Hierarchic Superposition With Weak Abstraction

Peter Baumgartner
NICTA* and Australian National University
Canberra, Australia

Uwe Waldmann
Max-Planck-Institut für Informatik
Saarbrücken, Germany

February 19, 2013

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 a new completeness result for the fragment where all background-sorted terms are ground.

1 Introduction

Many applications of automated deduction require reasoning modulo background theories, in particular some form of integer arithmetic. 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 [NOT06] for the quantifier-free case by instantiation heuristics for quantifiers [GBT07, GdM09, e.g.]. Another line of research is concerned with adding black-box reasoners for specific background theories to first-order automated reasoning methods (resolution [BGW94, KV07, AKW09], sequent calculi [Rüm08], instantiation methods [GK06, BFT08, BT11], etc). In both cases, a major unsolved research challenge

*NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative.

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.

In [BGW94], 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, where one investigates (un)satisfiability 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 in fact destroy sufficient completeness with respect to simple instances. We show that this problem can be avoided by using a new form of clause abstraction, which also has the advantage that it often results in shorter clauses and better supports clause indexing.

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 behaviour: The prover might for instance try to *make* the input sufficiently complete by adding further theory axioms. In the calculus by Bachmair, Ganzinger, and Waldmann, however, 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 instead of one, 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 and how our approach tackles it in relation to [BGW94]. Consider the clause set $N = \{C\}$ where $C = \mathsf{f}(1) < \mathsf{f}(1)$. Assume the background theory is integer arithmetic and that f is an integer-sorted operator from the foreground (free) signature. In the intended semantics N is unsatisfiable. However, N is not sufficiently complete, and, indeed, it admits models in which $\mathsf{f}(1)$ is interpreted as some junk element , an element of the domain of the integer sort that is not a numeric constant. Hence, both the calculus in [BGW94] and ours are excused to not find a refutation. In order to fix that one could consider several alternatives. First, 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 [BGW94] 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 . Our abstraction mechanism always preserves sufficient completeness and our calculus will find a refutation.

From this example one might be tempted to think that replacing the abstraction mechanism in [BGW94] with ours gives all the advantages of our calculus. This is not the case, however. 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 [BGW94]. 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 $\mathsf{f}(1) \approx e$, where e is a fresh symbolic constant belonging to the background signature (a "parameter"). The set $N''' = \{C, \mathsf{f}(1) \approx e\}$ 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 correctness results where appropriate. These include a general completeness result, modulo compactness of the background theory, and a specific completeness result for clause sets over ground background-sorted terms that does not require compactness. We also report on initial experiments with a prototypical implementation on the TPTP problem library.

Related Work. The relation with the predecessor calculus in [BGW94] is discussed above and also further below. What we say there also applies to other developments rooted in that calculus, [AKW09, e.g.]. The specialised version of hierarchic superposition in [KW12] will be discussed in Sect. 7 below. The resolution calculus in [KV07] 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 [BFT08, BT11] lacks the treatment of background-sorted foreground function symbols. The same applies to the sequent calculus in [Rüm08], which treats linear arithmetic with built-in rules for quantifier elimination. The instantiation method in [GK06] 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 combining DPLL(T)(-like) procedures with general first-order theorem provers as (semi-)decision procedures for specific theories [dMB08, ABRS09, BLdM11].

The paper [GdM09] will be discussed in Sect. 9.

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.

An *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 *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. A *renaming* is a substitution that is also a permutation (i.e., a bijective substitution whose codomain consists of variables only). Substitutions are 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 $t\sigma = s$. The terms s and t are *variants* iff there is a renaming substitution ρ such that $s\rho = t$. All these notions carry over to equations, literals and clauses in the obvious way.

The *domain* of a substitution σ is the set $\mathrm{dom}(\sigma) = \{x \mid x \neq x\sigma\}$. We work with substitutions with finite domains only, 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.

An *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

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

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 $N \models \{C\}$.

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*. We say that a class of Σ-interpretations $\mathcal{J}$ 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 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:

[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.

(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 $\{\, n \leq b \mid n \in \mathbb{N} \,\}$, where b 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 $\{\, n \leq b \mid n \in \mathbb{N} \,\}$ 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 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 use $\{0, 1, 2, \dots\}$ to denote BG domain elements, $\{+, -, <, \leq\}$ to denote (non-parameter) BG operators, and the possibly subscripted letters $\{x, y\}$, $\{X, Y\}$, $\{a, b\}$, 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 equations, 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*.

[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.

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*. If N is a set of BG clauses, it is $\mathcal{B}$-satisfiable if and only if some interpretation of $\mathcal{B}$ is a model of N.

Refutational hierarchic theorem proving amounts to checking 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 is *simple* if it maps every abstraction variable in its domain to a *pure* BG term. For example, $[x \mapsto 1 + Y + a]$, $[X \mapsto 1 + Y + a]$ and $[x \mapsto \mathsf{f}(1)]$ all are simple, whereas $[X \mapsto 1 + y + a]$ and $[X \mapsto \mathsf{f}(1)]$ are not. Let F be a clause or (possibly infinite) clause set. By *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 sufficiently complete (w.r.t. simple instances) iff for every Σ-model J of sgi$(N) \cup \mathrm{GndTh}(\mathcal{B})$ and every ground BG-sorted FG term s there is a ground BG term t such that $J \models s \approx t$.*[4]

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 $\succ$.

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

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

5 Weak Abstraction

Hierarchic superposition calculi take FG clauses as inputs, 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 t in a clause by a new variable and adds a disequations to the clause, so that $C[t]$ is converted into the equivalent clause $\zeta \not\approx t \vee C[\zeta]$, where ζ is a new (abstraction or ordinary) variable.

The calculus by Bachmair, Ganzinger, and Waldmann [BGW94] works on "fully abstracted" clauses: Background terms occuring 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 operator symbols. In the terminology of the current paper, all BG-sorted variables in [BGW94] have the kind "abstraction", that is, one considers only instances where BG-sorted variables are mapped to BG terms.[5]

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 w.r.t. simple instances. 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 w.r.t. simple instances. In other words, the calculus of [BGW94] is refutationally complete for clause sets that are fully abstracted and sufficiently complete w.r.t. simple instances, but full abstraction may destroy sufficient completeness w.r.t. simple instances. (In fact, the calculus is not able to refute N'.)

