\mathcal{R}_{\mathrm{Inf}}$ from sets of formulae to sets of inferences that are meant to specify formulae that may be removed from N and inferences that need not be computed. ($\mathcal{R}_{\mathrm{Cl}}(N)$ need not be a subset of N and $\mathcal{R}_{\mathrm{Inf}}(N)$ will usually also contain inferences whose premises are not in N.)

Definition 6.2. *A pair $\mathcal{R} = (\mathcal{R}_{\mathrm{Inf}}, \mathcal{R}_{\mathrm{Cl}})$ is called a* redundancy criterion *(*with respect to an inference system $\mathcal{I}$ and a consequence relation $\models$*), if the following conditions are satisfied for all sets of formulae N and N':*

(i) $N \setminus \mathcal{R}_{\mathrm{Cl}}(N) \models \mathcal{R}_{\mathrm{Cl}}(N)$.
(ii) *If $N \subseteq N'$, then $\mathcal{R}_{\mathrm{Cl}}(N) \subseteq \mathcal{R}_{\mathrm{Cl}}(N')$ and $\mathcal{R}_{\mathrm{Inf}}(N) \subseteq \mathcal{R}_{\mathrm{Inf}}(N')$.*
(iii) *If ι is an inference and its conclusion is in N, then $\iota \in \mathcal{R}_{\mathrm{Inf}}(N)$.*
(iv) *If $N' \subseteq \mathcal{R}_{\mathrm{Cl}}(N)$, then $\mathcal{R}_{\mathrm{Cl}}(N) \subseteq \mathcal{R}_{\mathrm{Cl}}(N \setminus N')$ and $\mathcal{R}_{\mathrm{Inf}}(N) \subseteq \mathcal{R}_{\mathrm{Inf}}(N \setminus N')$.*

The inferences in $\mathcal{R}_{\mathrm{Inf}}(N)$ and the formulae in $\mathcal{R}_{\mathrm{Cl}}(N)$ are said to be redundant *with respect to N.* □

Let SSP be the ground standard superposition calculus using the inference rules equality resolution, negative superposition, positive superposition, and equality factoring (Bachmair and Ganzinger [3], Nieuwenhuis [23], Nieuwenhuis and Rubio [25]). To define a redundancy criterion for $\mathrm{HSP}_{\mathrm{Base}}$ and to prove the refutational completeness of the calculus, we use the same approach as in [5] and relate $\mathrm{HSP}_{\mathrm{Base}}$ inferences to the corresponding SSP inferences.

For a set of ground clauses N, we define $\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(N)$ to be the set of all clauses C such that there exist clauses $C_1, \ldots, C_n \in N$ that are smaller than C with respect to $\succ$ and $C_1, \ldots, C_n \models C$. We define $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N)$ to be the set of all ground SSP inferences ι such that either a premise of ι is in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(N)$ or else C_0 is the conclusion of ι and there exist clauses $C_1, \ldots, C_n \in N$ that are smaller with respect to $\succ^{\mathrm{c}}$ than the maximal premise of ι and $C_1, \ldots, C_n \models C_0$.

The following results can be found in [3] and [23]:

Theorem 6.3. *The (ground) standard superposition calculus* SSP *and $\mathcal{R}^{\mathcal{S}} = (\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}, \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}})$ satisfy the following properties:*

(i) $\mathcal{R}^{\mathcal{S}}$ *is a redundancy criterion with respect to* $\models$.
(ii) SSP *together with* $\mathcal{R}^{\mathcal{S}}$ *is refutationally complete.*

Let ι be an $\mathrm{HSP}_{\mathrm{Base}}$ inference with premises $C_1, \ldots, C_n$ and conclusion $\mathrm{abstr}(C)$, where the clauses $C_1, \ldots, C_n$ have no variables in common. Let ι' be a ground SSP inference with premises $C'_1, \ldots, C'_n$ and conclusion C'. If σ is a simple substitution such that $C' = C\sigma$ and $C'_i = C_i\sigma$ for all i, and if none of the C'_i is a BG clause, then ι' is called a *simple ground instance* of ι. The set of all simple ground instances of an inference ι is denoted by $\mathrm{sgi}(\iota)$.

Definition 6.4. *Let N be a set of weakly abstracted clauses. We define $\mathcal{R}^{\mathcal{H}}_{\mathrm{Inf}}(N)$ to be the set of all inferences ι such that either ι is not a* Close *inference and $\mathrm{sgi}(\iota) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}))$, or else ι is a* Close *inference and $\square \in N$. We define $\mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N)$ to be the set of all weakly abstracted clauses C such that $\mathrm{sgi}(C) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \cup \mathrm{GndTh}(\mathcal{B})$.*[9] □

7 Refutational Completeness

To prove that $\mathrm{HSP}_{\mathrm{Base}}$ and $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_{\mathrm{Inf}}, \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}})$ are refutationally complete for sets of weakly abstracted Σ-clauses and compact BG specifications $(\Sigma_{\mathrm{B}}, \mathcal{B})$, we use the same technique as in [5]:

First we show that $\mathcal{R}^{\mathcal{H}}$ is a redundancy criterion with respect to $\models_{\mathcal{B}}$, and that a set of clauses remains sufficiently complete if new clauses are added or if redundant clauses are deleted. The proofs are rather technical and can be found in [12]. They are similar to the corresponding ones in [5]; the differences are due, on the one hand, to the fact that we include $\mathrm{GndTh}(\mathcal{B})$ in the redundancy criterion and in the definition of sufficient completeness, and, on the other hand, to the explicit abstraction steps in our inference rules.

Lemma 7.1. *If* $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}) \models \mathrm{sgi}(C)$, *then* $N \models_{\mathcal{B}} C$.

Proof. Suppose that $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}) \models \mathrm{sgi}(C)$. Let I' be a Σ-model of N whose restriction to Σ_{B} is contained in $\mathcal{B}$. Clearly, I' is also a model of $\mathrm{GndTh}(\mathcal{B})$. Since I' does not add new elements to the sorts of $I = I'|_{\Sigma_{\mathrm{B}}}$ and I is a term-generated Σ_{B}-interpretation, we know that for every ground Σ-term t' of a BG sort there is a ground BG term t such that t and t' have the same interpretation in I'. Therefore, for every ground substitution σ' there is an equivalent simple ground substitution σ; since $C\sigma$ is valid in I', $C\sigma'$ is also valid. □

We call the simple most general unifier σ that is computed during an inference ι and applied to the conclusion the pivotal substitution of ι. (For ground inferences, the pivotal substitution is the identity mapping.) If L is the literal $[\neg]\, s \approx t$ or $[\neg]\, s[u] \approx t$ of the second or only premise that is eliminated in ι, we call $L\sigma$ the pivotal literal of ι, and we call $s\sigma$ or $s[u]\sigma$ the pivotal term of ι.

[9] In contrast to [5], we include $\mathrm{GndTh}(\mathcal{B})$ in the redundancy criterion. (This is independent of the abstraction method used; it would also have been useful in [5].).

Lemma 7.2. *Let ι be an* $\mathrm{HSP}_{\mathrm{Base}}$ *inference*

$$\frac{C_1}{\mathrm{abstr}(C_0\sigma)} \quad \textit{or} \quad \frac{C_2 \qquad C_1}{\mathrm{abstr}(C_0\sigma)}$$

from weakly abstracted premises with pivotal substitution σ. Let ι' be a simple ground instance of ι of the form

$$\frac{C_1\tau}{C_0\sigma\tau} \quad \textit{or} \quad \frac{C_2\tau \qquad C_1\tau}{C_0\sigma\tau}$$

Then there is a simple ground instance of $\mathrm{abstr}(C_0\sigma)$ *that has the form $C_0\sigma\tau \vee E$, where E is a (possibly empty) disjunction of literals $s \not\approx s$, and each literal of E is smaller than the pivotal literal of ι'.*

As $M \subseteq M'$ implies $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(M) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(M')$, we obtain $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{sgi}(N)\backslash\mathrm{sgi}(N')) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{sgi}(N\backslash N'))$. Furthermore, it is fairly easy to see that $\mathrm{sgi}(N)\backslash(\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \cup \mathrm{GndTh}(\mathcal{B})) \subseteq \mathrm{sgi}(N \setminus \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N))$. Using these two results we can prove the following lemmas:

Lemma 7.3. *$\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_{\mathrm{Inf}}, \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}})$ is a redundancy criterion with respect to $\models_{\mathcal{B}}$.*

Lemma 7.4. *Let N, N' and M be sets of weakly abstracted clauses such that $N' \subseteq \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N)$. If N is sufficiently complete, then so are $N \cup M$ and $N \setminus N'$.*

We now encode arbitrary term-generated Σ_{B}-interpretation by sets of unit ground clauses in the following way: Let $I \in \mathcal{B}$ be a term-generated Σ_{B}-interpretation. For every Σ_{B}-ground term t let $m(t)$ be the smallest ground term of the congruence class of t in I. We define a rewrite system E'_I by $\mathrm{E}'_I = \{t \to m(t) \mid t \in \mathrm{T}_\Sigma,\, t \neq m(t)\}$. Obviously, E'_I is right-reduced; since all rewrite rules are contained in $\succ$, E'_I is terminating; and since every ground term t has $m(t)$ as its only normal form, E'_I is also confluent. Now let E_I be the set of all rules $l \to r$ in E'_I such that l is not reducible by $\mathrm{E}'_I \setminus \{l \to r\}$. Clearly every term that is reducible by E_I is also reducible by E'_I; conversely every term that is reducible by E'_I has a minimal subterm that is reducible by E'_I and the rule in E'_I that is used to rewrite this minimal subterm is necessarily contained in E_I. Therefore E'_I and E_I define the same set of normal forms, and from this we can conclude that E_I and E'_I induce the same equality relation on ground Σ_{B}-terms. We identify E_I with the set of clauses $\{t \approx t' \mid t \to t' \in \mathrm{E}_I\}$. Let D_I be the set of all clauses $t \not\approx t'$, such that t and t' are distinct ground Σ_{B}-terms in normal form with respect to E_I.[10]

Lemma 7.5. *Let $I \in \mathcal{B}$ be a term-generated Σ_{B}-interpretation and let C be a ground BG clause. Then C is true in I if and only if there exist clauses $C_1, \ldots, C_n$ in $\mathrm{E}_I \cup \mathrm{D}_I$ such that $C_1, \ldots, C_n \models C$ and $C \succeq C_i$ for $1 \leq i \leq n$.*

[10] Typically, E_I contains two kinds of clauses, namely clauses that evaluate non-constant BG terms, such as $2 + 3 \approx 5$, and clauses that map parameters to domain elements, such as $\alpha \approx 4$.

Proof. The "if" part follows immediately from the fact that $I \models \mathrm{E}_I \cup \mathrm{D}_I$. For the "only if" part assume that the ground BG clause C is true in I. Consequently, there is some literal $s \approx t$ or $s \not\approx t$ of C that is true in I. Then this literal follows from (i) the rewrite rules in E_I that are used to normalize s to its normal form s', (ii) the rewrite rules in E_I that are used to normalize t to its normal form t', and, in the case of a negated literal $s \not\approx t$, (iii) the clause $s' \not\approx t' \in \mathrm{D}_I$. It is routine to show that all these clauses are smaller than or equal to $s \approx t$ or $s \not\approx t$, respectively, and hence smaller than or equal to C. □

Corollary 7.6. *Let $I \in \mathcal{B}$ be a term-generated Σ_{B}-interpretation. Then* $\mathrm{E}_I \cup \mathrm{D}_I \models \mathrm{GndTh}(\mathcal{B})$.

Proof. Since $I \in \mathcal{B}$, we have $I \models \mathrm{GndTh}(\mathcal{B})$, hence $\mathrm{E}_I \cup \mathrm{D}_I \models \mathrm{GndTh}(\mathcal{B})$ by Lemma 7.5. □

Let N be a set of weakly abstracted clauses and $I \in \mathcal{B}$ be a term-generated Σ_{B}-interpretation, then N_I denotes the set $\mathrm{E}_I \cup \mathrm{D}_I \cup \{\, C\sigma \mid \sigma$ simple, reduced with respect to E_I, $C \in N$, $C\sigma$ ground $\}$.

Lemma 7.7. *If N is a set of weakly abstracted clauses, then* $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$.

