Hierarchic Superposition With Weak Abstraction

Peter Baumgartner
Uwe Waldmann

MPI–I–2013–RG1–002 June 2013

Authors' Addresses

Peter Baumgartner
NICTA and Australian National University
Tower A, 7 London Circuit
Canberra ACT 2601, Australia

Uwe Waldmann
Max-Planck-Institut für Informatik
Campus E 1 4
66123 Saarbrücken, Germany

Publication Notes

A shortened version of this research report will appear in M. P. Bonacina (ed.), *24th International Conference on Automated Deduction – CADE-24*, LNCS 7898, Springer-Verlag.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

We would like to thank the anonymous reviewers of CADE-24 for their helpful comments.

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.

Keywords

Hierarchic theorem proving, superposition.

Contents

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 [18] for the quantifier-free case by instantiation heuristics for quantifiers [11, 12, e. g.]. Another line of research is concerned with adding black-box reasoners for specific background theories to first-order automated reasoning methods (resolution [5, 13, 1], sequent calculi [21], instantiation methods [10, 7, 8], 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.

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 hierarchical superposition 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 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. 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 , 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 . 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. 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 [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 a specific completeness result for clause sets over ground background-sorted terms that does not require compactness. We also report on experiments with a prototypical implementation on the TPTP problem library.

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 specialised version of hierarchic superposition in [14] will be discussed in Sect. 8 below. The resolution calculus in [13] 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 [7, 8] lacks the treatment of background-sorted foreground function symbols. The same applies to the sequent calculus in [21], which treats linear arithmetic with built-in rules for quantifier elimination. The instantiation method in [10] 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 [15, 2, 9]. 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 [15], 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 $P(t_1, \ldots, t_n) \approx \mathit{true}_P$; 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 $\square$ 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 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 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

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

domain (at least) 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.

[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.

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 $\Omega^D_{\mathrm{B}} \subseteq \Omega_{\mathrm{B}}$ that has the property that $d_1^I \neq d_2^I$ for any two distinct $d_1, d_2 \in \Omega^D_{\mathrm{B}}$ 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.[1] 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 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 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*.

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 $N \models_\mathcal{B} \Box$, we call N *$\mathcal{B}$-unsatisfiable*; otherwise, we call it *$\mathcal{B}$-satisfiable*.[2]

[1]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.

[2]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.

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 is *simple* if it maps every abstraction variable in its domain to a *pure* BG term. 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.

Standard unification algorithms can be modified in a straightforward way for computing simple mgus. Note that a simple mgu can map an ordinary variable to an abstraction variable but not vice versa, as ordinary variables are not pure BG terms. An impure BG term t and an abstraction variable X can be unified, but the unifier has to map the ordinary variables in t to new abstraction variables; this is similar to the use of weakening substitutions in many-sorted logics with subsorts [24]. For example, a most general unifier of $x + y$ and Z is $[x \mapsto X,\, y \mapsto Y,\, Z \mapsto X + Y]$.

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 *iff for every Σ-model J of* $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})$[3] *and every ground BG-sorted FG term s there is a ground BG term t such that $J \models s \approx t$.*[4]

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.

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

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

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 $x \in \text{vars}(s)$. 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 $s \succeq t$ 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

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 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.

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. The approach used in [5] to reduce this problem is to consider 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 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. We need a slightly different abstraction rule to avoid this problem:

Definition 5.1 *A BG term q occurring in a clause C is called* target term *if q is neither a domain element nor a variable and if C has the form $C[f(s_1, \dots, q, \dots, s_n)]$, where f is a FG operator or at least one of the s_i is a FG or impure BG term.*[1]

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 C is denoted by $\mathrm{abstr}(C)$*.*

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 $\mathsf{write}(\mathsf{a}, 2, \mathsf{read}(\mathsf{a}, 1) + 1) \approx$

[1]The motivation for this definition and in particular the reason why it is permissible to treat domain elements in a special way will become clear in Sect. 7.

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 z, 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 Def. 5.1 does in fact terminate. For any clause, we consider the multiset of the numbers of non-variable occurrences in the left and right-hand sides of its literals: If we abstract out a target term q, then q has $k \geq 1$ non-variable occurrences and it occurs as a subterm of a left or right-hand side $s[q]$ with $n > k$ non-variable occurrences. After the abstraction step, $s[\zeta]$ has $n - k$ non-variable occurrences and the two terms ζ and q in the new abstraction literal have 0 and k non-variable occurrences. Since $n - k$, k, and 0 are strictly smaller than n, the multiset decreases. Termination follows from the well-foundedness of the multiset ordering.

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.

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. $\square$

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 $N = \{\, \neg\mathsf{f}(x) > \mathsf{g}(x) \vee \mathsf{h}(x) \approx 1,\ \neg\mathsf{f}(x) \le \mathsf{g}(x) \vee \mathsf{h}(x) \approx 2,\ \neg\mathsf{h}(x) > 0 \,\}$ 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 $X > Y \vee X \le Y$ 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 $x > y \vee x \le y$, 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 abstrac-*

tion 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 Prop. 5.2, we know already that $\mathrm{sgi}(N)$ and $\mathrm{sgi}(N')$ are equivalent. Moreover, it is easy to check the proof of Prop. 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 the latter, we can conclude that N' is sufficiently complete whenever N'' is. $\square$

In the rest of the paper we will need some technical lemmas that relate clauses, their (partial) abstractions, instances of these clauses, and the target terms in these clauses.

Lemma 5.4 *Let w be a subterm of a literal $L = [\neg]\, v[w] \approx v'$. Let σ be a simple substitution and let K be a literal $[\neg]\, v\sigma[w\sigma] \approx v''$. If $w\sigma$ is a target term in K, then w is a variable or a target term in L.*

Proof. If w is neither a variable nor a target term in L, then w is a FG term or a domain element, or it equals v, or it occurs below a BG operator f and all other terms occurring below f are pure BG terms. Each of these properties is preserved by instantiation with a simple substitution. (Note that simple instances of pure BG terms are again pure BG terms.) $\square$

Lemma 5.5 *Let w be a subterm of a literal $L = [\neg]\, v[w] \approx v'$. Let σ be a substitution that maps all abstraction variables in its domain to pure BG terms and all ordinary BG-sorted variables in its domain to impure BG terms, and let K be a literal $[\neg]\, v\sigma[w\sigma] \approx v''$. If w is a target term in L then $w\sigma$ is a target term in K.*

Proof. The term w is a target term in L if and only if (i) w is a BG term that is neither a domain element nor a variable, and (ii) w occurs in L in a term $f(s_1, \ldots, w, \ldots, s_n)$, where f is a FG operator or at least one of the s_i is a FG or impure BG term. Obviously $w\sigma$ cannot be a domain element or a variable. Moreover it is easy to check that σ maps BG terms to BG terms, impure BG terms to impure BG terms, and FG terms to FG terms, so both properties (i) and (ii) are preserved by σ. $\square$

Lemma 5.6 *Let D_0 be a clause. Let τ_0 be a simple substitution such that $D_0\tau_0$ is a ground instance of D_0 and let ρ_0 be the identity substitution. For $n \in \{0, \ldots, k{-}1\}$ let D_{n+1} be clauses obtained from D_0 by successively abstracting out target terms as described in Def. 5.1, that is, let q_n be a target term in $D_n = D_n[q_n]$ and let $D_{n+1} = D_n[\zeta_n] \vee \zeta_n \not\approx q_n$ (where ζ_n is a new abstraction variable, if q_n is a pure target term, and a new ordinary variable otherwise). Let $\tau_{n+1} = \tau_n[\zeta_n \mapsto q_n\tau_n]$ and $\rho_{n+1} = \rho_n[\zeta_n \mapsto q_n\rho_n]$. Then the following properties hold:*

(1) If $n \in \{0, \ldots, k\}$, then τ_n and ρ_n are simple substitutions and ρ_n maps all abstraction variables in its domain to pure BG terms and all ordinary variables in its domain to impure BG terms.