We will now describe another abstraction technique that avoids this problem. We call a term t occurring in a clause $C[t]$ a *maximal BG term* if it is not a proper subterm of another BG term. We say that t is a *target term* if t is a maximal BG term in $C[t]$ that is a subterm of some FG term and if t is neither a domain element nor a variable. While C has a target term t, we replace $C[t]$ by $C[X] \vee X \not\approx t$ for a new abstraction variable X if t is pure, and by $C[x] \vee x \not\approx t$ for a new ordinary variable x if t is impure.

[5] In a footnote in [BGW94] it is claimed that it is sufficient to abstract out only non-variable BG terms that occur below a FG operator. This is incorrect: Using this abstraction rule, neither our calculus nor the calculus of [BGW94] would not 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 w.r.t. simple instances.

The resulting clause has the form $C' \vee E$, where the literals of C' are obtained from the literals of C by replacing BG subterms by variables, and where the subclause E consists of the new disequations. We call $C' \vee E$ the *weakly abstracted version of* C, denoted by $\mathrm{abstr}(C)$, and we say that C is *weakly abstracted* iff $C = \mathrm{abstr}(C)$.

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

In principle, one could always use ordinary variables instead of abstraction variables for weak abstraction (and one could also define full abstraction using ordinary variables instead of abstraction variables). However, the introduction of new variables during abstraction 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. This negative effect is reduced by using abstraction variables. For example, if we use an LPO with precedence $\mathsf{f} > \mathsf{g} > a > b$ then $\mathsf{f}(a)$ is the only maximal term in the clause $\mathsf{f}(a) \approx \mathsf{g}(b)$, as $\mathsf{f}(a) \succ \mathsf{g}(b)$. If we use ordinary variables to abstract out a and b, we obtain the two terms $\mathsf{f}(x)$ and $\mathsf{g}(y)$, which are incomparable. If we use abstraction variables, we obtain $\mathsf{f}(X)$ and $\mathsf{g}(Y)$, and for every simple ground substitution γ, $\mathsf{f}(X)\gamma$ is strictly larger that $\mathsf{g}(Y)\gamma$ in the same ordering, so we can still consider $\mathsf{f}(X)$ as the only maximal term in the clause.

On the other hand, we cannot avoid to use ordinary variables for abstraction, if the term to be abstracted out is impure (i.e., if it contains ordinary variables itself); 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 w.r.t. simple instances.

Proposition 5.1 *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 $sgi(N)$ and $sgi(N')$ are equivalent and N' is sufficiently complete w.r.t. simple instances whenever N is.*

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 w.r.t. simple instances, 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 w.r.t. simple instances. However, if we add the clause

$x > y \vee x \leq y$, the set of clauses becomes (vacuously) sufficiently complete w.r.t. simple instances 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 5.2 *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 sgi(N') and sgi(N'') are equivalent and sgi(N') is sufficiently complete w.r.t. simple instances if and only if sgi(N'') is.*

6 Base Inference System

An *inference system* $\mathcal{I}$ is a set of inference rules. Evey inference rule defined in this paper is applicable only to weakly abstracted premise clauses, without explicitly stating that. 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.[6]

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

if (i) neither s nor t is a pure BG term, (ii) σ is a simple mgu of s and t, and (iii) $(s \not\approx t)\sigma$ is maximal in $(s \not\approx t \vee C)\sigma$.[7]

For example, Equality resolution is applicable to $1 + \mathsf{c} \not\approx 1 + x$ with the simple mgu $[y \mapsto \mathsf{c}]$, but it is not applicable to $1 + a \not\approx 1 + x$. Recall that a simple mgu cannot map an abstraction variable to an ordinary variable, as ordinary variables are not pure BG terms. Standard unification algorithms can be modified in a straightforward way for computing simple mgus.

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

[6] With weak abstraction, it is *not* possible to replace Equality factoring by Factoring and Merging paramodulation. The inference system can be extended by selection functions, but only negative FG literals in clauses that do not contain ordinary BG variables may be selected.

[7] As in [BGW94], it is possible to strengthen condition (iii) 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).

if (i) neither l nor u is a pure BG term, (ii) u is not a variable, (iii) σ is a simple mgu of l and u, (iv) $r\sigma \not\succeq l\sigma$, (v) $(l \approx r)\sigma$ is strictly maximal in $(l \approx r \vee C)\sigma$, (vi) $t\sigma \not\succeq s\sigma$, and (vii) $(s \not\approx t)\sigma$ 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) neither l nor u is a pure BG term, (ii) u is not a variable, (iii) σ is a simple mgu of l and u, (iv) $r\sigma \not\succeq l\sigma$, (v) $(l \approx r)\sigma$ is strictly maximal in $(l \approx r \vee C)\sigma$, (vi) $t\sigma \not\succeq s\sigma$, and (vii) $(s \not\approx t)\sigma$ is strictly maximal in $(s \approx t \vee D)\sigma$.

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

where (i) both l and s are not pure BG terms, (ii) σ is a simple mgu of l and s, (iii) $(l \approx r)\sigma$ is maximal in $(l \approx r \vee s \approx t \vee C)\sigma$, (iv) $r\sigma \not\succeq l\sigma$, and (v) $t\sigma \not\succeq s\sigma$.

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

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.

In contrast to [BGW94], the inference rules above include an explicit weak abstraction in their conclusion. Without it, conclusions would not be weakly abstracted in general. For example Negative superposition applied to the weakly abstracted clauses $\mathsf{f}(X) \approx 1 \vee X \not\approx a$ and $\mathsf{P}(\mathsf{f}(1)+1)$ would then yield $\mathsf{P}(1+1) \vee 1 \not\approx a$, whose P-literal is not weakly abstracted.

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 6.1 *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* $\mathcal{R}_{\mathrm{Cl}}(N) \subseteq \mathcal{R}_{\mathrm{Cl}}(N')$.