Proof. By part (i) of Theorem 6.3 we have obviously $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{sgi}(N)) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}))$. Let C be a clause in $\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})$ and not in N_I. If $C \in \mathrm{GndTh}(\mathcal{B})$, then it is true in I, so by Lemma 7.5 it is either contained in $\mathrm{E}_I \cup \mathrm{D}_I \subseteq N_I$ or it follows from smaller clauses in $\mathrm{E}_I \cup \mathrm{D}_I$ and is therefore in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N))$. If $C \notin \mathrm{GndTh}(\mathcal{B})$, then $C = C'\sigma$ for some $C' \in N$, so it follows from $C'\rho$ and $\mathrm{E}_I \cup \mathrm{D}_I$, where ρ is the substitution that maps every variable ζ to the E_I-normal form of $\zeta\sigma$. Since C follows from smaller clauses in $\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N)$, it is in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N))$. Hence $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$. □

A clause set N is called *saturated (with respect to an inference system $\mathcal{I}$ and a redundancy criterion $\mathcal{R}$)* if $\iota \in \mathcal{R}_{\mathrm{Inf}}(N)$ for every inference ι with premises in N.

Theorem 7.8. *Let $I \in \mathcal{B}$ be a term-generated Σ_{B}-interpretation and let N be a set of weakly abstracted Σ-clauses. If I satisfies all BG clauses in* $\mathrm{sgi}(N)$ *and N is saturated with respect to* $\mathrm{HSP}_{\mathrm{Base}}$ *and $\mathcal{R}^{\mathcal{H}}$, then N_I is saturated with respect to* SSP *and $\mathcal{R}^{\mathcal{S}}$.*

Proof. We have to show that every SSP-inference from clauses of N_I is contained in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$. We demonstrate this in detail for the equality resolution and the negative superposition rule. The analysis of the other rules is similar. Note that by Lemma 7.5 every BG clause that is true in I and is not contained in $\mathrm{E}_I \cup \mathrm{D}_I$ follows from smaller clauses in $\mathrm{E}_I \cup \mathrm{D}_I$, thus it is in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(N_I)$; every inference involving such a clause is in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$.

The equality resolution rule is obviously not applicable to clauses from $\mathrm{E}_I \cup \mathrm{D}_I$. Suppose that ι is an equality resolution inference with a premise $C\sigma$, where

$C \in N$ and σ is a simple substitution and reduced with respect to E_I. If $C\sigma$ is a BG clause, then ι is in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$. If the pivotal term of ι is a pure BG term then the pivotal literal is pure BG as well. Because the pivotal literal is maximal in $C\sigma$ it follows from properties of the ordering that $C\sigma$ is a BG clause. Because we have already considered this case we can assume from now on that the pivotal term of ι is not pure BG and that $C\sigma$ is an FG clause. It follows that ι is a simple ground instance of a hierarchic inference ι' from C. Since ι' is in $\mathcal{R}^{\mathcal{H}}_{\mathrm{Inf}}(N)$, ι is in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}))$, by Lemma 7.7, this implies again $\iota \in \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$.

Obviously a clause from D_I cannot be the first premise of a negative superposition inference. Suppose that the first premise is a clause from E_I. The second premise cannot be a FG clause, since the maximal sides of maximal literals in a FG clause are reduced; as it is a BG clause, the inference is redundant. Now suppose that ι is a negative superposition inference with a first premise $C\sigma$, where $C \in N$ and σ is a simple substitution and reduced with respect to E_I. If $C\sigma$ is a BG clause, then ι is in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$. Otherwise, with the same arguments as for the equality resolution case above, the pivotal term is not pure BG and $C\sigma$ is a FG clause. Hence we can conclude that the second premise can be written as $C'\sigma$, where $C' \in N$ is a FG clause (without loss of generality, C and C' do not have common variables). If the overlap takes place below a variable occurrence, the conclusion of the inference follows from $C\sigma$ and some instance $C'\rho$, which are both smaller than $C'\sigma$. Otherwise, ι is a simple ground instance of a hierarchic inference ι' from C. In both cases, ι is contained in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$. □

The crucial property of abstracted clauses that is needed in the proof of this theorem is that there are no superposition inferences between clauses in E_I and FG ground instances $C\sigma$ in N_I, or in other words, that all FG terms occurring in ground instances $C\sigma$ are reduced w.r.t. E_I. This motivates the definition of target terms in Definition 5.1: Recall that two different domain elements must always be interpreted differently in I and that a domain element is smaller in the term ordering than any ground term that is not a domain element. Consequently, any domain element is the smallest term in its congruence class, so it is reduced by E_I. Furthermore, by the definition of N_I, $\zeta\sigma$ is reduced by E_I for every variable ζ. So variables and domain elements never need to be abstracted out. Other BG terms (such as parameters α or non-constant terms $\zeta_1 + \zeta_2$) have to be abstracted out if they occur below a FG operator, or if one of their sibling terms is a FG term or an impure BG term (since σ can map the latter to a FG term). On the other hand, abstracting out FG terms as in [5] is never necessary to ensure that FG terms are reduced w.r.t. E_I.

If N is saturated with respect to $\mathrm{HSP}_{\mathrm{Base}}$ and $\mathcal{R}^{\mathcal{H}}$ and does not contain the empty clause, then Close cannot be applicable to N. If $(\Sigma_{\mathrm{B}}, \mathcal{B})$ is compact, this implies that there is some term-generated Σ_{B}-interpretation $I \in \mathcal{B}$ that satisfies all BG clauses in $\mathrm{sgi}(N)$. Hence, by Theorem 7.8, the set of *reduced simple* ground instances of N has a model that also satisfies $\mathrm{E}_I \cup \mathrm{D}_I$. Sufficient completeness allows us to show that this is in fact a model of *all* ground instances of clauses in N and that I is its restriction to Σ_{B}:

Theorem 7.9. *If the BG specification* $(\Sigma_{\mathrm{B}}, \mathcal{B})$ *is compact, then* $\mathrm{HSP}_{\mathrm{Base}}$ *and* $\mathcal{R}^{\mathcal{H}}$ *are statically refutationally complete for all sufficiently complete sets of clauses, i. e., if a set of clauses* N *is sufficiently complete and saturated w.r.t.* $\mathrm{HSP}_{\mathrm{Base}}$ *and* $\mathcal{R}^{\mathcal{H}}$*, and* $N \models_{\mathcal{B}} \square$*, then* $\square \in N$*.*

Proof. Let N be a set of weakly abstracted clauses that is sufficiently complete, and saturated w.r.t.the hierarchic superposition calculus and $\mathcal{R}^{\mathcal{H}}$ and does not contain $\square$. Consequently, the Close rule is not applicable to N. By compactness, this means that the set of all Σ_{B}-clauses in $\mathrm{sgi}(N)$ is satisfied by some term-generated Σ_{B}-interpretation $I \in \mathcal{B}$. By Theorem 7.8, N_I is saturated with respect to the standard superposition calculus. Since $\square \notin N_I$, the refutational completeness of standard superposition implies that there is a Σ-model I' of N_I. Since N is sufficiently complete, we know that for every ground term t' of a BG sort there exists a BG term t such that $t' \approx t$ is true in I'. Consequently, for every ground instance of a clause in N there exists an equivalent simple ground instance, thus I' is also a model of all ground instances of clauses in N. To see that the restriction of I' to Σ_{B} is isomorphic to I and thus in $\mathcal{B}$, note that I' satisfies $\mathrm{E}_I \cup \mathrm{D}_I$, preventing confusion, and that N is sufficiently complete, preventing junk. Since I' satisfies N and $I'|_{\Sigma_{\mathrm{B}}} \in \mathcal{B}$, we have $N \not\models_{\mathcal{B}} \square$ □

A theorem proving derivation $\mathcal{D}$ is a finite or infinite sequence of weakly abstracted clause sets $N_0, N_1, ...$, such that N_i and N_{i+1} are equisatisfiable w.r.t. $\models_{\mathcal{B}}$ and $N_i \setminus N_{i+1} \subseteq \mathcal{R}^{\mathcal{H}}_{\mathrm{Inf}}(N_{i+1})$ for all indices i. The set $N_\infty = \bigcup_{i\geq 0} \bigcap_{j\geq i} N_j$ is called the limit of $\mathcal{D}$; the set $N^\infty = \bigcup_{i\geq 0} N_i$ is called the union of $\mathcal{D}$. It is easy to show that every clause in N^∞ is either contained in N_∞ or redundant w.r.t. N_∞. The derivation $\mathcal{D}$ is said to be fair, if every $\mathrm{HSP}_{\mathrm{Base}}$-inference with (non-redundant) premises in N_∞ becomes redundant at some point of the derivation. The limit of a fair derivation is saturated [4]; this is the key result that allows us to deduce dynamic refutational completeness from static refutational completeness:

Theorem 7.10. *If the BG specification* $(\Sigma_{\mathrm{B}}, \mathcal{B})$ *is compact, then* $\mathrm{HSP}_{\mathrm{Base}}$ *and* $\mathcal{R}^{\mathcal{H}}$ *are dynamically refutationally complete for all sufficiently complete sets of clauses, i. e., if* $N \models_{\mathcal{B}} \square$*, then every fair derivation starting from* $\mathrm{abstr}(N)$ *eventually generates* $\square$*.*

In the rest of the paper, we consider only theorem proving derivations where each set N_{i+1} results from from N_i by either adding the conclusions of inferences from N_i, or by deleting clauses that are redundant w.r.t. N_{i+1}, or by applying the following generic simplification rule for clause sets:

$$\textsf{Simp}\ \frac{N \cup \{C\}}{N \cup \{D_1, \ldots, D_n\}}$$

if $n \geq 0$ and (i) D_i is weakly abstracted, for all $i = 1, \ldots, n$, (ii) $N \cup \{C\} \models_{\mathcal{B}} D_i$, and (iii) $C \in \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N \cup \{D_1, \ldots, D_n\})$.

Condition (ii) is needed for soundness, and condition (iii) is needed for completeness. The Simp rule covers the usual simplification rules of the standard superposition calculus, such as demodulation by unit clauses and deletion of tautologies and (properly) subsumed clauses. It also covers simplification of arithmetic terms, e. g., replacing a subterm $(2 + 3) + \alpha$ by $5 + \alpha$ and deleting an unsatisfiable BG literal $5 + \alpha < 4 + \alpha$ from a clause. Any clause of the form $C \vee \zeta \not\approx d$ where d is domain element can be simplified to $C[\zeta \mapsto d]$. Notice, though, that impure BG terms or FG terms can in general not be simplified by BG tautologies. Although, e. g., $\mathsf{f}(X) + 1 \not\approx y + 1$ is larger than $1 + \mathsf{f}(X) \not\approx y + 1$ (with a LPO), such a "simplification" is not justified by the redundancy criterion. Indeed, in the example it destroys sufficient completeness.

We have to point out a limitation of the calculus described above. The standard superposition calculus SSP exists in two variants: either using the Equality factoring rule, or using the Factoring and Merging paramodulation rules. Only the first of these variants works together with weak abstraction. Consider the following example. Let $N = \{\, \alpha + \beta \approx \alpha,\ \mathsf{c} \not\approx \beta \vee \mathsf{c} \not\approx 0,\ \mathsf{c} \approx \beta \vee \mathsf{c} \approx 0 \,\}$. All clauses in N are weakly abstracted. Since the first clause entails $\beta \approx 0$ relative to linear arithmetic, the second and the third clause are obviously contradictory. The $\mathrm{HSP}_{\mathrm{Base}}$ calculus as defined above is able to detect this by first applying Equality factoring to the third clause, yielding $\mathsf{c} \approx 0 \vee \beta \not\approx 0$, followed by two Negative superposition steps and Close. If Equality factoring is replaced by Factoring and Merging paramodulation, however, the refutational completeness of $\mathrm{HSP}_{\mathrm{Base}}$ is lost. The only inference that remains possible is a Negative superposition inference between the third and the second clause. But since the conclusion of this inference is a tautology, the inference is redundant, so the clause set is saturated. (Note that the clause $\beta \approx 0$ is entailed by N, but it is not explicitly present, so there is no way to perform a Merging paramodulation inference with the smaller side of the maximal literal of the third clause.)