(2) If $n \in \{0, \ldots, k\}$, then $\tau_n = \rho_n\tau_0$.

(3) If $n \in \{0, \ldots, k\}$ then $D_n\tau_n$ is a ground instance of D_n and has the form $D_n\tau_n = D_0\tau_0 \vee \bigvee_{0\le i<n} q_i\tau_i \not\approx q_i\tau_i$, where the literals $q_i\tau_i \not\approx q_i\tau_i$ are ground instances of the abstraction literals introduced so far.

(4) If $n \in \{0, \ldots, k\}$ and if D'_n is the subclause of D_n that is obtained by dropping the abstraction literals introduced so far, then $D'_n\rho_n = D_0$.

(5) If $n \in \{0, \ldots, k{-}1\}$ and if the target term q_n occurs in the subclause D'_n of D_n, then there is a target term $\bar{q}_n$ in D_0 such that $q_n\tau_n = \bar{q}_n\tau_0$.

Proof. Property (1) follows by induction from the fact that all the mappings $[\zeta_n \mapsto q_n\tau_n]$ and $[\zeta_n \mapsto q_n\rho_n]$ are simple and from the fact that $[\zeta_n \mapsto q_n\rho_n]$ maps an abstraction variable to a pure BG term or an ordinary variable to an impure BG term.

Property (2) is obvious for $n = 0$. By induction, we obtain $\tau_{n+1} = \tau_n[\zeta_n \mapsto q_n\tau_n] = \rho_n\tau_0[\zeta_n \mapsto q_n\rho_n\tau_0] = \rho_n[\zeta_n \mapsto q_n\rho_n]\tau_0 = \rho_{n+1}\tau_0$ as required.

Property (3) is again obvious for $n = 0$. By induction, we obtain $D_{n+1}\tau_{n+1} = D_n\tau_{n+1}[\zeta_n\tau_{n+1}] \vee \zeta_n\tau_{n+1} \not\approx q_n\tau_{n+1} = D_n\tau_n[q_n\tau_n] \vee q_n\tau_n \not\approx q_n\tau_n = D_n\tau_n \vee q_n\tau_n \not\approx q_n\tau_n = D_0\tau_0 \vee \bigvee_{0\le i<n} q_i\tau_i \not\approx q_i\tau_i \vee q_n\tau_n \not\approx q_n\tau_n = D_0\tau_0 \vee \bigvee_{0\le i<n+1} q_i\tau_i \not\approx q_i\tau_i$ as required. Since q_n is a subterm of D_n and $D_n\tau_n$ is ground by induction, we can conclude that $q_i\tau_n \not\approx q_i\tau_n$ is ground as well.

To prove property (4), we write D_n in the form $D_n = D'_n \vee E_n$, where E_n is the subclause consisting of all abstraction literals introduced so far. For $n = 0$, there is nothing to prove. If q_n occurs in D'_n, then $D_n = D'_n[q_n] \vee E_n$ and $D_{n+1} = D'_n[\zeta_n] \vee E_n \vee \zeta_n \not\approx q_n$, therefore $D'_{n+1} = D'_n[\zeta_n]$ and $D'_{n+1}\rho_{n+1} = D'_n\rho_{n+1}[\zeta_n\rho_{n+1}] = D'_n\rho_n[q_n\rho_n] = D'_n\rho_n = D_0$. Otherwise q_n occurs in E_n, then $D_n = D'_n \vee E_n[q_n]$ and $D_{n+1} = D'_n \vee E_n[\zeta_n] \vee \zeta_n \not\approx q_n$, therefore $D'_{n+1} = D'_n$ and $D'_{n+1}\rho_{n+1} = D'_n\rho_{n+1} = D'_n\rho_n = D_0$.

It remains to prove property (5). According to property (4) ρ_n maps D'_n to D_0; since q_n occurs in D'_n, $q_n\rho_n$ is a subterm $\bar{q}_n$ of D_0. By property (1) and Lemma 5.5, $\bar{q}_n$ must be a target term in D_0. Property (2) yields $\bar{q}_n\tau_0 = q_n\rho_n\tau_0 = q_n\tau_n$. □

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. All inference rules are applicable only to weakly abstracted premise clauses. 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.

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

if (i) neither s nor t is a pure BG term, (ii) σ is a simple mgu of s and t, and (iii) 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 $(s \not\approx t \vee C)\sigma$.[1]

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.

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

if (i) 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) 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 $(s \not\approx t \vee D)\sigma$.

[1]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).

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

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) $t\sigma \not\succeq s\sigma$, (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) neither s nor l is a pure BG term, (ii) σ is a simple mgu of s and l, (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.

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 6.1 *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.*

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 [17], Nieuwenhuis and Rubio [19]). 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 $C_1, \dots, C_n \in N$ that are smaller than C with respect to $\succ$ and $C_1, \dots, C_n \models C$. 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 $C_1, \dots, C_n \in N$ that are smaller with respect to $\succ^{\mathrm{c}}$ than the maximal premise of ι and $C_1, \dots, C_n \models C_0$.

The following results can be found in [3] and [17]:

Theorem 6.2 *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.*

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

Let ι be an $\mathrm{HSP}_{\mathrm{Base}}$ inference with premises $C_1, \dots, C_n$ and conclusion $\mathrm{abstr}(C)$, where the clauses $C_1, \dots, C_n$ have no variables in common. Let ι' be a ground SSP inference with premises $C'_1, \dots, C'_n$ 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 6.3 *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* $\mathrm{sgi}(C) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Cl}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \cup \mathrm{GndTh}(\mathcal{B})$.[2]

[2]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].)

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 for both properties 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 $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}) \models \mathrm{sgi}(C)$ 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. $\square$

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.

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

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

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

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

Then there is a simple ground instance of $\operatorname{abstr}(C_0\sigma)$ *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'.*

Proof. Without loss of generality, we may assume that σ is an idempotent most general unifier over $\operatorname{vars}(C_1) \cup \operatorname{vars}(C_2)$ and that $\tau = \sigma\rho$ for some substitution ρ; hence $\sigma\tau = \sigma\sigma\rho = \sigma\rho = \tau$ and in particular $C_0\sigma\tau = C_0\tau$.