(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* $\mathcal{R}_{\mathrm{Inf}}(N) \subseteq \mathcal{R}_{\mathrm{Inf}}(N \setminus N')$*.*

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

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 [BGW94] and relate $\mathrm{HSP}_{\mathrm{Base}}$ inferences to the corresponding ground inferences in the standard superposition calculus SSP [NR01].

Let N be a set of ground clauses. 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 $C_1, \ldots, C_n \models C_0$. It is well known that ground SSP together with $(\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}, \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}})$ 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 6.2 *Let* N *be a set of 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* $sgi(\iota) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(sgi(N) \cup \mathrm{GndTh}(\mathcal{B}))$*, 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 abstracted clauses* C *such that* $sgi(C) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(sgi(N) \cup \mathrm{GndTh}(\mathcal{B})) \cup \mathrm{GndTh}(\mathcal{B})$*.*[8]

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 terminating, right-reduced, and confluent. Now let E_I be the set of all rules $l \to r$ in E'_I such that l is not reducible by $\mathrm{E}'_I \setminus \{l \to r\}$. It is fairly easy to prove that 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.

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 irreducible by E_I.

[8] In contrast to [BGW94], we include $\mathrm{GndTh}(\mathcal{B})$ in the redundancy criterion (and also in the definition of sufficient completeness w.r.t. simple instances).

Let N be a set of 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 $\}$.

Theorem 6.3 *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 $sgi(N)$ 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 $\mathcal{R}^{\mathcal{S}}$.*

We do not spell out in detail theorem proving *processes* here, because the well-known framework of standard resolution [BG01] can be readily instantiated with our calculus. In particular, it justifies the following version of a generic simplification rule for clause sets.

$$\mathsf{Simp}\ \frac{N \cup \{C\}}{N \cup \{D\}}$$

if (i) D is weakly abstracted, (ii) $\mathrm{GndTh}(\mathcal{B}) \cup N \cup \{C\} \models D$, and (iii) C is redundant w.r.t. $N \cup \{D\}$.

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 $(2 + 3) + a$ by $5 + a$ and deleting an unsatisfiable BG literal $5 + a < 4 + a$ from a clause. Any clause of the form $C \vee X \not\approx d$ where d is domain element can be simplified to $C[X \mapsto d]$.

Theorem 6.4 *If the BG specification $(\Sigma, \mathcal{B})$ is compact, then* $\mathrm{HSP}_{\mathrm{Base}}$ *and $\mathcal{R}^{\mathcal{H}}$ are refutationally complete for all sets of clauses that are sufficiently complete w.r.t. simple instances.*

7 Sufficient Completeness by Define

The hierarchic superposition calculus HSP is obtained by extending $\mathrm{HSP}_{\mathrm{Base}}$ with a new inference rule, Define. It derives "definitions" of the form $t \approx e$, where t is a ground BG-sorted FG term and e is a parameter. Its effect is to force in every interpretation t to be equal to some element of the carrier set of the proper sort, denoted by the parameter e. This way, it achieves sufficient completeness "locally" for t. For economy of reasoning, definitions are introduced only on a by-need basis, when t appears in a current clause, and $t \approx e$ is used to simplify that clause immediately.

We need one more preliminary definition before introducing Define formally.

Definition 7.1 (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 $\zeta \in vars(t)$.*

Every clause can be unabstracted by repeatedly replacing $C \vee \zeta \not\approx t$ by $C[\zeta \mapsto t]$ whenever $t = \zeta$ or $\zeta \notin \mathsf{vars}(t)$. Every such disequation $\zeta \not\approx t$ is called an *assignment*. By $\mathrm{unabstr}(C)$ we denote an unabstracted version of C that can be obtained this way.[9] 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 $\mathrm{unabstr}(t, C)$.

$$\mathsf{Define}\ \frac{N \cup \{L[t[\zeta_1, \ldots, \zeta_n]] \vee D\}}{N \cup \{\mathrm{abstr}(t[t_1, \ldots, t_n] \approx e_{t[t_1,\ldots,t_n]}),\ \mathrm{abstr}(L[e_{t[t_1,\ldots,t_n]}] \vee D)\}}$$

if (i) $t[\zeta_1, \ldots, \zeta_n]$ is a minimal BG-sorted non-variable term with a toplevel FG operator, (ii) $t[t_1, \ldots, t_n] = \mathrm{unabstr}(t[\zeta_1, \ldots, \zeta_n], L[t[\zeta_1, \ldots, \zeta_n]] \vee D)$, (iii) $t[t_1, \ldots, t_n]$ is ground, and (iv) $e_{t[t_1,\ldots,t_n]}$ is a parameter, uniquely determined by the term $t[t_1, \ldots, t_n]$.

In condition (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. Instead of $e_{t[t_1,\ldots,t_n]}$ a fresh parameter could be used as well, but this would be less economical. Notice the Define-rule preserves $\mathcal{B}$-satisfiability, not $\mathcal{B}$-equivalence. In our main application, Thm. 7.4 below, every ζ_i will always be an abstraction variable.

Example 7.2 Consider the 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 $\mathsf{f}(0) \approx e_{\mathsf{f}(0)}$ and $e_{\mathsf{f}(0)} > 0$, the latter replacing $\mathsf{f}(0) > 0$. From the third clause and $\mathsf{f}(0) \approx e_{\mathsf{f}(0)}$ we obtain $\mathsf{Q}(e_{\mathsf{f}(0)})$, and with the fourth clause $0 > e_{\mathsf{f}(0)}$. Finally we apply Close to $\{e_{\mathsf{f}(0)} > 0,\ 0 > e_{\mathsf{f}(0)}\}$. □ □

In general, the Define rule is not strong enough to establish sufficient completeness. In practice, hence, it is interesting to identify conditions under which sufficient completeness can be established by means of Define *and* compactness poses no problems, so that a complete calculus results. In the following we treat one such case, the *ground BG-sorted term fragment (GBT fragment)*.