8 Local Sufficient Completeness

The definition of sufficient completeness w.r.t. simple instances that was given in Sect. 3 requires that *every* ground BG-sorted FG term s is equal to some ground BG term t in every Σ-model J of $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})$. It is rather evident, however, that this condition is sometimes stronger than needed. For instance, if the set of input clauses N is ground, then we only have to consider the ground BG-sorted FG terms that actually occur in N [22] (analogously to the Nelson-Oppen combination procedure). A relaxation of sufficient completeness that is also useful for non-ground clauses and that still ensures refutational completeness was given by Kruglov [21]:

Definition 8.1 (Smooth ground instance). *We say that a substitution σ is* smooth *if for every variable $\zeta \in \mathrm{dom}(\sigma)$ all BG-sorted (proper or non-proper) subterms of $\zeta\sigma$ are pure BG terms. If $F\sigma$ is a ground instance of a term or clause F and σ is smooth, $F\sigma$ is called a smooth ground instance. (Recall that*

every ground BG term is necessarily pure.) If N is a set of clauses, $\mathrm{smgi}(N)$ *denotes the set of all smooth ground instances of clauses in N.* □

Every smooth substitution is a simple substitution, but not vice versa. For instance, if x is a FG-sorted variable and y is an ordinary BG-sorted variable, then $\sigma_1 = [x \mapsto \mathsf{cons}(\mathsf{f}(1) + 2, \mathsf{empty})]$ and $\sigma_2 = [y \mapsto \mathsf{f}(1)]$ are simple substitutions, but neither of them is smooth, since $x\sigma_1$ and $y\sigma_2$ contain the BG-sorted FG subterm $\mathsf{f}(1)$.

Definition 8.2 (Local sufficient completeness). *Let N be a Σ-clause set. We say that N is* locally sufficiently complete w.r.t. smooth instances *if for every Σ_{B}-interpretation $I \in \mathcal{B}$, every Σ-model J of* $\mathrm{sgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I$*, and every BG-sorted FG term s occurring in* $\mathrm{smgi}(N) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$ *there is a ground BG term t such that $J \models s \approx t$. (Again, we will from now on omit the phrase "w.r.t. smooth instances" for brevity.)* □

Example 8.3. The clause set $N = \{\, X \not\approx \alpha \vee \mathsf{f}(X) \approx \beta \,\}$ is locally sufficiently complete: The smooth ground instances have the form $s' \not\approx \alpha \vee \mathsf{f}(s') \approx \beta$, where s' is a pure BG term. We have to show that $\mathsf{f}(s')$ equals some ground BG term t whenever the smooth ground instance is not redundant. Let $I \in \mathcal{B}$ be a Σ_{B}-interpretation and J be a Σ-model of $\mathrm{sgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I$. If $I \models s' \not\approx \alpha$, then $s' \not\approx \alpha$ follows from some clauses in $\mathrm{E}_I \cup \mathrm{D}_I$, so $s' \not\approx \alpha \vee \mathsf{f}(s') \approx \beta$ is contained in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$ and $\mathsf{f}(s')$ need not be considered. Otherwise $I \models s' \approx \alpha$, then $\mathsf{f}(s')$ occurs in a non-redundant smooth ground instance of a clause in N and $J \models f(s') \approx \beta$, so $t := \beta$ has the desired property. On the other hand, N is clearly not sufficiently complete, since there are models of $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})$ in which $\mathsf{f}(\beta)$ is interpreted by some junk element that is different from the interpretation of any ground BG term.

The example demonstrates that local sufficient completeness is significantly more powerful than sufficient completeness, but this comes at a price. For instance, as shown by the next example, local sufficient completeness is not preserved by abstraction:

Example 8.4. Suppose that the BG specification is linear integer arithmetic (including parameters α, β, γ), the FG operators are $\mathsf{f} : \mathit{int} \to \mathit{int}$, $\mathsf{g} : \mathit{int} \to \mathit{data}$, $\mathsf{a} : \to \mathit{data}$, the term ordering is an LPO with precedence $\mathsf{g} > \mathsf{f} > \mathsf{a} > \gamma > \beta > \alpha > 3 > 2 > 1$, and the clause set N is given by

$$\gamma \approx 1 \quad (1) \qquad \mathsf{f}(2) \approx 2 \quad (4) \qquad \mathsf{g}(\mathsf{f}(\alpha)) \approx \mathsf{a} \;\vee\; \mathsf{g}(\mathsf{f}(\beta)) \approx \mathsf{a} \quad (6)$$

$$\beta \approx 2 \quad (2) \qquad \mathsf{f}(3) \approx 3 \quad (5) \qquad \mathsf{g}(\mathsf{f}(\alpha)) \not\approx \mathsf{a} \;\vee\; \mathsf{g}(\mathsf{f}(\beta)) \approx \mathsf{a} \quad (7)$$

$$\alpha \approx 3 \quad (3) \qquad\qquad\qquad \mathsf{g}(\mathsf{f}(\gamma)) \approx \mathsf{a} \;\vee\; \mathsf{g}(\mathsf{f}(\beta)) \approx \mathsf{a} \quad (8)$$

Since all clauses in N are ground, $\mathrm{smgi}(N) = \mathrm{sgi}(N) = N$. Clause (8) is redundant w.r.t. $\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I$ (for any I): it follows from clauses (6) and (7), and both are smaller than (8). The BG-sorted FG terms in non-redundant clauses are $\mathsf{f}(2)$, $\mathsf{f}(3)$, $\mathsf{f}(\alpha)$, and $\mathsf{f}(\beta)$, and in any Σ-model J of $\mathrm{sgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I$,

these are necessarily equal to the BG terms 2 or 3, respectively, so N is locally sufficiently complete.

Let $N' = \mathrm{abstr}(N)$, let I be a BG-model such that E_I contains $\alpha \approx 3$, $\beta \approx 2$, and $\gamma \approx 1$ (among others), D_I contains $1 \not\approx 2$, $1 \not\approx 3$, and $2 \not\approx 3$ (among others), and let J be a Σ-model of $\mathrm{sgi}(N') \cup \mathrm{E}_I \cup \mathrm{D}_I$ in which $\mathsf{f}(1)$ is interpreted by some junk element. The set N' contains the clause $\mathsf{g}(\mathsf{f}(X)) \approx \mathsf{a} \vee \mathsf{g}(\mathsf{f}(Y)) \approx \mathsf{a} \vee \gamma \not\approx X \vee \beta \not\approx Y$ obtained by abstraction of (8). Its smooth ground instance $C = \mathsf{g}(\mathsf{f}(1)) \approx \mathsf{a} \vee \mathsf{g}(\mathsf{f}(2)) \approx \mathsf{a} \vee \gamma \not\approx 1 \vee \beta \not\approx 2$ is not redundant: it follows from other clauses in $\mathrm{smgi}(N') \cup \mathrm{E}_I \cup \mathrm{D}_I$, namely

$$\alpha \approx 3 \tag{3}$$

$$\mathsf{g}(\mathsf{f}(3)) \approx \mathsf{a} \vee \mathsf{g}(\mathsf{f}(2)) \approx \mathsf{a} \vee \alpha \not\approx 3 \vee \beta \not\approx 2 \tag{6'}$$

$$\mathsf{g}(\mathsf{f}(3)) \not\approx \mathsf{a} \vee \mathsf{g}(\mathsf{f}(2)) \approx \mathsf{a} \vee \alpha \not\approx 3 \vee \beta \not\approx 2 \tag{7'}$$

but the ground instances (6′) and (7′) that are needed here are larger than C. Since C contains the BG-sorted FG term $\mathsf{f}(1)$ which is interpreted differently from any BG term in J, N' is not locally sufficiently complete.

Local sufficient completeness of a clause set suffices to ensure refutational completeness. Kruglov's proof [21] works also if one uses weak abstraction instead of strong abstraction and ordinary as well as abstraction variables, but it relies on an additional restriction on the term ordering.[11] We give an alternative proof that works without this restriction.

The proof is based on a transformation on Σ-interpretations. Let J be an arbitrary Σ-interpretation. We transform J into a term-generated Σ-interpretation $\mathrm{nojunk}(J)$ without junk in two steps. In the first step, we define a Σ-interpretation J' as follows:

- For every FG sort ξ, define $J'_\xi = J_\xi$.
- For every BG sort ξ, define $J'_\xi = \{\, t^J \mid t \text{ is a ground BG term of sort } \xi \,\}$.
- For every $f : \xi_1 \dots \xi_n \to \xi_0$ the function $J'_f : J'_{\xi_1} \times \cdots \times J'_{\xi_n} \to J'_{\xi_0}$ maps $(a_1, \dots, a_n)$ to $J_f(a_1, \dots, a_n)$, if $J_f(a_1, \dots, a_n) \in J'_{\xi_0}$, and to an arbitrary element of J'_{ξ_0} otherwise.

That is, we obtain J' from J be deleting all junk elements from J_ξ if ξ is a BG sort, and by redefining the interpretation of f arbitrarily whenever $J_f(a_1, \dots, a_n)$ is a junk element.

In the second step, we define the Σ-interpretation $\mathrm{nojunk}(J) = J''$ as the term-generated subinterpretation of J', that is,

- For every sort ξ, $J''_\xi = \{\, t^{J'} \mid t \text{ is a ground term of sort } \xi \,\}$,
- For every $f : \xi_1 \dots \xi_n \to \xi_0$, the function $J''_f : J''_{\xi_1} \times \cdots \times J''_{\xi_n} \to J''_{\xi_0}$ satisfies $J''_f(a_1, \dots, a_n) = J'_f(a_1, \dots, a_n)$.

[11] In [21], it is required that every ground term that contains a (proper or improper) BG-sorted FG subterm must be larger than any (BG or FG) ground term that does not contain such a subterm.

Lemma 8.5. *Let J, J', and* $\mathrm{nojunk}(J) = J''$ *be given as above. Then the following properties hold:*

(i) $t^{J''} = t^{J'}$ *for every ground term* t.
(ii) $J''_\xi = J'_\xi$ *for every BG sort* ξ.
(iii) J'' *is a term-generated* Σ*-interpretation and* $J''|_{\Sigma_\mathrm{B}}$ *is a term-generated* Σ_B*-interpretation.*
(iv) *If* $t = f(t_1, \ldots, t_n)$ *is ground,* $t_i^{J'} = t_i^J$ *for all* i*, and* $t^J \in J'_\xi$*, then* $t^{J'} = t^J$.
(v) *If* t *is ground and all BG-sorted subterms of* t *are BG terms, then* $t^{J'} = t^J$.
(vi) *If* C *is a ground BG clause, then* $J \models C$ *if and only if* $J'' \models C$ *if and only if* $J''|_{\Sigma_\mathrm{B}} \models C$.

Proof. Properties (i)–(iv) follow directly from the definition of J' and J''. Property (v) follows from (iv) and the definition of J' by induction over the term structure. By (i) and (v), every ground BG term is interpreted in the same way in J and J'', moreover it is obvious that every ground BG term is interpreted in the same way in J'' and $J''|_{\Sigma_\mathrm{B}}$; this implies (vi). □

Lemma 8.6. *If J is a Σ-interpretation and* $I = \mathrm{nojunk}(J)$*, then for every ground term s there exists a ground term t such that $s^I = t^I$ and all BG-sorted (proper or non-proper) subterms of t are BG terms.*

Proof. If s has a BG sort ξ, then this follows directly from the fact that $s^I \in I_\xi$ and that every element of I_ξ equals t^I for some ground BG term t of sort ξ. If s has a FG sort, let $s_1, \ldots, s_n$ be the maximal BG-sorted subterms of $s = s[s_1, \ldots, s_n]$. Since for every s_i there is a ground BG term t_i with $s_i^I = t_i^I$, we obtain $s^I = (s[s_1, \ldots, s_n])^I = (s[t_1, \ldots, t_n])^I$. Set $t := s[t_1, \ldots, t_n]$. □

Corollary 8.7. *Let J be a Σ-interpretation and* $I = \mathrm{nojunk}(J)$*. Let $C\sigma$ by a ground instance of a clause C. Then there is a smooth ground instance $C\tau$ of C such that $(t\sigma)^I = (t\tau)^I$ for every term occurring in C and such that $I \models C\sigma$ if and only if $I \models C\tau$.*

Proof. Using the previous lemma, we define τ such that for every variable ζ occurring in C, $(\zeta\tau)^I = (\zeta\sigma)^I$ and all BG-sorted (proper or non-proper) subterms of $\zeta\tau$ are BG terms. Clearly τ is smooth. The other properties follow immediately by induction over the term or clause structure. □

Lemma 8.8. *Let N be a set of Σ-clauses that is locally sufficiently complete. Let $I \in \mathcal{B}$ be a Σ_B-interpretation, let J be a Σ-model of* $\mathrm{sgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I$*, and let* $J'' = \mathrm{nojunk}(J)$*. Let $C \in N$ and let $C\tau$ by a smooth ground instance in* $\mathrm{smgi}(N) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$*. Then $(t\tau)^J = (t\tau)^{J''}$ for every term t occurring in C and $J \models C\tau$ if and only if $J'' \models C\tau$.*

Proof. Let J' be defined as above, then $(t\tau)^{J'} = (t\tau)^{J''}$ for any term t occurring in C by Lemma 8.5-(i). We prove that $(t\tau)^J = (t\tau)^{J'}$ by induction over the term structure: If t is a variable, then by smoothness all BG-sorted subterms of $t\tau$ are BG terms, hence $(t\tau)^{J'} = (t\tau)^J$ by Lemma 8.5-(v). Otherwise let

$t = f(t_1, \ldots, t_n)$. If $t\tau$ is a BG term, then again $(t\tau)^{J'} = (t\tau)^J$ by Lemma 8.5-(v). If $t\tau$ is a FG term of sort ξ, then t must be a FG term of sort ξ as well. By the induction hypothesis, $(t_i\tau)^J = (t_i\tau)^{J'}$ for every i. If ξ is a FG sort, then trivially $(t\tau)^J = J_f((t_1\tau)^J, \ldots, (t_n\tau)^J)$ is contained in J'_ξ, so $(t\tau)^{J'} = (t\tau)^J$ by Lemma 8.5-(iv). Otherwise, $t\tau$ is a BG-sorted FG term occurring in $\mathrm{smgi}(N) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$. By local sufficient completeness, there exists a ground BG term s such that $s^J = (t\tau)^J$, hence $(t\tau)^J \in J'_\xi$. Again, Lemma 8.5-(iv) yields $(t\tau)^{J'} = (t\tau)^J$.