Let $\tau_0 = \tau$ and let $D_0 = C_0\sigma$. Let D_n be the result of the nth abstraction step starting from D_0 for $n \in \{0, \ldots, k\}$, and let $\operatorname{abstr}(C_0\sigma) = D_k$. According to part (3) of Lemma 5.6, there are simple substitutions τ_n for $1 \leq n \leq k$ such that $D_n\tau_n$ is a ground instance of D_n and has the form $D_n\tau_n = D_0\tau_0 \vee \bigvee_{0 \leq i < n} q_i\tau_i \not\approx q_i\tau_i$, where the literals $q_i\tau_i \not\approx q_i\tau_i$ are ground instances of the abstraction literals introduced so far and q_i is a target term in D_i. Since $C_0\sigma\tau = D_0\tau = D_0\tau_0$, this clause has essentially the required form. We still have to prove, though, that each literal $q_i\tau_i \not\approx q_i\tau_i$ is smaller than the pivotal literal L of ι'.

If q_i occurs in D_i in a literal K that has been generated by the jth abstraction step ($j < i$), then $K\tau_j$ is a literal $q_j\tau_j \not\approx q_j\tau_j$ and $q_i\tau_i$ is a proper subterm of $q_j\tau_j$. By induction on the number of abstraction steps we obtain $q_i\tau_i \not\approx q_i\tau_i \prec q_j\tau_j \not\approx q_j\tau_j \prec L$.

If q_i occurs in D_i in a literal that has not been generated by a previous abstraction step, then by part (5) of Lemma 5.6 there is a target term $\bar{q}_i$ in $D_0 = C_0\sigma$ such that $q_i\tau_i = \bar{q}_i\tau_0 = \bar{q}_i\tau$. The term $\bar{q}_i$ is either a term from $C_1\sigma$, or a term from $C_2\sigma$, or a subterm of the term $s[r]\sigma$ (the last two cases are possible only for superposition inferences). We analyze these cases separately.

Case 1: $\bar{q}_i$ is a proper subterm of a term $v\sigma$, where $[\neg]v\sigma \approx v''$ is a literal of $C_0\sigma$ and $[\neg]v \approx v'$ is a literal of C_1.

Case 1.1: C_1 does not have selected literals. In this case, every literal of $C_1\tau$ is smaller than or equal to the pivotal literal L of ι'. Consequently, we obtain $q_i\tau_i = \bar{q}_i\tau \prec v\sigma\tau = v\tau$, thus $q_i\tau_i \not\approx q_i\tau_i \prec [\neg]\, v\tau \approx v'\tau \preceq L$.

Case 1.2: C_1 has selected literals. In this case, L is a selected literal in $C_1\tau$ and ι and ι' are either Negative superposition or Equality resolution inferences. Since $\bar{q}_i$ is a target term in $C_0\sigma$, it is not a variable.

If $\bar{q}_i$ occurs in $v\sigma$ below a variable position of v, then there is a $\zeta \in \operatorname{dom}(\sigma) \cap \operatorname{vars}(C_1)$ such that $\bar{q}_i$ is a subterm of $\zeta\sigma$. Otherwise, there is a subterm w of v such that $\bar{q}_i = w\sigma$. Since C_1 is weakly abstracted, w cannot be a target term, so by Lemma 5.4, w must be a variable ζ, and again $\zeta \in \mathrm{dom}(\sigma) \cap \mathrm{vars}(C_1)$. So in both cases, $\bar{q}_i$ is a subterm of $\zeta\sigma$, and consequently, $q_i\tau_i = \bar{q}_i\tau$ is a subterm of $\zeta\sigma\tau = \zeta\tau$.

Let us first consider a Negative superposition inference operating on the literal $s[u] \not\approx t$ in C_1, where $L = (s[u] \not\approx t)\tau$. As σ is a simple most general unifier of l and u and $\zeta \in \mathrm{dom}(\sigma) \cap \mathrm{vars}(C_1)$, we know that ζ must occur in u. Moreover u is not a variable, so ζ is a proper subterm of u. This implies $q_i\tau_i \preceq \zeta\tau \prec u\tau \preceq s[u]\tau$, hence $q_i\tau_i \not\approx q_i\tau_i \prec (s[u] \not\approx t)\tau = L$.

Otherwise ι is an Equality resolution inference operating on the literal $s \not\approx t$ in C_1, where $L = (s \not\approx t)\tau$. As σ is a simple most general unifier of s and t and $\zeta \in \mathrm{dom}(\sigma) \cap \mathrm{vars}(C_1)$, we know that ζ must occur in s or t. Assume without loss of generality that ζ occurs in s. If $s = \zeta$, then by the restrictions on selection functions t must be a FG term, so $\zeta\tau = t\tau$ is a FG term as well. Since the BG term $q_i\tau_i$ is a subterm of the FG term $\zeta\tau$, it must be a proper subterm, hence $q_i\tau_i \prec \zeta\tau = t\tau$ and $q_i\tau_i \not\approx q_i\tau_i \prec (s \not\approx t)\tau = L$. Otherwise ζ is a proper subterm of s, then $q_i\tau_i \preceq \zeta\tau \prec s\tau = t\tau$ and we obtain again $q_i\tau_i \not\approx q_i\tau_i \prec (s \not\approx t)\tau = L$.

Case 2: $\bar{q}_i$ is a proper subterm of a term $v\sigma$, where $[\neg]v\sigma \approx v''$ is a literal of $C_0\sigma$ and $[\neg]v \approx v'$ is a literal of C_2. This case is proved similarly to case 1.1 above: By the structure of SSP inferences, the literal $l\tau \approx r\tau$ of $C_2\tau$ that has been used to replace $s\tau[l\tau]$ by $s\tau[r\tau]$ in the pivotal literal is strictly maximal in $C_2\tau$ and $l\tau \succ r\tau$. Consequently, we obtain $q_i\tau_i = \bar{q}_i\tau \prec v\sigma\tau = v\tau \preceq l\tau \preceq s\tau[l\tau]$ and thus $q_i\tau_i \not\approx q_i\tau_i \prec [\neg]\, s\tau[l\tau] \approx t\tau = L$.

Case 3: It remains to consider the case that $\bar{q}_i$ is a subterm of the term $s[r]\sigma$ produced in a superposition inference. Then $q_i\tau_i = \bar{q}_i\tau \prec s[r]\sigma\tau = s[r]\tau \prec s[l]\tau$, hence $q_i\tau_i \not\approx q_i\tau_i \prec [\neg]\, s[l]\tau \approx t\tau = L$. $\square$

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 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 $\models_{\mathcal{B}}$.*

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 $\mathrm{sgi}(N \setminus \mathcal{R}^{\mathcal{H}}_{\mathrm{Cl}}(N)) \cup \mathrm{GndTh}(\mathcal{B}) \models D$. By Lemma 7.1, 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 $\mathrm{concl}(\iota) = \mathrm{abstr}(C_0)$ is in N. Showing that $\iota \in \mathcal{R}^{\mathcal{H}}_{\mathrm{Inf}}(N)$ amounts to proving that every simple ground instance of ι is redundant w. r. t. $\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})$. Let ι' be such a simple ground instance with maximal premise $C_1\tau$ and conclusion $C_0\tau$. By Lemma 7.2, there is a simple ground instance of $\mathrm{abstr}(C_0)$ that has the form $C_0\tau \vee E$, 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'.