A clause set N belongs to the GBT fragment iff every clause $C \in N$ is a GBT clause, that is, all BG-sorted terms in C are ground. Every clause C derivable from $N_0 = \mathrm{abstr}(N)$ by the HSP calculus without Simp remains a GBT clause *after unabstraction*, that is $\mathrm{unabstr}(C)$ is a GBT clause. This is straightforward to check by analyzing the inference rules.[10] We exploit this property for defining a (mildly) specialized version of the HSP calculus and reasoning about its properties. In particular we can define simplification in a compatible way:

[9] In general, unabstraction does not yield a unique result. All results are equivalent, however, and we can afford to select any one and disregard the others.

[10] For instance, a superposition inference that replaces a BG-sorted FG term inside a BG-sorted FG term may require weak abstraction of its conclusion and this way take it outside the GBT fragment (but only in an inessential way). Still, abstraction always results in disequations $X \not\approx t$ between abstraction variables and (hence pure) BG terms, and unabstraction removes all these variables again.

Definition 7.3 *Let $\succ_{fin}$ be any strict (partial) term ordering such that for every ground BG term s only finitely many ground BG terms t with $s \succ_{fin} t$ exist.*[11] *We say that a* Simp *inference with premise $N \cup \{C\}$ and conclusion $N \cup \{D\}$ is suitable (for the GBT fragment) iff (i)* unabstr(D) *is a GBT clause, (ii) for every BG term occuring in* unabstr(D) *there is a BG term $s \in$* unabstr(C) *such that $s \succeq_{fin} t$, and (iii) every term t in D contains a BG-sorted FG operator only at toplevel position, if at all. We say the* Simp *inference rule is suitable iff every* Simp *inference is.*

Expected simplification techniques like demodulation, subsumption deletion and evaluation of BG subterms are all covered by suitable Simp rules. The latter is possible because simplifications are not only decreasing w.r.t. $\succ$ but *additionally* also decreasing w.r.t. $\succeq_{\text{fin}}$, as expressed in condition (ii). Without it, e.g., the clause $\mathsf{P}(1+1,0)$ would admit infinitely many simplified versions $\mathsf{P}(2,0)$, $\mathsf{P}(2,0+0)$, $\mathsf{P}(2,0+(0+0))$, ... Moreover, we need to have that all clauses derivable from GBT clauses are GBT clauses after unabstraction. For Simp (only) we have to demand that explicitly in condition (i). Condition (iii) is needed to make sure that no new BG terms are generated in derivations.

Theorem 7.4 *The* HSP *calculus with a suitable* Simp *inference rule and a specific strategy is refutationally complete for the ground BG-sorted term fragment. More precisely, let N be a clause set such that every BG-sorted term in N is ground. Let $N_0 =$* abstr(N) *and $N_0, \ldots, N_k$ a derivation such that every $C \in N_k$ either does not contain any BG-sorted FG operator or* unabstr(C) *is a ground positive unit clause of the form $f(t_1, \ldots, t_n) \approx t$ where f is a BG-sorted FG operator and $t_1, \ldots, t_n, t$ do not contain BG-sorted FG operators.*

If N is $\mathcal{B}$-unsatisfiable then there there is a refutation of N_k without the Define *inference rule.*

For all GBT clause sets N, thanks to the Define rule, all occurences of BG-sorted FG terms in N_0 can be eliminated in finitely many steps by introducing definitions of the form stated in the theorem. This shows that the clause set N_k in Thm. 7.4 always exists.

In [KW12] it has been 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 as our Define rule. The resulting clause set then is fully abstracted and hierarchic superposition is applied. Certain modifications of the inference rules make sure derivations always terminate. Simplification is restricted to subsumption deletion.

Interestingly, basification and subsequent full abstraction yields a clause set with a syntactic structure that is similar to what we get with weak abstraction and subsequent Define applications, as explained after Thm. 7.4. We expect we can get decidability results for the same fragments with similar techniques. But notice our focus is a different

[11] A KBO with non-zero weight can be used for $\succ_{\text{fin}}$.

one by allowing arbitrary FG signatures and quantification over FG terms, which forbids a decidability result. This is why we defined an *inference rule*, Define, in order to be able to eliminate BG sorted FG symbols also during derivations. Ex. 7.2, for instance, cannot be solved with basification during preprocessing.

8 Implementation and Experiments

We have implemented the HSP calculus and carried out experiments with the TPTP Library [Sut09]. Our implementation, "Beagle", is intended as a testbed for rapidly trying out theoretical ideas for their practical viability. Beagle is in an early stage of development. Nevertheless it is a full implementation of HSP and accepts TPTP formulas over linear integer arithmetic ("TFF formulas", see [SSCB12]). The BG reasoner is a quantifier elimination procedure for linear integer arithmetic (LIA) based on Cooper's algorithm. The HSP calculus itself is implemented in a straightforward way. Fairness is achieved through a combination of measuring clause lengths, depths and their derivation-age. Implemented simplification rules are evaluation of ground parameter-free BG terms and literals, expressing literals with the predicate symbols $\geq$ and $>$ in terms of $<$ and $\leq$, demodulation by unit clauses, proper subsumption deletion, and removing a positive literal L from a clause in presence of a unit clause that instantiates to the complement of L. Prover options allow the user to enable/disable the Define rule and to add certain LIA-valid clauses over ordinary variables. Unit clauses like $-(-x) \approx x$, $(x + (-y)) + y \approx x$, $x + 0 \approx x$, $x * 0 \approx 0$, $\neg(x < x)$, etc, are always helpful as demodulators. Transitivity clauses for $<$ and $\leq$ are helpful sometimes. Optionally, a split rule can be enabled for branching out into complementary unit clauses if they simplify some current clause. Dependency-directed backtracking is used for search space pruning then.

Beagle is implemented in Scala. The choice of a slow programming language and, more severely, the absence of any form of term indexing limit Beagle's applicability to small problems only. Indeed, Beagle's performance on problems that require significant combinatorial search is poor. For example, the propositional pigeonhole problem with 8 pigeons takes more than two hours, SPASS solves it in under 4 seconds using settings to get a comparable calculus and proof procedure (including splitting).

Nevertheless we tried Beagle on all first-order problems from the TPTP library (version 5.4.0) over linear integer arithmetic. The experiments were run on a MacBook Pro with a 2.4 GHz Intel Core 2 Duo processor. Here is our summary, by problem category.