Since all left and right-hand sides of equations in $C\tau$ are evaluated in the same way in J'' and J, it follows that $J \models C\tau$ if and only if $J'' \models C\tau$. □

Lemma 8.9. *Let N be a set of Σ-clauses that is locally sufficiently complete. Let $I \in \mathcal{B}$ be a Σ_{B}-interpretation, let J be a Σ-model of* $\mathrm{sgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I$*, and let $J'' = \mathrm{nojunk}(J)$. Then J'' is a model of N.*

Proof. The proof proceeds in three steps. In the first step we show that J'' is a model of $\mathrm{smgi}(N) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$: Let $C \in N$ and let $C\tau$ be a smooth ground instance in $\mathrm{smgi}(N) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$. Since every smooth ground instance is a simple ground instance and J is a Σ-model of $\mathrm{sgi}(N)$, we know that $J \models C\tau$. By Lemma 8.8, this implies $J'' \models C\tau$.

In the second step we show that J'' is a model of $\mathrm{smgi}(N)$. Since we already know that J'' is a model of $\mathrm{smgi}(N) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$, it is clearly sufficient to show that J'' is a model of $\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$: First we observe that by Lemma 8.5 $J'' \models \mathrm{E}_I \cup \mathrm{D}_I$. Using the result of the first step, this implies that J'' is a model of $(\mathrm{smgi}(N) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)) \cup \mathrm{E}_I \cup \mathrm{D}_I$, and since this set is a superset of $(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$, J'' is also a model of the latter. By Definition 6.2-(i), $(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I) \models \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$. So J'' is a model of $\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$.

We can now show the main statement: We know that J'' is a term-generated Σ-interpretation, so $J'' \models N$ holds if and only if J'' is a model of all ground instances of clauses in N. Let $C\sigma$ be an arbitrary ground instance of $C \in N$. By Corollary 8.7, there is a smooth ground instance $C\tau$ such that $J'' \models C\sigma$ if and only if $J'' \models C\tau$. As the latter has been shown in the second step, the result follows. □

Theorem 8.10. *If the BG specification $(\Sigma_{\mathrm{B}}, \mathcal{B})$ is compact and if the clause set N is locally sufficiently complete, then* $\mathrm{HSP}_{\mathrm{Base}}$ *and $\mathcal{R}^{\mathcal{H}}$ are dynamically refutationally complete for* $\mathrm{abstr}(N)$*, i. e., if $N \models_{\mathcal{B}} \square$, then every fair derivation starting from* $\mathrm{abstr}(N)$ *eventually generates $\square$.*

Proof. Let $\mathcal{D} = (N_i)_{i \geq 0}$ be a fair derivation starting from $N_0 = \mathrm{abstr}(N)$, and let N_∞ be the limit of $\mathcal{D}$. By fairness, N_∞ is saturated w.r.t. $\mathrm{HSP}_{\mathrm{Base}}$ and $\mathcal{R}^{\mathcal{H}}$. If $\square \notin N_\infty$, then the Close rule is not applicable to N_∞. Since $(\Sigma_{\mathrm{B}}, \mathcal{B})$ is compact, this means that the set of all Σ_{B}-clauses in $\mathrm{sgi}(N_\infty)$ is satisfied by some term-generated Σ_{B}-interpretation $I \in \mathcal{B}$. By Theorem 7.8, $(N_\infty)_I$ is saturated with

respect to the standard superposition calculus. Since $\square \notin (N_\infty)_I$, the refutational completeness of standard superposition implies that there is a Σ-model J of $(N_\infty)_I$, and since $\mathrm{E}_I \cup \mathrm{D}_I \subseteq (N_\infty)_I$, J is also a Σ-model of $\mathrm{sgi}(N_\infty) \cup \mathrm{E}_I \cup \mathrm{D}_I$. Since every clause in N_0 is either contained in N_∞ or redundant w.r.t. N_∞, every simple ground instance of a clause in N_0 is a simple ground instance of a clause in N_∞ or contained in $\mathrm{GndTh}(\mathcal{B})$ or redundant w.r.t. $\mathrm{sgi}(N_\infty) \cup \mathrm{GndTh}(\mathcal{B})$. We conclude that J is a Σ-model of $\mathrm{sgi}(N_0)$, and since $\mathrm{sgi}(N_0)$ and $\mathrm{sgi}(N)$ are equivalent, J is a Σ-model of $\mathrm{sgi}(N)$. Now define $J'' = \mathrm{nojunk}(J)$. By Lemma 8.5, J'' is a term-generated Σ-interpretation, $J''|_{\Sigma_\mathrm{B}}$ is a term-generated Σ_B-interpretation, and $J''|_{\Sigma_\mathrm{B}}$ satisfies $\mathrm{E}_I \cup \mathrm{D}_I$. Consequently, $J''|_{\Sigma_\mathrm{B}}$ is isomorphic to I and thus contained in $\mathcal{B}$. Finally, J'' is a model of N by Lemma 8.9. □

If all BG-sorted FG terms in a set N of clauses are ground, local sufficient completeness can be established automatically by adding a "definition" of the form $t \approx \alpha$, where t is a ground BG-sorted FG term and α is a parameter. The following section explains this idea in a more general way.

9 Local Sufficient Completeness by Define

The $\mathrm{HSP}_\mathrm{Base}$ inference system will derive a contradiction if the input clause set is inconsistent and (locally) sufficiently complete (cf. Sect. 8). In this section we extend this functionality by adding an inference rule, Define, which can turn input clause sets that are not sufficiently complete into locally sufficiently complete ones. Technically, the Define rule derives "definitions" of the form $t \approx \alpha$, where t is a ground BG-sorted FG term and α is a parameter of the proper sort. For economy of reasoning, definitions are introduced only on a by-need basis, when t appears in a current clause, and $t \approx \alpha$ is used to simplify that clause immediately.

We need one more preliminary definition before introducing Define formally.

Definition 9.1 (Unabstracted clause). *A clause is* unabstracted *if it does not contain any disequation $\zeta \not\approx t$ between a variable ζ and a term t unless $t \neq \zeta$ and $\zeta \in \mathrm{vars}(t)$.* □

Any clause can be unabstracted by repeatedly replacing $C \vee \zeta \not\approx t$ by $C[\zeta \mapsto t]$ whenever $t = \zeta$ or $\zeta \notin \mathrm{vars}(t)$. Let $\mathrm{unabstr}(C)$ denote an unabstracted version of C obtained this way. If $t = t[\zeta_1, \ldots, \zeta_n]$ is a term in C and ζ_i is finally instantiated to t_i, we denote its unabstracted version $t[t_1, \ldots, t_n]$ by $\mathrm{unabstr}(t[\zeta_1, \ldots, \zeta_n], C)$. For a clause set N let $\mathrm{unabstr}(N) = \{\mathrm{unabstr}(C) \mid C \in N\}$.

The *full inference system* HSP of the hierarchic superposition calculus consists of the inference rules of $\mathrm{HSP}_\mathrm{Base}$ and the following Define inference rule. As for the other inference rules we suppose that all premises are weakly abstracted.

$$\textsf{Define}\ \frac{N \cup \{L[t[\zeta_1,\ldots,\zeta_n]] \vee D\}}{N \cup \text{abstr}(\{t[t_1,\ldots,t_n] \approx \alpha_{t[t_1,\ldots,t_n]},\ L[\alpha_{t[t_1,\ldots,t_n]}] \vee D\}}$$

if

(i) $t[\zeta_1,\ldots,\zeta_n]$ is a minimal BG-sorted non-variable term with a toplevel FG operator,
(ii) $t[t_1,\ldots,t_n] = \text{unabstr}(\{t[\zeta_1,\ldots,\zeta_n], L[t[\zeta_1,\ldots,\zeta_n]] \vee D\})$,
(iii) $t[t_1,\ldots,t_n]$ is ground,
(iv) $\alpha_{t[t_1,\ldots,t_n]}$ is a parameter, uniquely determined by $t[t_1,\ldots,t_n]$, and
(v) $L[t[\zeta_1,\ldots,\zeta_n]] \vee D \in \mathcal{R}^{\mathcal{H}}_{\text{Cl}}(N \cup \text{abstr}(\{t[t_1,\ldots,t_n] \approx \alpha_{t[t_1,\ldots,t_n]}, L[\alpha_{t[t_1,\ldots,t_n]}] \vee D\}))$.

In (i), by minimality we mean that no proper subterm of $t[\zeta_1,\ldots,\zeta_n]$ is a BG-sorted non-variable term with a toplevel FG operator. In effect, the Define rule eliminates such terms inside-out. Conditions (iii) and (iv) are needed for soundness. Condition (v) is needed to guarantee that Define is a simplifying inference rule, much like the Simp rule in Sect. 7.[12] In particular, it makes sure that Define cannot be applied to definitions themselves.

Theorem 9.2. *The inference rules of* HSP *are satisfiability-preserving w.r.t.* $\models_{\mathcal{B}}$*, i. e., for every inference with premise* N *and conclusion* N' *we have* $N \models_{\mathcal{B}} \square$ *if and only if* $N' \models_{\mathcal{B}} \square$*. Moreover,* $N' \models_{\mathcal{B}} N$*.*

Proof. For the inference rules of HSP_{Base}, the result follows from Theorem 6.1.

For Define, we observe first that condition (ii) implies that $L[t[\zeta_1,\ldots,\zeta_n]] \vee D$ and $L[t[t_1,\ldots,t_n]] \vee D$ are equivalent. If $N \cup \{L[t[t_1,\ldots,t_n]] \vee D\}$ is $\mathcal{B}$-satisfiable, let I be a Σ-model of all ground instances of $N \cup \{L[t[t_1,\ldots,t_n]] \vee D\}$ such that $I|_{\Sigma_{\text{B}}}$ is in $\mathcal{B}$. By condition (iii), $t[t_1,\ldots,t_n]$ is ground. Let J be the Σ-interpretation obtained from J by redefining the interpretation of $\alpha_{t[t_1,\ldots,t_n]}$ in such a way that $\alpha^J_{t[t_1,\ldots,t_n]} = t[t_1,\ldots,t_n]^I$, then J is a Σ-model of every ground instance of N, $t[t_1,\ldots,t_n] \approx \alpha_{t[t_1,\ldots,t_n]}$ and $L[\alpha_{t[t_1,\ldots,t_n]}] \vee D$, and hence also a model of the abstractions of these clauses. Conversely, every model of $t[t_1,\ldots,t_n] \approx \alpha_{t[t_1,\ldots,t_n]}$ and $L[\alpha_{t[t_1,\ldots,t_n]}] \vee D$ is a model of $L[t[t_1,\ldots,t_n]] \vee D$. $\square$

Example 9.3. Let $C = \mathsf{g}(\mathsf{f}(x,y)+1,x,y) \approx 1 \vee x \not\approx 1+\beta \vee y \not\approx \mathsf{c}$ be the premise of a Define inference. We get $\text{unabstr}(C) = \mathsf{g}(\mathsf{f}(1+\beta,\mathsf{c})+1,1+\beta,\mathsf{c}) \approx 1$. The (unabstracted) conclusions are the definition $\mathsf{f}(1+\beta,\mathsf{c}) \approx \alpha_{\mathsf{f}(1+\beta,\mathsf{c})}$ and the clause $\mathsf{g}(\alpha_{\mathsf{f}(1+\beta,\mathsf{c})}+1,x,y) \approx 1 \vee x \not\approx 1+\beta \vee y \not\approx \mathsf{c}$. Abstraction yields $\mathsf{f}(X,\mathsf{c}) \approx \alpha_{\mathsf{f}(1+\beta,\mathsf{c})} \vee X \not\approx 1+\beta$ and $\mathsf{g}(Z,x,y) \approx 1 \vee x \not\approx 1+\beta \vee y \not\approx \mathsf{c} \vee Z \not\approx \alpha_{\mathsf{f}(1+\beta,\mathsf{c})}+1$.