By the structure of superposition inferences, the clause $C_0\tau$ is obtained from $C_1\tau$ by replacing the pivotal literal in $C_1\tau$ by (zero or more) smaller literals. Since the literals in E are also smaller than the pivotal literal, $C_0\tau \vee E$ is still smaller than $C_1\tau$. Moreover, $C_0\tau \vee E$ entails $C_0\tau$, so $\iota' \in \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}))$. As $\mathrm{sgi}(\iota) \subseteq \mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}))$, the inference ι is contained in $\mathcal{R}^{\mathcal{H}}_{\mathrm{Inf}}(N)$. This proves condition (iii).

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

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 $N \setminus N'$.*

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 7.3. □

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 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.[1]

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 $C_1, \dots, C_n$ in $\mathrm{E}_I \cup \mathrm{D}_I$ such that $C_1, \dots, C_n \models C$ and $C \succeq C_i$ for $1 \leq i \leq n$.*

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 7.6 *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 (iii) of Thm. 6.2 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 $\mathcal{R}^\mathcal{S}_\mathrm{Inf}(\mathrm{E}_I \cup \mathrm{D}_I \cup \mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B})) \subseteq \mathcal{R}^\mathcal{S}_\mathrm{Inf}(N_I)$. ☐

Theorem 7.7 *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 redundant with respect to N_I, i. e., that it 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 $\mathcal{R}^\mathcal{S}_\mathrm{Inf}(N_I)$.

[1]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 $\alpha \approx 4$.

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 $C\sigma$ 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 $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(\mathrm{sgi}(N) \cup \mathrm{GndTh}(\mathcal{B}))$, by Lemma 7.6, 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, 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 $\mathcal{R}^{\mathcal{S}}_{\mathrm{Inf}}(N_I)$. □

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 Def. 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 Thm. 7.7, 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 7.8 *If the BG specification* $(\Sigma_\mathrm{B}, \mathcal{B})$ *is compact, then* $\mathrm{HSP}_\mathrm{Base}$ *and* $\mathcal{R}^\mathcal{H}$ *are refutationally complete for all sets of clauses that are sufficiently complete.*

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 $\Box$. 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. 7.7, N_I is saturated with respect to the standard superposition calculus. Since $\Box \notin N_I$, 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 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 $N \not\models_\mathcal{B} \Box$ $\Box$

We do not spell out in detail theorem proving *processes* here, because the well-known framework of standard resolution [4] 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 $(2 + 3) + \alpha$ 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 $C[\zeta \mapsto d]$.

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 Sufficient Completeness by Define

In this section we introduce an additional inference rule, Define. It augments the $\mathrm{HSP}_{\mathrm{Base}}$ inference system with complementary functionality: while the $\mathrm{HSP}_{\mathrm{Base}}$ inference system will derive a contradiction if the input clause set is inconsistent and sufficiently complete, the Define rule may turn input clause sets that are not sufficiently complete into 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. This way, sufficient completeness is achieved "locally" for t, by forcing t to be equal to some element of the carrier set of the proper sort, denoted by the parameter α. 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 8.1 (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 $\zeta \in \mathrm{vars}(t)$.*

Every clause can be unabstracted by repeatedly replacing $C \vee \zeta \not\approx t$ by $C[\zeta \mapsto t]$ whenever $t = \zeta$ or $\zeta \notin \mathrm{vars}(t)$. By $\mathrm{unabstr}(C)$ we denote an unabstracted version of C that can be obtained this way.[1] 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 $\mathrm{unabstr}(t, C)$. For a clause set N let $\mathrm{unabstr}(N) = \{\mathrm{unabstr}(C) \mid C \in N\}$.

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

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

[1]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.

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. Notice the Define-rule preserves $\mathcal{B}$-satisfiability, not $\mathcal{B}$-equivalence. In our main application, Thm. 8.8 below, every ζ_i will always be an abstraction variable.

Example 8.2 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 $\{\alpha_{\mathsf{f}(0)} > 0,\ 0 > \alpha_{\mathsf{f}(0)}\}$.

In practice, 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. The *ground BG-sorted term fragment (GBT fragment)* discussed below is one such case.

Definition 8.3 *A clause C is a GBT clause if all BG-sorted terms in C are ground. A clause set N belongs to the GBT fragment if every clause $C \in N$ is a GBT clause.*

To get the desired completeness result we need to establish that the Define rule preserves the GBT property.

Lemma 8.4 *If* $\mathrm{unabstr}(N)$ *belongs to the GBT fragment and N' is obtained from N by a* Define *inference, then* $\mathrm{unabstr}(N')$ *also belongs to the GBT fragment.*

Below we will equip the HSP calculus with a specific strategy that first applies Define exhaustively before the derivation proper starts. In both phases, it may be beneficial to also apply Simp. But then, we have to to make certain (mild) assumptions.

Definition 8.5 *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.*[2] *We say that a* Simp *inference with premise $N \cup \{C\}$ and conclusion $N \cup \{D\}$ is* suitable (for the GBT fragment) *iff (i) if* $\mathrm{unabstr}(C)$ *is a GBT clause then* $\mathrm{unabstr}(D)$ *is a GBT clause, (ii) for every BG term t occurring in* $\mathrm{unabstr}(D)$ *there is a BG term $s \in$* $\mathrm{unabstr}(C)$ *such that $s \succeq_{\mathrm{fin}} t$, (iii) every occurrence*

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

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, (iv) every BG term in D is pure, and (v) 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 *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. $\succeq_{\mathrm{fin}}$, as expressed in condition (ii). Without it, e. g., the clause $\mathsf{P}(1+1, 0)$ would admit infinitely many simplified versions $\mathsf{P}(2, 0)$, $\mathsf{P}(2, 0+0)$, $\mathsf{P}(2, 0+(0+0))$, etc. Condition (i) makes sure that Simp inferences preserve GBT clauses. Condition (iii) is needed to make sure that no new BG terms are generated in derivations.

As said, we need to equip the HSP calculus with a specific strategy. Assume a suitable Simp rule and let N be a set of GBT clauses. By N^{pre} we denote any clause set obtained by a derivation of the form $(N_0 = \mathrm{abstr}(N)), N_1, \ldots, (N_k = N^{\mathrm{pre}})$ with the inference rules Define and Simp only, and such that every $C \in N^{\mathrm{pre}}$ either does not contain any BG-sorted FG operator or $\mathrm{unabstr}(C)$ is 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 pure background term.

For all GBT clause sets N, thanks to the effect of the Define rule and Lemma 8.4, all offending occurrences of BG-sorted FG terms in N can stepwisely be eliminated until a clause set N^{pre} results. Notice that in every GBT clause set all BG terms are ground, hence pure. By definition, weak abstraction can introduce only abstraction variables then.

The $\mathrm{HSP}_{\mathrm{Base}}$ inferences do in general not preserve the shape of the clauses in N^{pre}. However, they do preserve a somewhat weaker property which is sufficient for our purposes. Let us say that a weakly abstracted clause C is *clean* if (i) every BG term in C is pure, (ii) every BG term in $\mathrm{unabstr}(C)$ is ground, and (iii) every occurrence of a BG-sorted FG operator f in $\mathrm{unabstr}(C)$ 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, assuming c is FG-sorted, $\mathsf{P}(\mathsf{f}(\mathsf{c}) + 1)$ is not clean, while $\mathsf{f}(x) \approx 1 + \alpha \vee P(x)$ is. A clause set is *clean* iff every clause in N is. From its definition it follows that N^{pre} is clean.