ARI. Relevant are 223 problems. Many ARI problems are very simple, but roughly half of them are non-trivial by including integer-sorted non-constant FG function symbols and free predicates over the integers. The most difficult rated solved problem is ARI595=1 (Rating 0.88). Beagle times out after 60 seconds on two problems (ARI184=1 and ARI621=1), terminates with an undecided result on one satisfiable problem (ARI603=1) and solves all other 220 problems correctly, 10 non-theorems and 210 theorems. All but

two solved problems were solved very quickly, the other two below 20 seconds. Without Define, only 205 instead of 220 could be solved. Different prover options did not noticeably impact the outcome.

GEG. The five relevant problems are variations of each other. They deal with traversing weighted graphs and computing with the (sum of the) weights along paths. All five problems are solved with runtime between 3 and 66 seconds, where the hardest problem has a rating of 0.67. Essential is the use of additional LIA-valid axioms, in particular transitivity of $\leq$ and splitting. (Splitting helps here to keep the structure of BG formulas simple, which helps our implementation of Cooper's algorithm.)

NUM. The only non-easy problem that is solvable is NUM858=1 (rating 0.56). Essential is the use of abstraction variables for the input clauses and additional LIA axioms without transitivity. All easy problems are solved easily.

SEV/HWV. Six problems of SEV are relevant, about sets, stemming from a software verification context. Only two can be solved, in 3 and 130 seconds, respectively. The arithmetic involved is marginal, and Beagle is overwhelmed with the pure first-order part of the problems. The same applies to HWV, where no problem is solved.

SWV/SWW. Only one problem can be solved. For SWV996=1 (rating 1.0) which is satisfiable, the result is unknown. The Define rule cannot be applied enough to achieve sufficient completeness. The problem SWV997=1 (rating 0.44) is solved in 12 seconds. All other problems are too big to be converted into CNF in reasonable time. The same holds for all SWW problems.

SYO. Of the four problems, SYO521=1 (rating 0.50) and SYO523=1 (rating 0.67) are easily solvable.

PUZ. The only relevant problem (PUZ133=2) is not solved.

9 Conclusions

The main theoretical contribution of this paper is an improved variant of the hierarchic superposition calculus. The improvements over its predecessor [BGW94] are grounded in a different form of "abstracted" clauses, the clauses the calculus works with internally. Because of that, a new completeness proof is required. We have argued informally for the benefits over the calculus in [BGW94]. 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. "car" 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 possibility of adding background theory axioms. 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 a specific completeness result for clause sets over ground background-sorted terms and that does not require compactness. As far as we know this result is new. It is loosely related to the decidability results in [KW12], as discussed in Sect. 7. It is also loosely related to results in SMT-based theorem proving. For instance, the method in [GdM09] 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 Thm. 7.4 (when it applies). However, according to Thm. 7.4 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

[ABRS09] Alessandro Armando, Maria Paola Bonacina, Silvio Ranise, and Stephan Schulz. New results on rewrite-based satisfiability procedures. *ACM Trans. Comput. Log.*, 10(1), 2009.

[AKW09] Ernst Althaus, Evgeny Kruglov, and Christoph Weidenbach. Superposition modulo linear arithmetic SUP(LA). In *FroCos*, volume 5749 of *Lecture Notes in Computer Science*, pages 84–99. Springer, 2009.

[BFT08] Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. ME(LIA) – Model Evolution With Linear Integer Arithmetic Constraints. In *15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'08)*, volume 5330 of *Lecture Notes in Artificial Intelligence*, pages 258–273. Springer, November 2008.

[BG01] L. Bachmair and H. Ganzinger. Resolution Theorem Proving. In *Handbook of Automated Reasoning*. North Holland, 2001.

[BGW94] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Refutational theorem proving for hierarchic first-order theories. *Appl. Algebra Eng. Commun. Comput*, 5:193–212, 1994.

[BLdM11] Maria Paola Bonacina, Christopher Lynch, and Leonardo Mendonça de Moura. On deciding satisfiability by theorem proving with speculative inferences. *J. Autom. Reasoning*, 47(2):161–189, 2011.

[BT11] Peter Baumgartner and Cesare Tinelli. Model evolution with equality modulo built-in theories. In *CADE-23 – The 23nd International Conference on Automated Deduction*, volume 6803 of *Lecture Notes in Artificial Intelligence*, pages 85–100. Springer, 2011.

[dMB08] Leonardo Mendonça de Moura and Nikolaj Bjørner. Engineering DPLL(T) + saturation. In *Automated Reasoning, 4th International Joint Conference, IJCAR*, volume 5195 of *Lecture Notes in Computer Science*, pages 475–490. Springer, 2008.

[GBT07] Yeting Ge, Clark Barrett, and Cesare Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In F. Pfenning, editor, *21st International Conference on Automated Deduction (CADE-21), Bremen, Germany*, volume 4603 of *Lecture Notes in Computer Science*. Springer, 2007.

[GdM09] Yeting Ge and Leonardo Mendonça de Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2009.

[GK06] H. Ganzinger and K. Korovin. Theory instantiation. In *13th Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*, volume 4246 of *Lecture Notes in Computer Science*, pages 497–511. Springer, 2006.

[KV07] K. Korovin and A. Voronkov. Integrating linear arithmetic into superposition calculus. In *Computer Science Logic (CSL'07)*, volume 4646 of *Lecture Notes in Computer Science*, pages 223–237. Springer, 2007.

[KW12] Evgeny Kruglov and Christoph Weidenbach. Superposition decides the first-order logic fragment over ground theories. *Mathematics in Computer Science*, pages 1–30, 2012.

[NOT06] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, November 2006.

[NR01] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In *Handbook of Automated Reasoning*, pages 371–443. Elsevier and MIT Press, 2001.

[Rüm08] Philipp Rümmer. A constraint sequent calculus for first-order logic with linear integer arithmetic. In *15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'08)*, volume 5330 of *Lecture Notes in Artificial Intelligence*, pages 274–289. Springer, November 2008.