One might be tempted to first unabstract the premise C before applying Define. However, unabstraction may eliminate FG terms (c in the example) which is not undone by abstraction. This may lead to incompleteness. $\square$

[12] Condition (i) of Simp is obviously satisfied and condition (iii) there is condition (v) of Define. Instead of condition (ii), Define inferences are only $\mathcal{B}$-satisfiability preserving, which however does not endanger soundness.

Example 9.4. The following clause set demonstrates the need for condition (v) in Define. Let $N = \{\mathsf{f}(\mathsf{c}) \approx 1\}$ and suppose condition (v) is absent. Then we obtain $N' = \{\mathsf{f}(\mathsf{c}) \approx \alpha_{\mathsf{f}(\mathsf{c})},\ \alpha_{\mathsf{f}(\mathsf{c})} \approx 1\}$. By demodulating the first clause with the second clause we get $N'' = \{\mathsf{f}(\mathsf{c}) \approx 1,\ \alpha_{\mathsf{f}(\mathsf{c})} \approx 1\}$. Now we can continue with N'' as with N. The problem is, of course, that the new definition $\mathsf{f}(\mathsf{c}) \approx \alpha_{\mathsf{f}(\mathsf{c})}$ is greater w.r.t. the term ordering than the parent clause, in violation of condition (v). □

Example 9.5. Consider the weakly abstracted clauses $\mathsf{P}(0)$, $\mathsf{f}(x) > 0 \vee \neg\mathsf{P}(x)$, $\mathsf{Q}(\mathsf{f}(x))$, $\neg\mathsf{Q}(x) \vee 0 > x$. Suppose $\neg\mathsf{P}(x)$ is maximal in the second clause. By superposition between the first two clauses we derive $\mathsf{f}(0) > 0$. With Define we obtain $\mathsf{f}(0) \approx \alpha_{\mathsf{f}(0)}$ and $\alpha_{\mathsf{f}(0)} > 0$, the latter replacing $\mathsf{f}(0) > 0$. From the third clause and $\mathsf{f}(0) \approx \alpha_{\mathsf{f}(0)}$ we obtain $\mathsf{Q}(\alpha_{\mathsf{f}(0)})$, and with the fourth clause $0 > \alpha_{\mathsf{f}(0)}$. Finally we apply Close to $\{\alpha_{\mathsf{f}(0)} > 0,\ 0 > \alpha_{\mathsf{f}(0)}\}$. □

It is easy to generalize Theorem 8.10 to the case that local sufficient completeness does not hold initially, but is only established with the help of Define inferences:

Theorem 9.6. *Let $\mathcal{D} = (N_i)_{i\geq 0}$ be a fair* HSP *derivation starting from $N_0 = \mathrm{abstr}(N)$, let $k \geq 0$, such that $N_k = \mathrm{abstr}(N')$ and N' is locally sufficiently complete. If the BG specification $(\Sigma_{\mathrm{B}}, \mathcal{B})$ is compact, then the limit of $\mathcal{D}$ contains $\square$ if and only if N is $\mathcal{B}$-unsatisfiable.*

Proof. Since every derivation step in an HSP derivation is satisfiability-preserving, the "only if" part is again obvious.

For the "if" part, we assume that N_∞, the limit of $\mathcal{D}$, does not contain $\square$. By fairness, N_∞ is saturated w.r.t. HSP and $\mathcal{R}^{\mathcal{H}}$. We start by considering the subderivation $(N_i)_{i\geq k}$ starting with $N_k = \mathrm{abstr}(N')$. Like in the proof of Theorem 8.10, we can show that N' is $\mathcal{B}$-satisfiable, that is, there exists a model J of N' that is a term-generated Σ-interpretation, and whose restriction $J|_{\Sigma_{\mathrm{B}}}$ is contained in $\mathcal{B}$. From Lemma 7.1 and Proposition 5.2 we see that $N' \models_{\mathcal{B}} N_k$, and similarly $N_0 \models_{\mathcal{B}} N$. Furthermore, since every clause in $N_0 \setminus N_k$ must be redundant w.r.t. N_k, we have $N_k \models_{\mathcal{B}} N_0$. Combining these three entailments, we conclude that $N' \models_{\mathcal{B}} N$, so N is $\mathcal{B}$-satisfiable and J is a model of N. □

Condition (v) of the Define rule requires that the clause that is deleted during a Define inference must be redundant with respect to the remaining clauses. This condition is needed to preserve refutational completeness. There are cases, however, where condition (v) prevents us from introducing a definition for a subterm. Consider the clause set $N = \{C\}$ where $C = \mathsf{f}(\mathsf{c}) \approx 1 \vee \mathsf{c} \approx \mathsf{d}$, the constants c and d are FG-sorted, f is a BG-sorted FG operator, and $\mathsf{c} \succ \mathsf{d} \succ 1$. The literal $\mathsf{f}(\mathsf{c}) \approx 1$ is maximal in C. The clause set $N = \mathrm{abstr}(N)$ is not locally sufficient complete (the BG-sorted FG-term $\mathsf{f}(\mathsf{c})$ may be interpreted differently from all BG terms in a Σ-model). Moreover, it cannot be made locally sufficient complete using the Define rule, since the definition $\mathsf{f}(\mathsf{c}) \approx \alpha_{\mathsf{f}(\mathsf{c})}$ is larger w.r.t. the clause ordering than C, in violation of condition (v) of Define.

However, at the beginning of a derivation, we may be a bit more permissive. Let us define the *reckless* Define inference rule in the same way as the Define rule

except that the applicability condition (v) is dropped. Clearly, in the example above, the reckless Define rule allows us to derive the locally sufficiently complete clause set $N' = \{\alpha_{\mathsf{f(c)}} \approx 1 \vee \mathsf{c} \approx \mathsf{d},\ \mathsf{f(c)} \approx \alpha_{\mathsf{f(c)}}\}$ as desired. In fact, we can show that this is always possible if N is a finite clause set in which all BG-sorted FG terms are ground.

Definition 9.7 (Pre-derivation). *Let N_0 be a weakly abstracted clause set. A* pre-derivation (of a clause set N^{pre}) *is a derivation of the form $N_0, N_1, \ldots, (N_k = N^{\mathrm{pre}})$, for some $k \geq 0$, with the inference rule reckless* Define *only, and such that each clause $C \in N^{\mathrm{pre}}$ either does not contain any BG-sorted FG operator or $C = \mathrm{abstr}(C')$ and C' is a* definition*, i. e., a ground positive unit clause of the form $\mathsf{f}(t_1, \ldots, t_n) \approx t$ where f is a BG-sorted FG operator, $t_1, \ldots, t_n$ do not contain BG-sorted FG operators, and t is a background term.* □

Lemma 9.8. *Let N be a finite clause set in which all BG-sorted FG terms are ground. Then there is a pre-derivation starting from $N_0 = \mathrm{abstr}(N)$ such that N^{pre} is locally sufficiently complete.*

Proof. Since every term headed by a BG-sorted FG operator in $\mathrm{unabstr}(N_0)$ is ground, we can incrementally eliminate all occurrences of terms headed by BG-sorted FG operators from N_0, except those in abstractions of definitions. Let $N_0, N_1, \ldots, (N_k = N^{\mathrm{pre}})$ be the sequence of sets of clauses obtained in this way. We will show that N^{pre} is locally sufficiently complete.

Let $I \in \mathcal{B}$ be a Σ_{B}-interpretation, let J be a Σ-model of $\mathrm{sgi}(N^{\mathrm{pre}}) \cup \mathrm{E}_I \cup \mathrm{D}_I$ and let $C\theta$ be a smooth ground instance in $\mathrm{smgi}(N) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$. We have to show that for every BG-sorted FG term s occurring in $C\theta$ there is a ground BG term t such that $J \models s \approx t$.

If C does not contain any BG-sorted FG operator, then there are no BG-sorted FG terms in $C\theta$, so the property is vacuously true. Otherwise $C = \mathrm{abstr}(C')$ and C' is a definition $\mathsf{f}(t_1, \ldots, t_n) \approx t$ where f is a BG-sorted FG operator, $t_1, \ldots, t_n$ do not contain BG-sorted FG operators, and t is a background term. In this case, C must have the form $\mathsf{f}(u_1, \ldots, u_n) \approx u \vee E$, such that E is a BG clause, $u_1, \ldots, u_n$ do not contain BG-sorted FG operators, and u is a BG term. The only BG-sorted FG term in the smooth instance $C\theta$ is therefore $\mathsf{f}(u_1\theta, \ldots, u_n\theta)$. If any literal of $E\theta$ were true in J, then it would follow from $\mathrm{E}_I \cup \mathrm{D}_I$, therefore $C\theta \in \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{smgi}(N) \cup \mathrm{E}_I \cup \mathrm{D}_I)$, contradicting the assumption. Hence $J \models \mathsf{f}(u_1\theta, \ldots, u_n\theta) \approx u\theta$, and since $u\theta$ is a ground BG term, the requirement is satisfied. □

Lemma 9.8 will be needed to prove a completeness result for the fragment defined in the next section.

10 The Ground BG-Sorted Term Fragment

According to Theorem 8.10, the $\mathrm{HSP}_{\mathrm{Base}}$ calculus is refutationally complete provided that the clause set is locally sufficiently complete and the BG specification

is compact. We have seen in the previous section that the (reckless) Define rule can help to establish local sufficient completeness by introducing new parameters. In fact, finite clause sets in which all BG-sorted FG terms are ground can always be converted into locally sufficiently complete clause sets (cf. Lemma 9.8). On the other hand, as noticed in Sect. 3, the introduction of parameters can destroy the compactness of the BG specification. In this and the following section, we will identify two cases where we can not only establish local sufficient completeness, but where we can also guarantee that compactness poses no problems. The *ground BG-sorted term fragment (GBT fragment)* is one such case:

Definition 10.1 (GBT fragment). *A clause C is a GBT clause if all BG-sorted terms in C are ground. A finite clause set N belongs to the GBT fragment if all clauses in N are GBT clauses.* □

Clearly, by Lemma 9.8 for every clause set N that belongs to the GBT fragment there is a pre-derivation that converts $\mathrm{abstr}(N)$ into a locally sufficiently complete clause set. Moreover, pre-derivations also preserve the GBT property:

Lemma 10.2. *If* $\mathrm{unabstr}(N)$ *belongs to the GBT fragment and N' is obtained from N by a reckless* Define *inference, then* $\mathrm{unabstr}(N')$ *also belongs to the GBT fragment.*

The proof can be found in [12].