Lemma 8.6 *Let $C_1, \ldots, C_n$ be clean clauses. Assume a* $\mathrm{HSP}_{\mathrm{Base}}$ *inference with premises $C_1, \ldots, C_n$ and conclusion C. Then the following holds:*

(1) C is clean.

(2) Every BG term occurring in $\mathrm{unabstr}(C)$ *also occurs in some clause* $\mathrm{unabstr}(C_1), \ldots, \mathrm{unabstr}(C_n)$.

Proof. Let C' be the conclusion of the inference before weak abstraction, i.e., $C = \mathrm{abstr}(C')$. Regarding (1), property (i) of cleanness for C is trivial: all BG terms in the premises are pure, hence so are all BG terms in C'. Weak abstraction never introduces ordinary variables unless the given clause has ordinary variables. Thus all BG terms in C are pure as well.

The remaining properties for cleanness of C and property (2) can be seen from inspection of the $\mathrm{HSP}_{\mathrm{Base}}$ inference rules. We show only the case for superposition inferences, the other cases are similar. We distinguish three cases.

Case 1: a superposition inference into a subterm u of s_i of a literal $s \approx t$ of the right premise, where $s = \mathsf{f}(s_1, \ldots, s_i[u], \ldots, s_n)$ and f is a BG-sorted foreground operator. With items (i) and (iii) of cleanness it follows that u must be FG-sorted. In the conclusion of the inference u is replaced by a FG-sorted term $r\sigma$ where σ is the mgu of the inference. For any BG variable X occurring in C_1 or C_2, if $X\sigma \neq X$ then $X\sigma$ must be an (abstraction) variable occurring in C_1 or in C_2, or a domain element. This follows from weak abstraction of C_1 and C_2, as any composite BG term paired with X would have been abstracted out.

It is rather easy to see that C' does not require further abstraction, i.e., $C = C'$. By item (ii) of cleanness, every BG term in $\mathrm{unabstr}(C_1)$ and $\mathrm{unabstr}(C_2)$ is ground. In other words, every BG variable occurring in C_1 or C_2 is replaced by a ground term by unabstraction. This holds in particular for $X\sigma$ if $X\sigma$ is such a variable, as opposed to a domain element. It follows that every BG term in $\mathrm{unabstr}(C') = \mathrm{unabstr}(C)$ is ground and that every BG term in $\mathrm{unabstr}(C)$ also occurs in $\mathrm{unabstr}(C_1)$ or $\mathrm{unabstr}(C_2)$, i.e., property (2) of the lemma claim. Finally observe that property (iii) of cleanness holds trivially for C, which completes the proof of property (1).

Case 2: a superposition inference into any other FG-sorted subterm. This is proved in essentially the same way as in case 1.

Case 3: a superposition inference into the top position of the left side of $\mathsf{f}(s_1, \ldots s_n) \approx t$, where f is a BG-sorted foreground operator. From item (iii) of cleanness it follows this term is replaced by a ground term t', which does not need abstraction. The remaining argumentation is analogous to case 1 and is omitted. $\square$

Thanks to the additional assumptions, Simp also preserves cleanness:

Lemma 8.7 *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. We need to show properties (i)–(iii) of cleanness. That every BG term in D is pure follows from Def. 8.5-(iv). That every BG term in unabstr(D) is ground follows from Def. 8.5-(v) and cleanness of C. Finally, property (iii) follows from Def. 8.5-(iii). $\square$

With the above lemmas we can prove our main result:

Theorem 8.8 *The* HSP *calculus with a suitable* Simp *inference rule is refutationally complete for the ground BG-sorted term fragment. More precisely, if a set N of GBT clauses is $\mathcal{B}$-unsatisfiable then there is a refutation of N^{pre} without the* Define *rule.*

Proof. Let N be a GBT clause set. From Lemma 8.4 and Def. 8.5-(i) it follows that unabstr(N^{pre}) is also a GBT clause set. From the definitions of Define and Simp it follows that N^{pre} is weakly abstracted. Let $N_0 = N^{\text{pre}}$ and let $\mathcal{D} = (N_i)_{i \geq 0}$ be a fair derivation from N_0 without Define. Supposing $\mathcal{D}$ is not a refutation we need to show that N has a $\mathcal{B}$-model.

Let $N^\infty = \bigcup_{i \geq 0} N_i$ be the set of all derived clauses. We first show that unabstr(N^∞) contains only finitely many different BG terms and each of them is ground. Because N^{pre} is a GBT clause set this property holds trivially for unabstr(N^{pre}) $\subseteq$ unabstr(N^∞). Because Define is disabled in $\mathcal{D}$, only HSP_{Base} and (suitable) Simp inferences need to be analysed. Notice that N^{pre} is clean and both the HSP_{Base} and Simp inferences preserve cleanness, as per Lemmas 8.6-(1) and 8.7, respectively. The result then follows by induction using Lemma 8.6-(2) and Def. 8.5-(ii). More explicitly, HSP_{Base} inferences do not introduce new BG terms w. r. t. unabstr(N^{pre}), and the BG terms occurring in unabstr(N^{pre}) provide an upper bound w. r. t. the term ordering $\succ_{\text{fin}}$ 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 Def. 8.5-(ii).

Because every BG term occurring in unabstr(N^∞) is ground, every BG clause in unabstr(N^∞) 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 unabstr(N^∞), modulo equivalence. Because unabstraction is an equivalence transformation, there are only finitely many BG clauses in N^∞ as well, modulo equivalence.

The rest of the proof is similar to the proof of Thm. 7.8. Essentially, we need two changes, related to compactness and to sufficient completeness, respectively.

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 $I \in \mathcal{B}$. By Thm. 7.7, $(N_\infty)_I$ 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 I' of $(N_\infty)_I$. It follows that I' is also a Σ-model of $N_0(= N^{\mathrm{pre}})$.

Recall that N^{pre} is partitioned into two subsets, one containing only clauses without occurrences of BG-sorted FG symbols, and the other containing definitions only. The interpretation I' might still map BG-sorted FG terms other than those that have definitions in N^{pre} to non-domain elements. To fix that, we 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^{pre} is also mapped to an element of the carrier set of the proper sort. Notice that trivially I'' is still a Σ-model of N^{pre}. It follows that I'' is also a Σ-model of the given clause set N. With the same arguments as in the proof of Thm. 7.8 conclude that I'' is also a $\mathcal{B}$-model of N, which completes the proof. $\square$

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 Thm. 8.8 in that context as well.

Kruglov and Weidenbach [14] 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 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. The effect of basification is achieved in our calculus by the Define rule. Moreover, for GBT clause sets, by Thm. 8.8, Define needs to be applied as preprocessing only. Applying Define beyond that for non-GBT clause sets can still be useful. Ex. 8.2, for instance, cannot be solved with basification during preprocessing. The fragment of [14] is a further restriction of the GBT fragment. We expect we can get decidability results for that fragment with similar techniques.

9 Implementation and Experiments

We have implemented the HSP calculus and carried out experiments with the TPTP Library [22]. Our implementation, "Beagle", is intended as a testbed for rapidly trying out theoretical ideas for their practical viability.[1] 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 [23]).