[SSCB12] Geoff Sutcliffe, Stephan Schulz, Koen Claessen, and Peter Baumgartner. The TPTP typed first-order form with arithmetic. In *18th International Confer-*

ence on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-18), volume 7180 of *Lecture Notes in Artificial Intelligence*. Springer, 2012.

[Sut09] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

A Proofs and Auxiliary Results

Proposition 5.1 *Let N be a sufficiently complete clause set and N' be obtained from N by replacing one or more clauses by their weakly abstracted versions. Then sgi(N) and sgi(N') are equivalent, and N and N' are equivalent.*

Proof. Let $C[s_1, \ldots, s_m, t_1, \ldots, t_n] \in N$ a clause that undergoes weak abstraction and let $C' \vee E = C[x_1, \ldots, x_m, X_1, \ldots, X_n] \vee x_1 \not\approx s_1 \vee \cdots \vee x_m \not\approx s_m \vee X_1 \not\approx t_1 \vee \cdots \vee X_n \not\approx t_n$ its weakly abstracted version. It suffices to show that sgi(N) and sgi($(N \setminus \{C\}) \cup \{C' \vee E\}$) are equivalent.

In the one direction let J be any model of sgi(N). In particular, J satisfies every simple ground instance of C. Let $(C' \vee E)\gamma'$ be a simple ground instance of $C' \vee E$. It suffices to show that J satisfies $(C' \vee E)\gamma'$.

If J falsifies all disequations in $E\gamma'$, J satisfies all equations $x_i\gamma' \approx s_i\gamma'$ and $X_j\gamma' = t_j\gamma'$. Because γ' is a simple ground substitution for vars($C' \vee E$), γ' is also a simple ground substitution for vars(C), a subset of the former. Since J satisfies every simple ground instance of C, J satisfies in particular $C\gamma' = (C[s_1\gamma', \ldots, s_m\gamma', t_1\gamma', \ldots, t_n\gamma'])\gamma'$. By congruence, J satisfies $(C[x_1\gamma', \ldots, x_m\gamma', X_1\gamma', \ldots, X_n\gamma'])\gamma' = (C[x_1, \ldots, x_m, X_1, \ldots, X_n])\gamma'$, and, trivially, $(C' \vee E)\gamma'$.

For the other direction let J be any model of sgi(N'). It follows that J satisfies every simple ground instance of $C' \vee E$. Let $C\gamma$ be a simple ground instance of C. We have to show that J satisfies $C\gamma$. Without loss of generality assume $\mathrm{dom}(\gamma) \cap \{x_1, \ldots, x_m, X_1, \ldots, X_n\} = \emptyset$.

Consider the instance $(C' \vee E)\gamma = (C[x_1, \ldots, x_m, X_1, \ldots, X_n])\gamma \vee x_1 \not\approx s_1\gamma \vee \cdots \vee x_m \not\approx s_m\gamma \vee X_1 \not\approx t_1\gamma \vee \cdots \vee X_n \not\approx t_n\gamma$. Because by weak abstraction every t_i is a pure BG term and γ is simple, the substitution $\gamma' := \gamma[x_1 \mapsto s_1\gamma, \ldots, x_m \mapsto s_m\gamma, X_1 \mapsto t_1\gamma, \ldots, X_n \mapsto t_n\gamma]$ is simple as well. Because γ' is a ground substitution for $(C' \vee E)$ it follows that J satisfies $(C' \vee E)\gamma'$ and also, by congruence, $C'\gamma'$. On the other hand, $C'\gamma' = (C[x_1\gamma', \ldots, x_m\gamma', X_1\gamma', \ldots, X_n\gamma'])\gamma' = (C[s_1\gamma, \ldots, s_m\gamma, t_1\gamma, \ldots, t_n\gamma])\gamma = C\gamma$, and so J satisfies $C\gamma$. □

With essentially the same proof one gets that N and N' are equivalent.

The following results are due to Bachmair and Ganzinger[12] and Nieuwenhuis[13]

Theorem A.2 *The ground superposition calculus using the inference rules equality resolution, negative superposition, positive superposition, and equality factoring and $\mathcal{R}^{\mathcal{S}} = (\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}, \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}})$ satisfy the following properties:*

[12] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

[13] R. Nieuwenhuis. First-order completion techniques. Technical report, Universidad Politécnica de Cataluña, Dept. Lenguajes y Sistemas Informáticos, 1991.

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

(ii) *The ground superposition calculus together with* $\mathcal{R}^{\mathcal{S}}$ *is refutationally complete.*

(iii) $N \subseteq N'$ *implies* $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N')$.

(iv) $N' \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(N)$ *implies* $\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(N) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(N \setminus N')$.

Lemma A.3 *If* $sgi(N) \cup \mathrm{GndTh}(\mathcal{B}) \models sgi(C)$, *then* $N \models_{\mathcal{B}} C$.

Proof. Suppose that $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}) \models \mathrm{sgi}(C)$ and let I' be an Σ-model of N whose restriction to Σ_{B} is contained in $\mathcal{B}$. Obviously, 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 exists a ground BG term t, such that t and t' have the same interpretation in I'. Consequently, for every ground substitution σ' there exists an equivalent simple ground substitution σ; since $C\sigma$ is valid in I', $C\sigma'$ is also valid. □

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

Lemma A.4 $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_{\mathrm{Inf}}, \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}})$ *is a redundancy criterion with respect to* $\models_{\mathcal{B}}$.

Proof. We have to check the four conditions of Def. 6.1. The proof of property (ii) is rather trivial. To check property (i) let D be an arbitrary clause from $\mathrm{sgi}(\mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N))$. Consequently, $D \in \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \cup \mathrm{GndTh}(\mathcal{B})$. If $D \in \mathrm{GndTh}(\mathcal{B})$, then trivially $\mathrm{sgi}(N \setminus \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N)) \cup \mathrm{GndTh}(\mathcal{B}) \models D$. Otherwise $D \in \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}))$, and this implies $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \models D$. Since $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \subseteq \mathrm{sgi}(N) \setminus \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \cup \mathrm{GndTh}(\mathcal{B}) = \mathrm{sgi}(N) \setminus (\mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \cup \mathrm{GndTh}(\mathcal{B})) \cup \mathrm{GndTh}(\mathcal{B}) \subseteq \mathrm{sgi}(N \setminus \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N)) \cup \mathrm{GndTh}(\mathcal{B})$, we obtain again $\mathrm{sgi}(N \setminus \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N)) \cup \mathrm{GndTh}(\mathcal{B}) \models D$. We can conclude that $N \setminus \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N) \models_{\mathcal{B}} \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N)$.