As we have seen, N^{pre} is locally sufficiently complete. At this stage this suggests to exploit the completeness result for locally sufficiently complete clause sets, Theorem 8.10. However, Theorem 8.10 requires compact BG specifications, and the question is if we can avoid this. We can indeed get a complete calculus under rather mild assumptions on the Simp rule:

Definition 10.3 (Suitable Simp inference). *Let $\succ_{\mathrm{fin}}$ be a strict partial term ordering such that for every ground BG term s only finitely many ground BG terms t with $s \succ_{\mathrm{fin}} t$ exist.*[13] *We say that a* Simp *inference with premise $N \cup \{C\}$ and conclusion $N \cup \{D\}$ is* suitable (for the GBT fragment) *if*

(i) *for every BG term t occurring in* $\mathrm{unabstr}(D)$ *there is a BG term $s \in \mathrm{unabstr}(C)$ such that $s \succeq_{\mathrm{fin}} t$,*
(ii) *every occurrence of a BG-sorted FG operator* f *in* $\mathrm{unabstr}(D)$ *is of the form $\mathsf{f}(t_1, \ldots, t_n) \approx t$ where t is a ground BG term,*
(iii) *every BG term in D is pure, and*
(iv) *if every BG term in* $\mathrm{unabstr}(C)$ *is ground then every BG term in* $\mathrm{unabstr}(D)$ *is ground.*

We say the Simp *inference rule is* suitable *if every* Simp *inference is.* □

Expected simplification techniques like demodulation, subsumption deletion and evaluation of BG subterms are all covered as suitable Simp inferences. Also, evaluation of BG subterms is possible, because simplifications are not only decreasing

[13] A KBO with appropriate weights can be used for $\succ_{\mathrm{fin}}$.

w.r.t. $\succ$ but *additionally* also decreasing w.r.t. $\succeq_{\text{fin}}$, as expressed in condition (i). Without it, e. g., the clause $\mathsf{P}(1 + 1, 0)$ would admit infinitely many simplified versions $\mathsf{P}(2, 0)$, $\mathsf{P}(2, 0 + 0)$, $\mathsf{P}(2, 0 + (0 + 0))$, etc.

The HSP_{Base} inferences do in general not preserve the shape of the clauses in $\text{unabstr}(N^{\text{pre}})$; they do preserve a somewhat weaker property – cleanness – which is sufficient for our purposes.

Definition 10.4 (Clean clause). *A weakly abstracted clause C is* clean *if*

(i) *every BG term in C is pure,*
(ii) *every BG term in* $\text{unabstr}(C)$ *is ground, and*
(iii) *every occurrence of a BG-sorted FG operator* f *in* $\text{unabstr}(C)$ *is in a positive literal of the form* $\mathsf{f}(t_1, \ldots, t_n) \approx t$ *where t is a ground BG term.*

For example, if c is FG-sorted, then $\mathsf{P}(\mathsf{f}(\mathsf{c}) + 1)$ is not clean, while $\mathsf{f}(x) \approx 1 + \alpha \vee \mathsf{P}(x)$ is. A clause set is called *clean* if every clause in N is. Notice that N^{pre} is clean.

Lemma 10.5. *Let $C_1, \ldots, C_n$ be clean clauses. Assume a* HSP_{Base} *inference with premises $C_1, \ldots, C_n$ and conclusion C. Then C is clean and every BG term occurring in* $\text{unabstr}(C)$ *also occurs in some clause* $\text{unabstr}(C_1), \ldots,$ $\text{unabstr}(C_n)$.

The proof can be found in [12].

Thanks to conditions (ii)–(iv) in Definition 10.3, suitable Simp inferences preserves cleanness:

Lemma 10.6. *Let $N \cup \{C\}$ be a set of clean clauses. If $N \cup \{D\}$ is obtained from $N \cup \{C\}$ by a suitable* Simp *inference then D is clean.*

Proof. Suppose $N \cup \{D\}$ is obtained from $N \cup \{C\}$ by a suitable Simp inference. We need to show properties (i)–(iii) of cleanness for D. That every BG term in D is pure follows from Definition 10.3-(iii). That every BG term in $\text{unabstr}(D)$ is ground follows from Definition 10.3-(iv) and cleanness of C. Finally, property (iii) follows from Definition 10.3-(ii). □

With the above lemmas we can prove our main result:

Theorem 10.7. *The* HSP *calculus with a suitable* Simp *inference rule is dynamically refutationally complete for the ground BG-sorted term fragment. More precisely, let N be a finite set of GBT clauses and $\mathcal{D} = (N_i)_{i \geq 0}$ a fair* HSP *derivation such that reckless* Define *is applied only in a pre-derivation* $(N_0 = \text{abstr}(N)), \ldots, (N_k = N^{\text{pre}})$, *for some $k \geq 0$. Then the limit of $\mathcal{D}$ contains $\square$ if and only if N is $\mathcal{B}$-unsatisfiable.*

Notice that Theorem 10.7 does not appeal to compactness of BG specifications.

Proof. Our goal is to apply Theorem 9.6 and its proof, in a slightly modified way. For that, we first need to know that $N^{\text{pre}} = \text{abstr}(N')$ for some clause set N' that is locally sufficiently complete.

We are given that N is a set of GBT clauses. Recall that weak abstraction (recursively) extracts BG subterms by substituting fresh variables and adding disequations. Unabstraction reverses this process (and possibly eliminates additional disequations). It follows that with N being a set of GBT clauses, so is $\mathrm{unabstr}(\mathrm{abstr}(N)) = \mathrm{unabstr}(N_0)$. From Lemma 10.2 it follows that $\mathrm{unabstr}(N^{\mathrm{pre}})$ is also a GBT clause set.

Now chose N' as the clause set that is obtained from N^{pre} by replacing every clause $C \in N^{\mathrm{pre}}$ such that $\mathrm{unabstr}(C)$ is a definition by $\mathrm{unabstr}(C)$. By construction of definitions, unabstraction reverses weak abstraction of definitions. It follows $N^{\mathrm{pre}} = \mathrm{abstr}(N')$. By definition of pre-derivations, all BG-sorted FG terms occurring in $\mathrm{unabstr}(N^{\mathrm{pre}})$ occur in definitions. Hence, with $\mathrm{unabstr}(N^{\mathrm{pre}})$ being a set of GBT clauses so is N'. It follows easily that N' is locally sufficiently complete, as desired.

We cannot apply Theorem 9.6 directly now because it requires compactness of the BG specification, which cannot be assumed. However, we can use the following argumentation instead.

Let $N^\infty = \bigcup_{i \geq 0} N_i$ be the union of $\mathcal{D}$. We next show that $\mathrm{unabstr}(N^\infty)$ contains only finitely many different BG terms and each of them is ground. Recall that $\mathrm{unabstr}(N^{\mathrm{pre}})$ is a GBT clause set, and so every BG term in $\mathrm{unabstr}(N^{\mathrm{pre}})$ is ground. Because Define is disabled in $\mathcal{D}$, only $\mathrm{HSP}_{\mathrm{Base}}$ and (suitable) Simp inferences need to be analysed. Notice that N^{pre} is clean and both the $\mathrm{HSP}_{\mathrm{Base}}$ and Simp inferences preserve cleanness, as per Lemmas 10.5-(1) and 10.6, respectively.

With respect to $\mathrm{HSP}_{\mathrm{Base}}$ inferences, together with Definition 10.4-(ii) it follows that every BG term t in the unabstracted version $\mathrm{unabstr}(C)$ of the inference conclusion C is ground. Moreover, t also occurs in the unabstracted version of some premise clause by Lemma 10.5-(2). In other words, $\mathrm{HSP}_{\mathrm{Base}}$ inferences do not grow the set of BG terms w.r.t. unabstracted premises and conclusions.

With respect to Simp inferences, $\mathrm{unabstr}(N^{\mathrm{pre}})$ provide an upper bound w.r.t. the term ordering $\succ_{\mathrm{fin}}$ for all BG terms generated in Simp inferences. There can be only finitely many such terms, and each of them is ground, which follows from Definition 10.3-(i).

Because every BG term occurring in $\mathrm{unabstr}(N^\infty)$ is ground, every BG clause in $\mathrm{unabstr}(N^\infty)$ is a multiset of literals of the form $s \approx t$ or $s \not\approx t$, where s and t are ground BG terms. With only finitely many BG terms available, there are only finitely many BG clauses in $\mathrm{unabstr}(N^\infty)$, modulo equivalence. Because unabstraction is an equivalence transformation, there are only finitely many BG clauses in N^∞ as well, modulo equivalence.

Let $N_\infty = \bigcup_{i \geq 0} \bigcap_{j \geq i} N_j$ be the limit clause set of the derivation $\mathcal{D}$, which is saturated w.r.t. the hierarchic superposition calculus and $\mathcal{R}^{\mathcal{H}}$. Because $\mathcal{D}$ is not a refutation, it does not contain $\square$. Consequently the Close rule is not applicable to N_∞. The set N^∞, and hence also $N_\infty \subseteq N^\infty$, contains only finitely many BG clauses, modulo equivalence. This entails that the set of all Σ_{B}-clauses in $\mathrm{sgi}(N_\infty)$ is satisfied by some term-generated Σ_{B}-interpretation $I \in \mathcal{B}$. Now, the rest of the proof is literally the same as in the proof of Theorem 9.6. $\square$

Because unabstraction can also be applied to fully abstracted clauses, it is possible to equip the hierarchic superposition calculus of [5] with a correspondingly modified Define rule and get Theorem 10.7 in that context as well.

Kruglov and Weidenbach [22] have shown how to use hierarchic superposition as a decision procedure for ground clause sets (and for Horn clause sets with constants and variables as the only FG terms). Their method preprocesses the given clause set by "basification", a process that removes BG-sorted FG terms similarly to our reckless Define rule. The resulting clauses then are fully abstracted and hierarchic superposition is applied. Some modifications of the inference rules make sure derivations always terminate. Simplification is restricted to subsumption deletion. The fragment of [22] is a further restriction of the GBT fragment. We expect we can get decidability results for that fragment with similar techniques.

11 Linear Arithmetic

For the special cases of linear integer arithmetic (LIA) and linear rational arithmetic as BG specifications, the result of the previous section can be extended significantly: In addition to ground BG-sorted terms, we can also permit BG-sorted variables and, in certain positions, even variables with offsets.

Recall that we have assumed that equality is the only predicate symbol in our language, so that a non-equational atom, say $s < t$, is to be taken as a shorthand for the equation $(s < t) \approx \mathit{true}$. We refer to the terms that result from this encoding of atoms as *atom terms*; other terms are called *proper terms*.

Theorem 11.1. *Let N be a set of clauses over the signature of linear integer arithmetic (with parameters α, β, etc.), such that every proper term in these clauses is either (i) ground, or (ii) a variable, or (iii) a sum $\zeta + k$ of a variable ζ and a number $k \geq 0$ that occurs on the right-hand side of a positive literal $s < \zeta + k$. If the set of ground terms occurring in N is finite, then N is satisfiable in LIA over $\mathbb{Z}$ if and only if N is satisfiable w.r.t.the first-order theory of LIA.*

Proof. Let N be a set of clauses with the required properties, and let T be the finite set of ground terms occurring in N. We will show that N is equivalent to some *finite* set of clauses over the signature of linear integer arithmetic, which implies that it is satisfiable in the integer numbers if and only if it is satisfiable in the first-order theory of LIA.

In a first step, we replace every negative ordering literal $\neg s < t$ or $\neg s \leq t$ by the equivalent positive ordering literal $t \leq s$ or $t < s$. All literals of clauses in the resulting set N_0 have the form $s \approx t$, $s \not\approx t$, $s < t$, $s \leq t$, or $s < \zeta + k$, where s and t are either variables or elements of T and $k \in \mathbb{N}$. Note that the number of variables in clauses in N_0 may be unbounded.

In order to handle the various inequality literals in a more uniform way, we introduce new binary relation symbols $<_k$ (for $k \in \mathbb{N}$) that are defined by $a <_k b$ if and only if $a < b + k$. Observe that $s <_k t$ entails $s <_n t$ whenever $k \leq n$.

Obviously, we may replace every literal $s < t$ by $s <_0 t$, every literal $s \leq t$ by $s <_1 t$, and every literal $s < \zeta + k$ by $s <_k \zeta$. Let N_1 be the resulting clause set.

We will now transform N_1 into an equivalent set N_2 of ground clauses. We start by eliminating all equality literals that contain variables by exhaustively applying the following transformation rules:

$$\begin{array}{lcll} N \cup \{ C \vee \zeta \not\approx \zeta \} & \to & N \cup \{ C \} & \\ N \cup \{ C \vee \zeta \not\approx t \} & \to & N \cup \{ C[\zeta \mapsto t] \} & \text{if } t \neq \zeta \\ N \cup \{ C \vee \zeta \approx \zeta \} & \to & N & \\ N \cup \{ C \vee \zeta \approx t \} & \to & N \cup \{ C \vee \zeta <_1 t, C \vee t <_1 \zeta \} & \text{if } t \neq \zeta \end{array}$$

All variables in inequality literals are then eliminated in a Fourier-Motzkin-like manner by exhaustively applying the transformation rule

$$N \cup \{ C \vee \bigvee_{i \in I} \zeta <_{k_i} s_i \vee \bigvee_{j \in J} t_j <_{n_j} \zeta \} \quad \to \quad N \cup \{ C \vee \bigvee_{i \in I} \bigvee_{j \in J} t_j <_{k_i + n_j} s_i \}$$

where ζ does not occur in C and one of the index sets I and J may be empty.