Beagle's main loop is the well-known "Discount loop". It maintains two clause sets, *Old* and *New*, where *Old* is initially empty and new is initialized with the input clauses. On each round, a "selected" clause is removed from new and simplified by the clauses from *Old* and *New*. The simplified selected clause then is put into old and all inferences between it and the clauses in *Old* are carried out. The resulting clauses go into *New* again, this way closing the loop. By default, a split rule is enabled that breaks clauses into variable-disjoint subclauses and branches out correspondingly. Dependency-directed backtracking is used to avoid exploring irrelevant cases.

Fairness is achieved through a combination of clause weights and their derivation-age.[2] The BG reasoner is a quantifier elimination procedure for linear integer arithmetic (LIA) based on Cooper's algorithm; it is called with all current BG clauses as inputs whenever a new BG clause has been derived. More precisely, with every BG clause we cache a quantifier-free version, which is the one used for that.

Implemented simplification techniques include standard ones, like 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. Specific simplification rules related to BG reasoning come in two kinds: *cautious* simplification, which removes clause literals of the form $\zeta \not\approx d$ by unabstraction if d is a domain element. It also replaces ground terms of the form $d_1\ op\ d_2$, where $op \in \{+, -, \cdot\}$, by their evaluated result. Ground parameter-free literals are evaluated simi-

[1]`http://users.cecs.anu.edu.au/~baumgart/systems/beagle/`

[2]This is controlled by a parameter "weight-age-ratio", a non-negative number saying how many lightest clauses are selected before an oldest clause is selected. Clause weights are computed in such a way that selection based on weights only would be a fair strategy. In our experiments we used a weight-age-ratio of eight.

larly. *Aggressive* simplification goes beyond that by exploiting associativity and commutativity of operators. Unlike cautious simplification, it does not always preserve sufficient completeness. For example, in the clause set $N = \{\mathsf{P}(1 + (2 + \mathsf{c})),\ \neg\mathsf{P}(1 + (X + \mathsf{c}))\}$ the first clause is (aggressively) simplified, giving $N' = \{\mathsf{P}(3 + \mathsf{c}),\ \neg\mathsf{P}(1 + (X + \mathsf{c}))\}$. Notice that 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.

The user can explicitly provide *lemma clauses.* These are clauses that are meant to be valid background theory formulas that can help to find a refutation quicker (or to find it at all). All BG variables in lemma clauses are always ordinary variables. An example is the clause $\neg(x < x)$, as explained in Sect. 1. In our experiments we used the following lemmas (all variables integer-sorted).

$$\begin{aligned} -(-x) &\approx x & x * 1 &\approx x \\ (x + (-y)) + y &\approx x & 1 * x &\approx x \\ x + 0 &\approx x & 0 * x &\approx 0 \\ 0 + x &\approx x & x * 0 &\approx 0 \\ x + (-x) &\approx 0 & x &< x + 1 \\ (-x) + x &\approx 0 & &\neg(x < x) \\ & & x < y \lor y &< x \lor x \approx y \end{aligned}$$

Lemma clauses can be treated in two ways. First, a lemma clause can be added to the *Old* clause set straight away and without undergoing abstraction. The rationale is to avoid inferences among lemma clauses, for a more goal-oriented strategy, and to emphasize the use of lemma clauses as demodulators. A good example is the unit clause $(x + (-y)) + y \approx x$, which can be used for demodulation in this unabstracted form. However, notice that adding a set of lemma clauses that is not saturated wrt. $\mathrm{HSP}_{\mathrm{Base}}$ and $\mathcal{R}^{\mathcal{H}}$ will in general lead to refutational incompleteness. This is addressed by the second way, by treating a lemma clause as an input clause all whose (BG) variables are of the ordinary kind.

There are various other options to control the prover, for instance to turn off the Define rule, and to specify whether the BG variables in the input clauses are taken as abstraction variables (default) or as ordinary variables. (Recall from above that BG variables in lemma clauses are always ordinary variables.) By default, integer-sorted FG-constants are converted into parameters. This way integer-sorted FG constants never cause problems related to sufficient completeness.

Beagle is implemented in Scala. 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. In all our experiments we declared the input variables as abstraction variables and we enabled the split rule, which gave better results. The CPU time limit was 300 seconds. Our results are summarized in Table 9.1 below. None of the HWV-problems in the TPTP library was solvable within the time limit, though, and hence we omitted these.

Table 9.1 is explained as follows. The row FA ("Full Abstraction") contains the results when Beagle is set to the previous calculus [5]. That calculus is obtained by using abstraction variables, by full abstraction, without the Define rule, without lemma clauses and without the BG specific simplification techniques mentioned above. As explained earlier, the Define rule can be used in conjunction with that calculus as well, however in a modified form that uses full abstraction for its conclusions. The second row contains the results for that setting.

All subsequent rows, with WA ("Weak Abstraction"), refer to various settings using the new calculus. The setting WA alone differs from FA only by using weak abstraction, but no other improvements are in effect. The WA setting solves strictly more problems than FA and often quicker. That is, weak abstraction always pays always off. For instance, the ARI-problems solvable with weak abstraction are ARI602=1.p and ARI608=1.p. The problem ARI602=1.p asks to prove that from $f(X) > X$ follows there is a Y such that $4 < Y < f(Y)$. Thanks to not having to abstract out the term $f(Y)$ it is easy with WA, but impossible with FA. The same effect shows up with ARI608=1.p.

The next four rows show the results with cautions simplification in place ("+BGSC"). We have added lemmas ("+Lemmas") and the Define rule ("+Define") separately and in combination, as shown, to analyze their respective benefits. As the results show, the Define rule alone has more significant impact than adding lemmas. Their combination, though, is always preferable. An outlier is DAT048=1.p which is no longer solvable with cautious simplification alone or in combination with lemmas due to a timeout.

The rows with "+Lemmas" use lemmas in the first way described above. In our experiments with the "Theorem" TPTP problems the first way was more successfull. Nevertheless, the second way is interesting as well, in particular in combination with cautious simplification and the Define rule, as we get a calculus that is refutationally complete for the GBT fragment (cf. Thm. 8.8). The row “WA +BGSC +Define +Lemmas^c” contains the corresponding results. For that, we removed the lemma clauses $(x+(-y))+y \approx x$ and $x < y \vee y < x \vee x \approx y$ in order to not cause non-termination on the lemma clauses alone. This enables Beagle to correctly report “CounterSatisfiable” for GBT input clause sets with a finite saturation that do not contain the empty clause.[3]

The last two rows in Table 9.1 contain the result with aggressive simplification in place (“+BGSA”). We have also tried adding lemmas in conjunction with the Define rule, but that did not change much. Indeed, the aggressive simplification techniques overlap largely with the lemmas, and so this is not a surprise. As above, switching Define on is rather helpful in combination with aggressive simplification as well. This is the setting that solved most problems overall.

It is interesting to compare the results for “WA +BGSA +Define” with those for “WA +BGSC +Define +Lemmas”, the best settings with aggressive and cautious simplification, respectively. The latter did not fall too much behind the former and has the advantage that it is more flexible, as the lemmas can be specified by the user as desired. The runtimes are incomparable, but, by and large aggressive simplification seems to perform better.