Condition (iii) is obviously satisfied for all Close inferences. Suppose that ι is not a Close inference and its conclusion $\mathrm{concl}(\iota)$ is in N. Then $\mathrm{concl}(\iota) = \mathrm{abstr}(C) = C[x_1, \ldots, x_m, X_1, \ldots, X_n] \vee x_1 \not\approx s_1 \vee \cdots \vee x_m \not\approx s_m \vee X_1 \not\approx t_1 \vee \cdots \vee X_n \not\approx t_n$, where $C = C[s_1, \ldots, s_m, t_1, \ldots, t_n]$. Now consider a simple ground instance ι' of ι with maximal premise $C_1\gamma$ and conclusion $C\gamma = C\gamma[s_1\gamma, \ldots, s_m\gamma, t_1\gamma, \ldots, t_n\gamma]$. By the structure of superposition inferences, the clause $C\gamma$ is obtained from $C_1\gamma$ by replacing (one occurrence of) the maximal literal of $C_1\gamma$ by (zero or more) smaller literals. Since the terms $s_i\gamma$ and $t_j\gamma$ are proper subterms of terms occurring in $C\gamma$, each of the literals $s_i\gamma \not\approx s_i\gamma$ and $t_j\gamma \not\approx t_j\gamma$ is also smaller than the deleted literal of $C_1\gamma$. Consequently, the simple ground instance D of $\mathrm{abstr}(C)$ that has the form $D = C\gamma[s_1\gamma, \ldots, s_m\gamma, t_1\gamma, \ldots, t_n\gamma] \vee s_1\gamma \not\approx s_1\gamma \vee \cdots \vee s_m\gamma \not\approx s_m\gamma \vee t_1\gamma \not\approx t_1\gamma \vee \cdots \vee t_n\gamma \not\approx t_m\gamma$ is still smaller than $C_1\gamma$. Moreover

D entails $C\gamma$, so $\iota' \in \mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(\mathrm{sgi}(N))$. As $\mathrm{sgi}(\iota) \subseteq \mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(\mathrm{sgi}(N))$, the inference ι is contained in $\mathcal{R}_{\mathrm{Inf}}^{\mathcal{H}}(N)$. This proves condition (iii).

We come now to the proof of condition (iv). Note that $N' \subseteq \mathcal{R}_{\mathrm{Cl}}^{\mathcal{H}}(N)$ implies $\mathrm{sgi}(N') \subseteq \mathcal{R}_{\mathrm{Cl}}^{\mathcal{S}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \cup \mathrm{GndTh}(\mathcal{B})$, and thus $\mathrm{sgi}(N') \setminus \mathrm{GndTh}(\mathcal{B}) \subseteq \mathcal{R}_{\mathrm{Cl}}^{\mathcal{S}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}))$. If $\iota \in \mathcal{R}_{\mathrm{Inf}}^{\mathcal{H}}(N)$ is a Close inference, then $\square \in N$; since $\square \notin \mathcal{R}_{\mathrm{Cl}}^{\mathcal{H}}(N)$, ι is contained in $\mathcal{R}_{\mathrm{Inf}}^{\mathcal{H}}(N \setminus N')$. Otherwise $\mathrm{sgi}(\iota) \subseteq \mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \subseteq \mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}) \setminus (\mathrm{sgi}(N') \setminus \mathrm{GndTh}(\mathcal{B}))) = \mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(\mathrm{sgi}(N) \setminus \mathrm{sgi}(N') \cup \mathrm{GndTh}(\mathcal{B})) \subseteq \mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(\mathrm{sgi}(N \setminus N') \cup \mathrm{GndTh}(\mathcal{B}))$, hence ι is again contained in $\mathcal{R}_{\mathrm{Inf}}^{\mathcal{H}}(N \setminus N')$. Therefore $\mathcal{R}_{\mathrm{Inf}}^{\mathcal{H}}(N) \subseteq \mathcal{R}_{\mathrm{Inf}}^{\mathcal{H}}(N \setminus N')$. $\square$

Sufficient completeness w.r.t. simple instances is preserved by adding clauses to a set of clauses or by deleting redundant clauses:

Lemma A.5 *Let N, N' and M be sets of abstracted clauses such that $N' \subseteq \mathcal{R}_{\mathrm{Cl}}^{\mathcal{H}}(N)$. If N is sufficiently complete w.r.t. simple instances, then so are $N \cup M$ and $N \setminus N'$.*

Proof. The sufficient completeness of $N \cup M$ is obvious; the sufficient completeness of $N \setminus N'$ is proved in a similar way as in part (i) of the proof of Lemma A.4. $\square$

Lemma A.6 *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 $1 \leq i \leq n$.*

Lemma A.7 *If N is a set of abstracted clauses, then $\mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(\mathit{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \subseteq \mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(N_I)$.*

Proof. By part (iii) of Thm. A.2 we have obviously $\mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(\mathrm{sgi}(N)) \subseteq \mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}))$. 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 A.6 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 $\mathcal{R}_{\mathrm{Cl}}^{\mathcal{S}}(\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N))$. 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 x to the E_I-normal form of $x\sigma$. Since C follows from smaller clauses in $\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N)$, it is in $\mathcal{R}_{\mathrm{Cl}}^{\mathcal{S}}(\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N))$. Hence $\mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \subseteq \mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(N_I)$. $\square$

Theorem 6.3 *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 $\mathit{sgi}(N)$ 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 redundant with respect to N_I, i. e., that it is contained in $\mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(N_I)$. 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 A.6 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 $\mathcal{R}_{\mathrm{Cl}}^{\mathcal{S}}(N_I)$; every inference involving such a clause is in $\mathcal{R}_{\mathrm{Inf}}^{\mathcal{S}}(N_I)$.