The clauses in N_2 are constructed over the finite set T of proper ground terms, but the length of the clauses in N_2 is potentially unbounded. In the next step, we will transform the clauses in such a way that any pair of terms s, t from T is related by at most one literal in any clause: We apply one of the following transformation rules as long as two terms s and t occur in more than one literal:

$$\begin{array}{lcll} N \cup \{ C \vee s <_k t \vee s \approx t \} & \to & N \cup \{ C \vee s <_k t \} & \text{if } k \geq 1 \\ N \cup \{ C \vee s <_0 t \vee s \approx t \} & \to & N \cup \{ C \vee s <_1 t \} & \\ N \cup \{ C \vee s <_k t \vee s \not\approx t \} & \to & N & \text{if } k \geq 1 \\ N \cup \{ C \vee s <_0 t \vee s \not\approx t \} & \to & N \cup \{ C \vee s \not\approx t \} & \\ N \cup \{ C \vee s <_k t \vee s <_n t \} & \to & N \cup \{ C \vee s <_n t \} & \text{if } k \leq n \\ N \cup \{ C \vee s <_k t \vee t <_n s \} & \to & N & \text{if } k + n \geq 1 \\ N \cup \{ C \vee s <_0 t \vee t <_0 s \} & \to & N \cup \{ C \vee s \not\approx t \} & \\ N \cup \{ C \vee L \vee L \} & \to & N \cup \{ C \vee L \} & \text{for any literal } L \\ N \cup \{ C \vee s \approx t \vee s \not\approx t \} & \to & N & \end{array}$$

The length of the clauses in the resulting set N_3 is now bounded by $\frac{1}{2}m(m+1)$, where m is the cardinality of T. Still, due to the indices of the relation symbols $<_k$, N_3 may be infinite. We introduce an equivalence relation $\sim$ on clauses in N_3 as follows: Define $C \sim C'$ if for all $s, t \in T$ (i) $s \approx t \in C$ if and only if $s \approx t \in C'$, (ii) $s \not\approx t \in C$ if and only if $s \not\approx t \in C'$, and (iii) $s <_k t \in C$ for some k if and only if $s <_n t \in C'$ for some n. This relation splits N_3 into at most $(\frac{1}{2}m(m+1))^5$ equivalence classes.[14]

We will now show that each equivalence class is logically equivalent to a finite subset of itself. Let M be some equivalence class. Since any two clauses from

[14] Any pair of terms s, t is related in all clauses of an equivalence class by either a literal $s \approx t$, or $s \not\approx t$, or $s <_n t$ for some n, or $t <_n s$ for some n, or no literal at all, so there are five possibilities per unordered pair of terms.

M differ at most in the indices of their $<_k$-literals, we can write every clause $C_i \in M$ in the form

$$C_i \quad = \quad C \vee \bigvee_{1 \leq l \leq n} s_l <_{k_{il}} t_l$$

where C and the s_l and t_l are the same for all clauses in M. As we have mentioned above, $s_l <_{k_{il}} t_l$ entails $s_l <_{k_{jl}} t_l$ whenever $k_{il} \leq k_{jl}$; so a clause $C_i \in M$ entails $C_j \in M$ whenever the n-tuple $(k_{i1}, \ldots, k_{in})$ is pointwise smaller or equal to the n-tuple $(k_{j1}, \ldots, k_{jn})$ (that is, $k_{il} \leq k_{jl}$ for all $1 \leq l \leq n$).

Let Q be the set of n-tuples of natural numbers corresponding to the clauses in M. By Dickson's lemma [15], for every set of tuples in $\mathbb{N}^n$ the subset of minimal tuples (w.r.t.the pointwise extension of $\leq$ to tuples) is finite. Let Q' be the subset of minimal tuples in Q, and let M' be the set of clauses in M that correspond to the tuples in Q'. Since for every tuple in $Q \setminus Q'$ there is a smaller tuple in Q', we know that every clause in $M \setminus M'$ is entailed by some clause in M'. So the equivalence class M is logically equivalent to its finite subset M'. Since the number of equivalence classes is also finite and all transformation rules are sound, this proves our claim. □

In order to apply this theorem to hierarchic superposition, we must again impose some restrictions on the calculus. Most important, we have to change the definition of weak abstraction slightly: We drop the requirement that target terms are not domain elements from Definition 5.1, i. e., we abstract out a non-variable BG term q occurring in a clause $C[f(s_1, \ldots, q, \ldots, s_n)]$, where f is a FG operator or at least one of the s_i is a FG or impure BG term, even if q is a domain element. As we mentioned, all results obtained so far hold also for the modified definition of weak abstraction. In addition, we must again restrict to *suitable* Simp inferences (Definition 10.3). With these restrictions, we can prove our main result:

Theorem 11.2. *The hierarchic superposition calculus is dynamically refutationally complete w.r.t. LIA over $\mathbb{Z}$ for finite sets of Σ-clauses in which every proper BG-sorted term is either (i) ground, or (ii) a variable, or (iii) a sum $\zeta + k$ of a variable ζ and a number $k \geq 0$ that occurs on the right-hand side of a positive literal $s < \zeta + k$.*

Proof. Let N be a finite set of Σ-clauses with the required properties. By Lemma 9.8, a pre-derivation starting from $N_0 = \text{abstr}(N)$ yields a locally sufficiently complete finite set N_0 of abstracted clauses.

Now we run the hierarchic superposition calculus on N_0 (with the same restrictions on simplifications as in Sect. 10). Let N_1 be the (possibly infinite) set of BG clauses generated during the run. By unabstracting these clauses, we obtain an equivalent set N_2 of clauses that satisfy the conditions of Theorem 11.1, so N_2 is satisfiable in LIA over $\mathbb{Z}$ if and only if N is satisfiable w.r.t.the first-order theory of LIA. Since the hierarchic superposition calculus is dynamically refutationally complete w.r.t. the first-order theory of LIA, the result follows. □

Analogous results hold for linear rational arithmetic. Let n be the least common divisor of all numerical constants in the original clause set; then we define $a <_{2i} b$ by $a < b + \frac{i}{n}$ and $a <_{2i+1} b$ by $a \leq b + \frac{i}{n}$ for $i \in \mathbb{N}$ and express every inequation literal in terms of $<_k$. The Fourier-Motzkin transformation rule is replaced by

$$N \cup \{ C \vee \bigvee_{i \in I} \zeta <_{k_i} s_i \vee \bigvee_{j \in J} t_j <_{n_j} \zeta \} \quad \rightarrow \quad N \cup \{ C \vee \bigvee_{i \in I} \bigvee_{j \in J} t_j <_{k_i \bullet n_j} s_i \}$$

where ζ does not occur in C, one of the index sets I and J may be empty, and $k \bullet n$ is defined as $k + n - 1$ if both k and n are odd, and $k + n$ otherwise. The rest of the proof proceeds in the same way as before.

12 Experiments

We implemented the HSP calculus in the theorem prover *Beagle*.[15] *Beagle* is a testbed for rapidly trying out theoretical ideas but it is not a high-performance prover (in particular it lacks indexing of any form). The perhaps most significant calculus feature not yet implemented is the improvement for linear integer and rational arithmetic of Sect. 11.

Beagle's proof procedure and background reasoning, in particular for linear integer arithmetic, and experimental results have been described in [7]. Here we only provide an update on the experiments and report on complementary aspects not discussed in [7]. More specifically, our new experiments are based on a more recent version of the TPTP problem library [27] (by four years), and we discuss in more detail the impact of the various calculus variants introduced in this paper. We also compare *Beagle*'s performance to that of other provers.

We tested *Beagle* on the first-order problems from the TPTP library, version 7.2.0,[16] that involve some form of arithmetic, including non-linear, rational and real arithmetics. The problems in the TPTP are organized in categories, and the results for some of them are quickly dealt with: none of the HWV-problems in the TPTP library was solvable within the time limit and we ignore these below. We ignore also the SYN category as its sole problem is merely a syntax test, and the GEG category as all problems are zero-rated and easily solved by *Beagle*.

The experiments were run on a MacBook Pro with a 2.3 GHz Intel i7 processor and 16 GB main memory. The CPU time limit was 120 s (a higher time limit does not help much solving more problems). Tables 1 and 2 summarize the results for the problems with a known "theorem" or "unsatisfiable" status with non-zero rating. *Beagle* can also solve some satisfiable problems, but most of them are rather easy and can be solved by the BG solver alone. Unfortunately, the TPTP does not contain reasonably difficult satisfiable problems from the GBT-fragment, which would be interesting for exploiting the completeness result of Sect. 10.

[15] *Beagle* is available at https://bitbucket.org/peba123/beagle. The distribution includes the (Scala) source code and a ready-to-run Java jar-file.

[16] http://tptp.org.

Table 1. Number of TPTP version 7.2.0 problems solved, of all non-zero rated "theorem" or "unsatisfiable" problems involving any form of arithmetic. The flag settings giving the best result are in typeset in bold. The CPU time limit was 120 s. The column "Any" is the number of problems solved in the union of the four setting to its left. For the "Auto" column see the description of auto-mode in the main text further below. For auto-mode only, the CPU time limit was increased to 300 s.

Category	#Problems	Ordinary variables		Abstraction variables		Any	Auto
		BG simp cautious	BG simp aggressive	BG simp cautious	BG simp aggressive		
ARI	444	356	**357**	353	355	362	355
DAT	23	9	**12**	6	7	13	12
MSC	3	3	3	3	3	3	3
NUM	36	30	29	**34**	**34**	34	34
PUZ	1	1	1	1	1	1	1
SEV	2	0	0	0	0	0	0
SWV	1	1	1	1	1	1	1
SWW	244	91	88	**92**	89	97	95
SYO	1	0	0	0	0	0	0
Total	755	419	471	**490**	**490**	511	501

Table 1 is a breakdown of *Beagle*'s performance by TPTP problem categories and four flag settings. *Beagle* features a host of flags for controlling its search, but in Table 1 we varied only the two most influential ones: one that controls whether input arithmetic variables are taken as ordinary variables or as abstraction variables. (Sect. 5 discusses the trade-off between these two kinds of variables.) The other controls whether simplification of BG terms is done cautiously or aggressively.

To explain, the cautious simplification rules comprise evaluation of arithmetic terms, e. g. $3 \cdot 5$, $3 < 5$, $\alpha + 1 < \alpha + 1$ (equal lhs and rhs terms in inequations), and rules for TPTP-operators, e. g., $\mathsf{to\_rat}(5)$, $\mathsf{is\_int}(3.5)$. For aggressive simplification, integer sorted subterms are brought into a polynomial-like form and are evaluated as much as possible. For example, the term $5 \cdot \alpha + \mathsf{f}(3 + 6, \alpha \cdot 4) - \alpha \cdot 3$ becomes $2 \cdot \alpha + \mathsf{f}(9, 4 \cdot \alpha)$. These conversions exploit the associativity and commutativity laws for $+$ and $\cdot$. We refer the reader to [7] for additional aggressive simplification rules, but we note here that aggressive simplification does not always preserve sufficient completeness. For example, in the clause set $N = \{\mathsf{P}(1 + (2 + \mathsf{f}(X))),\ \neg\mathsf{P}(1 + (X + \mathsf{f}(X)))\}$ the first clause is aggressively simplified, giving $N' = \{\mathsf{P}(3 + \mathsf{f}(X)),\ \neg\mathsf{P}(1 + (X + \mathsf{f}(X)))\}$. Both N and N' are LIA-unsatisfiable, $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathrm{LIA})$ is unsatisfiable, but $\mathrm{sgi}(N') \cup \mathrm{GndTh}(\mathrm{LIA})$ is satisfiable. Thus, N is (trivially) sufficiently complete while N' is not.

These two flag settings, in four combinations in total, span a range from "most complete but larger search space" by using ordinary variables and cautious simplification, to "most incomplete but smaller search space" by using

abstraction variables and aggressive simplification. As the results in Table 1 show, the flag setting "abstraction variables" solves more problems than "ordinary variables", but not uniformly so. Indeed, as indicated by the "Any" column in Table 1, there are problems that are solved only with either ordinary or abstraction variables.