We experimented separately with the GEG-problems. These contain an “unusal” number of different integer constants. Four out of the five problems are solvable with removing load from the background reasoner by disabling Define, aggressive simplification and including a transitivity clause for the less-or-equal relation as a lemma.

The TSTP web page contains individual solutions to TPTP problems for various provers. About 12 provers can sensibly be applied to problems over linear integer arithmetic. The more difficult solvable problems can typically be solved by four or less systems, with CVC3 [6], Princess [21], SPASS+T [20] and Z3 [16] the most reliable ones. There are a number of problems that only Princess and Beagle solve.

[3]We also ran Beagle on the countersatisfiable LIA-problems from the TPTP library. All but three problems are easily solved that way, and the remaining three are too hard, for every suitable flag setting.

	ARI	**DAT**	**GEG**	**PUZ**	**NUM**	**SEV**	**SWV**	**SWW**	**SYO**
	223	54	5	1	27	6	2	3	3
FA	194	23	2	0	20	2	0	1	0
FA +Define	211	45	2	1	26	2	1	2	2
WA	194	28	2	0	20	2	2	3	0
WA +BGSC	194	27	2	0	20	2	2	3	0
WA +BGSC +Lemmas	194	27	2	0	20	2	2	3	0
WA +BGSC +Define	211	50	1	1	26	2	2	3	1
WA +BGSC +Define +Lemmas	211	53	1	1	22	2	2	3	1
WA +BGSC +Define +Lemmas^c	211	50	1	0	23	2	2	3	1
WA +BGSA	194	32	2	0	20	2	2	3	0
WA +BGSA +Define	211	54	2	1	26	2	2	3	2

Table 9.1: Experimental results on the "Theorem" problems over integer integer arithmetic in theTPTP library. *Rows:* Prover settings, where *FA* is "Full Abstraction, *WA* is "Weak Abstraction", *BGSC* is "Cautious BG simplification", *BGSA* is "Aggressive BG simplification", *+Define* means the Define rule is in effect, *+Lemmas* means the lemma clauses are used, in the first way described in the text, and *+Lemmas*^{c} ("c"omplete) means a subset of the lemma clauses are used in the second way. *Columns:* TPTP problem categories; each category is listed with the number of problems with status "Theorem" in it. *Entries:* number of problems with status "Theorem" solved.

10 Conclusions

The main theoretical contribution of this paper is an improved variant of the hierarchic superposition calculus. The improvements over its predecessor [5] are grounded in a different form of "abstracted" clauses, the clauses the calculus works with internally. Because of that, a modified completeness proof is required. We have 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., "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. They also confirm advantages of the new calculus over the old, the former solves strictly more more problems than the latter (and is never slower on the common set). 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 [14], as discussed in Sect. 8. It is also loosely related to results in SMT-based theorem proving. For instance, the method in [12] 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. 8.8 (when it applies). However, according to Thm. 8.8 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.

Bibliography

[1] E. Althaus, E. Kruglov, and C. Weidenbach. Superposition modulo linear arithmetic SUP(LA). In *FroCos*, 2009, *LNCS 5749*, pp. 84–99. Springer.

[2] A. Armando, M. P. Bonacina, S. Ranise, and S. Schulz. New results on rewrite-based satisfiability procedures. *ACM Trans. Comput. Log.*, 10(1), 2009.

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

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

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

[6] C. Barrett and C. Tinelli. CVC3. In W. Damm and H. Hermanns, eds., *Proceedings of the* 19^{th} *International Conference on Computer Aided Verification (CAV '07)*, 2007, *LNCS 4590*, pp. 298–302. Springer-Verlag. Berlin, Germany.

[7] P. Baumgartner, A. Fuchs, and C. Tinelli. ME(LIA) – Model Evolution With Linear Integer Arithmetic Constraints. In *15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'08)*, 2008, *LNAI 5330*, pp. 258–273. Springer.

[8] P. Baumgartner and C. Tinelli. Model evolution with equality modulo built-in theories. In *CADE-23 – The 23nd International Conference on Automated Deduction*, 2011, *LNAI 6803*, pp. 85–100. Springer.

[9] M. P. Bonacina, C. Lynch, and L. M. de Moura. On deciding satisfiability by theorem proving with speculative inferences. *J. Autom. Reasoning*, 47(2):161–189, 2011.

[10] H. Ganzinger and K. Korovin. Theory instantiation. In *13th Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*, 2006, *LNCS 4246*, pp. 497–511. Springer.

[11] Y. Ge, C. Barrett, and C. Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In F. Pfenning, ed., *21st International Conference on Automated Deduction (CADE-21), Bremen, Germany*, 2007, *LNCS 4603*. Springer.

[12] Y. Ge and L. M. de Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In *CAV*, 2009, *LNCS 5643*, pp. 306–320. Springer.

[13] K. Korovin and A. Voronkov. Integrating linear arithmetic into superposition calculus. In *Computer Science Logic (CSL'07)*, 2007, *LNCS 4646*, pp. 223–237. Springer.

[14] E. Kruglov and C. Weidenbach. Superposition decides the first-order logic fragment over ground theories. *Mathematics in Computer Science*, pp. 1–30, 2012.

[15] L. M. de Moura and N. Bjørner. Engineering DPLL(T) + saturation. In *Automated Reasoning, 4th International Joint Conference, IJCAR*, 2008, *LNCS 5195*, pp. 475–490. Springer.

[16] L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and J. Rehof, eds., *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008*, Budapest, Hungary, 2008, *LNCS 4963*, pp. 337–340. Springer.

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

[18] R. Nieuwenhuis, A. Oliveras, and C. 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, 2006.

[19] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In *Handbook of Automated Reasoning*, pp. 371–443. Elsevier and MIT Press, 2001.

[20] V. Prevosto and U. Waldmann. SPASS+T. In G. Sutcliffe, R. Schmidt, and S. Schulz, eds., *ESCoR: FLoC'06 Workshop on Empirically Successful Computerized Reasoning*, Seattle, WA, USA, 2006, *CEUR Workshop Proceedings*, vol. 192, pp. 18–33.

[21] P. 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)*, 2008, *LNAI 5330*, pp. 274–289. Springer.

[22] 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.

[23] G. Sutcliffe, S. Schulz, K. Claessen, and P. Baumgartner. The TPTP typed first-order form with arithmetic. In *18th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-18)*, 2012, *LNAI 7180*. Springer.

[24] C. Walther. Many-sorted unification. *Journal of the ACM*, 35(1):1–17, 1988.