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 is a BG clause, then ι is in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$. If C is a FG clause, then 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 $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}))$, again this implies $\iota \in \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$.

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 is a BG clause, then ι is in $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$. Otherwise, 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). We have to distinguish between two cases: 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 $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$. □

Theorem 6.4 *If the BG specification $(\Sigma, \mathcal{B})$ is compact, then the hierarchic superposition calculus and $\mathcal{R}^{\mathcal{H}}$ are refutationally complete for all sets of clauses that are sufficiently complete w.r.t. simple instances.*

Proof. Let N be a set of weakly abstracted clauses that is sufficiently complete w.r.t. simple instances, and saturated w.r.t. the hierarchic superposition calculus and $\mathcal{R}^{\mathcal{H}}$ and does not contain □. 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 Thm. 6.3, 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 w.r.t. simple instances, 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 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 w.r.t. simple instances, preventing junk. Since I' satisfies N and $I'|_{\Sigma_{\mathrm{B}}} \in \mathcal{B}$, we have $N \not\models_{\mathcal{B}} \square$ □

Theorem 7.4 *The* HSP *calculus with a suitable* Simp *inference rule and a specific strategy is refutationally complete for the ground BG-sorted term fragment. More precisely, let N be a clause set such that every BG-sorted term in N is ground. Let $N_0 = \mathrm{abstr}(N)$ and $N_0, \ldots, N_k$ a derivation such that every $C \in N_k$ does not contain any BG-sorted FG operator or $\mathrm{unabstr}(C)$ is a ground positive unit clause of the form $f(t_1, \ldots, t_n) \approx t$ where f is a BG-sorted FG operator and $t_1, \ldots, t_n, t$ do not contain BG-sorted FG operators.*

If N is $\mathcal{B}$-unsatisfiable then there there is a refutation of N_k without the Define *inference rule.*

Proof. Let $\mathcal{D}$ be a derivation from N_k. Supposing $\mathcal{D}$ is not a refutation we need to show that N has a $\mathcal{B}$-model.

First we show that $\mathcal{D}$ derives only finitely many BG clauses, modulo equivalence. More precisely, suppose $\mathcal{D} = (N_i)_{i \geq k}$ and let $N^\infty = \bigcup_{i \geq k} N_i$ be the set of all derived clauses. We show that N^∞ contains only finitely many BG clauses, modulo equivalence. For that, it suffices to show that the equivalent set $N^\infty_{\text{unabstr}} = \{\text{unabstr}(C) \mid C \in N^\infty\}$ contains only finitely many BG clauses, modulo equivalence. For that, it suffices to show that in $N^\infty_{\text{unabstr}}$ only finitely many ground BG terms occur. The latter is sufficient, because every BG clause in the GBT fragment is a multiset of literals of the form $s \approx t$ or $s \not\approx t$, where s and t are ground BG terms, there are only finitely many such literals over a finite set of ground BG terms, and $N^\infty_{\text{unabstr}}$ belongs to the GBT fragment.

To complete the proof of the first step it remains to show that in $N^\infty_{\text{unabstr}}$ only finitely many ground BG terms occur. Because Define is disabled in $\mathcal{D}$, only Simp and the remaining inference rules need to be analysed. Regarding the latter, it is straightforward to show that if C is the conclusion from premises $C_1, \ldots, C_n$ then $\text{unabstr}(C)$ cannot contain new BG terms w.r.t. any of $\text{unabstr}(C_1), \ldots, \text{unabstr}(C_n)$. The syntactic shape of definitions $\mathsf{f}(t_1, \ldots, t_n) \approx t$ in $\text{unabstr}(N_k)$ where f is a BG-sorted FG operator and $t_1, \ldots, t_n, t$ do not contain BG-sorted FG operators is essential for that. More precisely, BG-sorted FG operators occur in N_k only in top-positions, and this property is preserved by all inference rules. In particular, all superposition steps into non-top positions must be into FG sorted subterms. Although a new BG-sorted FG operator term may result from that, the (unabstracted) conclusion will contain no new BG term. All superposition inferences into a term with BG-sorted FG operator (at top level, hence) will result in a (dis)equation between BG terms, which never requires unabstraction and hence does not generate a new BG term either.[14] Simplification behaves in the same way due to condition (iii) of suitability. But then, the BG terms occuring in $\text{unabstr}(N_k)$ provide an upper bound w.r.t. $\succ$ for all terms generated in Simp inferences, of which there can be only finitely many by suitability of Simp, condition (ii).

Because $\mathcal{D}$ is not a refutation, the set of simple ground instances of its saturated limit clause set has a Σ-model I. Notice that compactness of the BG theory is not an issue here. Compactness is needed only in the proof of Thm. 6.4 to conclude the satisfiability of the set of all BG clauses in the saturated set from non-applicability of Close. Because $\mathcal{D}$ derives only finitely many BG clauses, modulo equivalence, non-applicability of Close entails that trivially.

The model I is also a Σ-model of all simple ground instances of all clauses in N_k. Recall that N_k is partitioned into two subsets, one containing only clauses without occurrences

[14] In Sect. 6 we gave an example that demonstrates the need for unabstraction as part of forming the conclusion of superposition inferences. By that, a the new BG term $1 + 1$ is generated. On the other hand, the premise clause $\mathsf{P}(\mathsf{f}(1) + 1)$ is impossible in our case.

of BG-sorted FG symbols, and the other containing definitions of the stated form only. This entails that I is also a Σ-model of all simple ground instances of all clauses in N and that it maps every BG-sorted FG term occurring in N to an element of the carrier set of the proper sort.

We now change I to a Σ-interpretation I' without junk by redefining the interpretation of operator symbols in such a way that every BG-sorted FG term not occurring in N is also mapped to an element of the carrier set of the proper sort. Notice that trivially I' is still a Σ-model of all simple ground instances of all clauses in N. With the same arguments as in the proof of Thm. 6.4, I' is a Σ-model of all ground instances of all clauses in N. Moreover, I' contains no junk and therefore is a $\mathcal{B}$-model of N, which completes the proof. □