Some more specific comments, by problem categories:

ARI. Of the 362 solved problems, 14 are not solved in every setting. Of these, four problems require cautious simplification, and five problems require aggressive simplification. This is independent from whether abstraction or ordinary variables are used.

DAT. The DAT category benefits significantly from using ordinary variables. There is only one problem, DAT075=1.p, that is not solved with ordinary variables. Two problems, DAT072=1.p and DAT086=1.p are solvable only with ordinary variables and aggressive simplification.

Many problems in the DAT category, including DAT086=1.p, state *existentially quantified* theorems about data structures such as arrays and lists. If they are of an arithmetic sort, these existentially quantified variables must be taken as ordinary variables. This way, they can be unified with BG-sorted FG terms such as $\mathsf{head}(\mathsf{cons}(x, y))$ (which appear in the list axioms) which might be necessary for getting a refutation at all.

A trivial example for this phenomenon is the entailment $\{\mathsf{P}(\mathsf{f}(1))\} \models \exists x\ \mathsf{P}(x)$, where f is BG-sorted, which is provable only with ordinary variables.

NUM. This category requires abstraction variables. With it, four of the problems can be solved in the NUM category (NUM859=1.p, NUM860=1.p, NUM861=1.p, NUM862=1.p), as the search space with ordinary variables is too big.

SWW. By and large, cautious BG simplification fares slightly better on the SWW problems. Of the 97 problems solved, 16 are not solved in every setting, and the settings that do solve it do not follow an obvious pattern.

We were also interested in *Beagle*'s performance, on the same problems, broken down by the calculus features introduced in this paper. Table 2 summarizes our findings for five configurations ①–⑤ obtained by progressively enabling these features. In order to assess the usefulness of the features we filtered the results by problem rating. The column "≥0.75", for instance, lists the number of solved problems, of all 80 known "theorem" or "unsatisfiable" problems with a rating 0.75 or higher and that involve some form of arithmetic.

The predecessor calculus of [5] uses an exhaustive abstraction mechanism that turns every side of an equation into either a pure BG or pure FG term. All BG variables are always abstraction variables. Configuration ① implements this calculus, with the only deviation of an added splitting rule. The splitting rule [29] breaks apart a clause into variable-disjoint parts and leads to a branching search space for finding corresponding sub-proofs. See again [7] for more details.

Table 2. Number of "theorem" or "unsatisfiable" problems solved, by calculus features and problem rating, excluding the HWV-problems.

	Abstraction	Feature	Rating, # Problems			
			≥0.1	≥0.5	≥0.75	≥0.88
			756	187	80	55
①	Standard	N/A	355	30	5	1
②		+Define	493	38	5	1
③	Weak	+Define	490	40	5	1
		+Define				
④		+Ordinary vars	500	44	5	1
		+Define				
		+Ordinary vars				
⑤		+BG simp aggressive	511	45	5	1

In our experiments splitting is always enabled, in particular also for configuration ① for better comparability of result. Cautious BG simplification is enabled for configuration ① and the subsequent configurations ②–④.

Configuration ② differs from configuration ① only by an additional Define rule. (As said earlier, the Define rule can be added without problems to the previous calculus.) By comparing the results for ① and ② it becomes obvious that adding Define improves performance dramatically. This applies to the new calculus as well. The Define rule stands out and should always be enabled.

Configuration ③ replaces the standard abstraction mechanism of [5] by the new weak abstraction mechanism of Sect. 5. Weak abstraction seems more effective than standard abstraction for problems with a higher rating, but the data set supporting this conclusion is very small.

There are five problems, all from the SWW category[17] that re solved *only* with configuration ②, and there is one problem, SWW607=2.p, that is solved only by configurations ① and ②.

There are four solvable problems with rating 0.75. These are ARI595=1.p – ARI598=1.p, which are "simple" problems involving a free predicate symbols over the integer background theory. The problem ARI595=1.p, for instance, is to prove the validity of the formula $(\forall\, z : \mathbb{Z}\ a \leq z \wedge z \leq a + 2 \rightarrow p(z)) \rightarrow \exists\, x : \mathbb{Z}\ p(3 \cdot x)$.[18] The calculus and implementation techniques needed for solving such problems are rather different to those needed for solving combinatory problems involving trivial arithmetics only, like, e.g., the HWV-problems.

[17] SWW583=2.p, SWW594=2.p, SWW607=2.p, SWW626=2.p, SWW653=2.p and SWW657=2.p.

[18] At the time of this writing, there are only four provers (including *Beagle*) registered with the TPTP web infrastructure that can solve these problems. Hence the rating 0.75.

Configuration ④ is the same as ③ except that it includes the results for general variables instead of abstraction variables. Similarly, configuration ⑤ is the same as ④ except that it includes the results for aggressive BG simplification. It is the union of all results in Table 1.

For comparison with other implemented theorem provers for first-order logic with arithmetics, we ran *Beagle* on the problem set used in the 2018 edition of the CADE ATP system competition (CASC-J9).[19]. The competing systems were CVC4 [6], Princess [26], and two versions of Vampire [20].

In the competition, the systems were given 200 problems from the TPTP problem library, 125 problems over the integers as the background theory (TFI category), and 75 over the reals (TFE category). The system that solves the most problems in the union of the TFI and TFE categories within a CPU time limit of 300 s wins. We applied *Beagle* in an "auto" mode, which time-slices (at most) three parameter settings. These differ mainly in their use of abstraction variables or ordinary variables, and the addition of certain arithmetic lemmas.

Table 3. CADE ATP system competition results 2018 and *Beagle*'s performance on the same problem sets.

	Vampire 4.3	Vampire 4.1	CVC4 1.6pre	Princess 170717	Beagle 0.9.51
#Solved TFI (of 125)	93	98	85	62	36
#Solved TFE (of 75)	70	64	72	43	44
#Solved TFA (of 200)	163	162	157	105	70

The results are summarized in Table 3. We note that *Beagle* was run on different hardware but the same timeout of 300 s. The results are thus only indicative of *Beagle*'s performance, but we do not expect significantly different result had it participated. In the TFI category, of the 36 problems solved, 5 require the use of ordinary variables. In the TFE category, 16 problems involve the ceiling or floor function, which is currently not implemented, and hence cannot be attempted.

In general, many problems used in the competition are rather large in size or search space and would require a more sophisticated implementation of *Beagle*.

13 Conclusions

The main theoretical contribution of this paper is an improved variant of the hierarchic superposition calculus. One improvement over its predecessor [5] is a different form of "abstracted" clauses, the clauses the calculus works with internally. Because of that, a modified completeness proof is required. We have

[19] http://tptp.cs.miami.edu/~tptp/CASC/J9/.

argued informally for the benefits over the old calculus in [5]. They concern making the calculus "more complete" in practice. It is hard to quantify that exactly in a general way, as completeness is impossible to achieve in presence of background-sorted foreground function symbols (e. g., "head" of integer-sorted lists). To compensate for that to some degree, we have reported on initial experiments with a prototypical implementation on the TPTP problem library. These experiments clearly indicate the benefits of our concepts, in particular the definition rule and the use of ordinary variables. There is no problem that is solved only by the old calculus only. Certainly more experimentation and an improved implementation is needed to also solve bigger-sized problems with a larger combinatorial search space.

We have also obtained two new completeness results for certain clause logic fragments that do not require compactness of the background specification, cf. Sects. 10 and 11. The former is loosely related to the decidability results in [22], as discussed in Sect. 9. It is also loosely related to results in SMT-based theorem proving. For instance, the method in [18] deals with the case that variables appear only as arguments of, in our words, foreground operators. It works by ground-instantiating all variables in order to being able to use an SMT-solver for the quantifier-free fragment. Under certain conditions, finite ground instantiation is possible and the method is complete, otherwise it is complete only modulo compactness of the background theory (as expected). Treating different fragments, the theoretical results are mutually non-subsuming with ours. Yet, on the fragment they consider we could adopt their technique of finite ground instantiation before applying Theorem 10.7 (when it applies). However, according to Theorem 10.7 our calculus needs instantiation of *background-sorted variables only*, this way keeping reasoning with foreground-sorted terms on the first-order level, as usual with superposition.

References

1. Althaus, E., Kruglov, E., Weidenbach, C.: Superposition modulo linear arithmetic SUP(LA). In: Ghilardi, S., Sebastiani, R. (eds.) FroCoS 2009. LNCS (LNAI), vol. 5749, pp. 84–99. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04222-5_5
2. Armando, A., Bonacina, M.P., Ranise, S., Schulz, S.: New results on rewrite-based satisfiability procedures. ACM Trans. Comput. Log. **10**(1), 4 (2009)
3. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Logic Comput. **4**(3), 217–247 (1994)
4. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Handbook of Automated Reasoning. North Holland (2001)
5. Bachmair, L., Ganzinger, H., Waldmann, U.: Refutational theorem proving for hierarchic first-order theories. Appl. Algebra Eng. Commun. Comput **5**, 193–212 (1994)
6. Barrett, C., et al.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 171–177. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_14

7. Baumgartner, P., Bax, J., Waldmann, U.: Beagle – a hierarchic superposition theorem prover. In: Felty, A.P., Middeldorp, A. (eds.) CADE 2015. LNCS (LNAI), vol. 9195, pp. 367–377. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21401-6_25
8. Baumgartner, P., Fuchs, A., Tinelli, C.: $\mathcal{ME}$(LIA) - model evolution with linear integer arithmetic constraints. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 258–273. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89439-1_19
9. Baumgartner, P., Tinelli, C.: Model evolution with equality modulo built-in theories. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS (LNAI), vol. 6803, pp. 85–100. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22438-6_9
10. Baumgartner, P., Waldmann, U.: Hierarchic superposition: completeness without compactness. In: Košta, M., Sturm, T. (eds.), Fifth International Conference on Mathematical Aspects of Computer and Information Sciences, MACIS 2013, pp. 8–12, Nanning, China (2013)
11. Baumgartner, P., Waldmann, U.: Hierarchic superposition with weak abstraction. In: Bonacina, M.P. (ed.) CADE 2013. LNCS (LNAI), vol. 7898, pp. 39–57. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38574-2_3
12. Baumgartner, P., Waldmann, U.: Hierarchic superposition revisited (2019). http://arxiv.org/abs/1904.03776
13. Bonacina, M.P., Lynch, C., de Moura, L.M.: On deciding satisfiability by theorem proving with speculative inferences. J. Autom. Reason. **47**(2), 161–189 (2011)
14. de Moura, L., Bjørner, N.: Engineering DPLL(T) + saturation. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 475–490. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71070-7_40
15. Dickson, L.E.: Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. Am. J. Math. **35**(4), 413–422 (1913)
16. Ganzinger, H., Korovin, K.: Theory instantiation. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS (LNAI), vol. 4246, pp. 497–511. Springer, Heidelberg (2006). https://doi.org/10.1007/11916277_34
17. Ge, Y., Barrett, C., Tinelli, C.: Solving quantified verification conditions using satisfiability modulo theories. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 167–182. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73595-3_12
18. Ge, Y., de Moura, L.: Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 306–320. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02658-4_25
19. Korovin, K., Voronkov, A.: Integrating linear arithmetic into superposition calculus. In: Duparc, J., Henzinger, T.A. (eds.) CSL 2007. LNCS, vol. 4646, pp. 223–237. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74915-8_19
20. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_1
21. Kruglov, E.: Superposition modulo theory. Doctoral dissertation, Universität des Saarlandes, Saarbrücken, October 2013
22. Kruglov, E., Weidenbach, C.: Superposition decides the first-order logic fragment over ground theories. Math. Comput. Sci. **6**, 427–456 (2012)

23. Nieuwenhuis, R.: First-order completion techniques. Technical report, Universidad Politécnica de Cataluña, Dept. Lenguajes y Sistemas Informáticos (1991)
24. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: from an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). J. ACM **53**(6), 937–977 (2006)
25. Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Handbook of Automated Reasoning, pp. 371–443. Elsevier and MIT Press (2001)
26. Rümmer, P.: A constraint sequent calculus for first-order logic with linear integer arithmetic. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 274–289. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89439-1_20
27. Sutcliffe, G.: The TPTP problem library and associated infrastructure. From CNF to TH0, TPTP v6.4.0. J. Autom. Reason. **59**(4), 483–502 (2017)
28. Walther, C.: Many-sorted unification. J. ACM **35**(1), 1–17 (1988)
29. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: Schmidt, R.A. (ed.) CADE 2009. LNCS (LNAI), vol. 5663, pp. 140–145. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02959-2_10