MPI-I-2012-RG1-002	A. Fietzke, E. Kruglov, C. Weidenbach	Automatic generation of inductive invariants by SUP(LA)
MPI-I-2012-RG1-001	M. Suda, C. Weidenbach	Labelled superposition for PLTL
MPI-I-2012-5-004	F. Alvanaki, S. Michel, A. Stupar	Building and maintaining halls of fame over a database
MPI-I-2012-5-003	K. Berberich, S. Bedathur	Computing n-gram statistics in MapReduce
MPI-I-2012-5-002	M. Dylla, I. Miliaraki, M. Theobald	Top-k query processing in probabilistic databases with non-materialized views
MPI-I-2012-5-001	P. Miettinen, J. Vreeken	MDL4BMF: Minimum Description Length for Boolean Matrix Factorization
MPI-I-2012-4-001	J. Kerber, M. Bokeloh, M. Wand, H. Seidel	Symmetry detection in large scale city scans
MPI-I-2011-RG1-002	T. Lu, S. Merz, C. Weidenbach	Towards verification of the pastry protocol using TLA+
MPI-I-2011-5-002	B. Taneva, M. Kacimi, G. Weikum	Finding images of rare and ambiguous entities
MPI-I-2011-5-001	A. Anand, S. Bedathur, K. Berberich, R. Schenkel	Temporal index sharding for space-time efficiency in archive search
MPI-I-2011-4-005	A. Berner, O. Burghard, M. Wand, N.J. Mitra, R. Klein, H. Seidel	A morphable part model for shape manipulation
MPI-I-2011-4-002	K.I. Kim, Y. Kwon, J.H. Kim, C. Theobalt	Efficient learning-based image enhancement: application to compression artifact removal and super-resolution
MPI-I-2011-4-001	M. Granados, J. Tompkin, K. In Kim, O. Grau, J. Kautz, C. Theobalt	How not to be seen – inpainting dynamic objects in crowded scenes
MPI-I-2010-RG1-001	M. Suda, C. Weidenbach, P. Wischnewski	On the saturation of YAGO
MPI-I-2010-5-008	S. Elbassuoni, M. Ramanath, G. Weikum	Query relaxation for entity-relationship search
MPI-I-2010-5-007	J. Hoffart, F.M. Suchanek, K. Berberich, G. Weikum	YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia
MPI-I-2010-5-006	A. Broschart, R. Schenkel	Real-time text queries with tunable term pair indexes
MPI-I-2010-5-005	S. Seufert, S. Bedathur, J. Mestre, G. Weikum	Bonsai: Growing Interesting Small Trees
MPI-I-2010-5-004	N. Preda, F. Suchanek, W. Yuan, G. Weikum	Query evaluation with asymmetric web services
MPI-I-2010-5-003	A. Anand, S. Bedathur, K. Berberich, R. Schenkel	Efficient temporal keyword queries over versioned text
MPI-I-2010-5-002	M. Theobald, M. Sozio, F. Suchanek, N. Nakashole	URDF: Efficient Reasoning in Uncertain RDF Knowledge Bases with Soft and Hard Rules
MPI-I-2010-5-001	K. Berberich, S. Bedathur, O. Alonso, G. Weikum	A language modeling approach for temporal information needs
MPI-I-2010-1-001	C. Huang, T. Kavitha	Maximum cfardinality popular matchings in strict two-sided preference lists
MPI-I-2009-RG1-005	M. Horbach, C. Weidenbach	Superposition for fixed domains
MPI-I-2009-RG1-004	M. Horbach, C. Weidenbach	Decidability results for saturation-based model building
MPI-I-2009-RG1-002	P. Wischnewski, C. Weidenbach	Contextual rewriting

MPI-I-2009-RG1-001	M. Horbach, C. Weidenbach	Deciding the inductive validity of $\forall\exists^*$ queries
MPI-I-2009-5-007	G. Kasneci, G. Weikum, S. Elbassuoni	MING: Mining Informative Entity-Relationship Subgraphs
MPI-I-2009-5-006	S. Bedathur, K. Berberich, J. Dittrich, N. Mamoulis, G. Weikum	Scalable phrase mining for ad-hoc text analytics
MPI-I-2009-5-005	G. de Melo, G. Weikum	Towards a Universal Wordnet by learning from combined evidenc
MPI-I-2009-5-004	N. Preda, F.M. Suchanek, G. Kasneci, T. Neumann, G. Weikum	Coupling knowledge bases and web services for active knowledge
MPI-I-2009-5-003	T. Neumann, G. Weikum	The RDF-3X engine for scalable management of RDF data
MPI-I-2009-5-003	T. Neumann, G. Weikum	The RDF-3X engine for scalable management of RDF data
MPI-I-2009-5-002	M. Ramanath, K.S. Kumar, G. Ifrim	Generating concise and readable summaries of XML documents
MPI-I-2009-4-006	C. Stoll	Optical reconstruction of detailed animatable human body models
MPI-I-2009-4-005	A. Berner, M. Bokeloh, M. Wand, A. Schilling, H. Seidel	Generalized intrinsic symmetry detection
MPI-I-2009-4-004	V. Havran, J. Zajac, J. Drahokoupil, H. Seidel	MPI Informatics building model as data for your research
MPI-I-2009-4-003	M. Fuchs, T. Chen, O. Wang, R. Raskar, H.P.A. Lensch, H. Seidel	A shaped temporal filter camera
MPI-I-2009-4-002	A. Tevs, M. Wand, I. Ihrke, H. Seidel	A Bayesian approach to manifold topology reconstruction
MPI-I-2009-4-001	M.B. Hullin, B. Ajdin, J. Hanika, H. Seidel, J. Kautz, H.P.A. Lensch	Acquisition and analysis of bispectral bidirectional reflectance distribution functions
MPI-I-2008-RG1-001	A. Fietzke, C. Weidenbach	Labelled splitting
MPI-I-2008-5-004	F. Suchanek, M. Sozio, G. Weikum	SOFI: a self-organizing framework for information extraction
MPI-I-2008-5-003	G. de Melo, F.M. Suchanek, A. Pease	Integrating Yago into the suggested upper merged ontology
MPI-I-2008-5-002	T. Neumann, G. Moerkotte	Single phase construction of optimal DAG-structured QEPs
MPI-I-2008-5-001	G. Kasneci, M. Ramanath, M. Sozio, F.M. Suchanek, G. Weikum	STAR: Steiner tree approximation in relationship-graphs
MPI-I-2008-4-003	T. Schultz, H. Theisel, H. Seidel	Crease surfaces: from theory to extraction and application to diffusion tensor MRI
MPI-I-2008-4-002	D. Wang, A. Belyaev, W. Saleem, H. Seidel	Estimating complexity of 3D shapes using view similarity
MPI-I-2008-1-001	D. Ajwani, I. Malinger, U. Meyer, S. Toledo	Characterizing the performance of Flash memory storage devices and its impact on algorithm design
MPI-I-2007-RG1-002	T. Hillenbrand, C. Weidenbach	Superposition for finite domains
MPI-I-2007-5-003	F.M. Suchanek, G. Kasneci, G. Weikum	Yago: a large ontology from Wikipedia and WordNet
MPI-I-2007-5-002	K. Berberich, S. Bedathur, T. Neumann, G. Weikum	A time machine for text search
MPI-I-2007-5-001	G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum	NAGA: searching and ranking knowledge
MPI-I-2007-4-008	J. Gall, T. Brox, B. Rosenhahn, H. Seidel	Global stochastic optimization for robust and accurate human motion capture
MPI-I-2007-4-007	R. Herzog, V. Havran, K. Myszkowski, H. Seidel	Global illumination using photon ray splatting
MPI-I-2007-4-006	C. Dyken, G. Ziegler, C. Theobalt, H. Seidel	GPU marching cubes on shader model 3.0 and 4.0