# Superposition Modulo Theory

Dissertation

zur Erlangung des Grades
des Doktors der Naturwissenschaften (Dr. rer. nat.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

von

Evgeny Kruglov

Saarbrücken, Oktober 2013

|  |  |
|---|---|
| Tag des Kolloquiums: | 31. Oktober 2013 |
| Dekan: | Prof. Dr. Mark Groves |
| Vorsitzender des Prüfungsausschusses: | Prof. Dr. Gert Smolka |
| Berichterstatter: | Prof. Dr. Ernst Althaus |
|  | Prof. Dr. Martin Fränzle |
|  | Prof. Dr. Christoph Weidenbach |
| Akademischer Mitarbeiter: | PD. Dr. Thomas Sturm |

To all my teachers ...

# Acknowledgements

*If I have seen farther, it is by standing on the shoulders of giants.*

Isaac Newton

I want to thank everybody who has invested his time and endeavor to deliver knowledge and educate me: school teachers, university lecturers, colleagues. My scientific achievements are to the greatest extent the result of your efforts and influence. Throughout this dissertation I always use the pronoun "we" instead of "I": "we have proved", "we have shown", etc., — to indicate that each of us (you and me) have contributed (implicitly or explicitly) to produce the results presented in the dissertation.

I thank Prof. Dr. Ernst Althaus for his expertise concerning Linear Algebra and Linear Programming and cooperative work on investigating the Hierarchic Superposition Modulo Linear Arithmetic calculus SUP(LA).

I thank Prof. Dr. Martin Fränzle and a very friendly research team of exceptionally talented and intelligent people, namely: Andreas Eggers, Stefan Kupferschmid, Karsten Scheibler, and Tino Teige, for cooperative work on investigating the Hierarchic Superposition Modulo Non-Linear Arithmetic calculus SUP(NLA) and implementing the calculus in SPASS(iSAT).

I also want to thank my colleagues in the Automation of Logic group at the Max Planck Institute for Informatics for responsive working environment and very friendly social atmosphere. Special thanks to Arnaud Fietzke for extensive discussions of various aspects of the Hierarchic Superposition Calculus SUP(T), to Marek Kosta for partly proofreading the chapter on hierarchic refutational theorem proving, to Willem Hagemann for translating the abstract to German.

To the highest degree I am grateful to my doctoral advisor Prof. Dr. Christoph Weidenbach, who has offered me such an interesting, exciting, and prolific research topic as Superposition Modulo Theory. I thank him for his persistent guidance, highly valuable advices, rigorous encouragement whenever I lacked any motivation to continue research, for his patience and support when I was long badly distressed; none of these can be overestimated.

This work has been partly supported by the German Transregional Collaborative Research Center SFB/TR 14 AVACS.

## Abstract

This thesis is about the Hierarchic Superposition calculus SUP(T) and its application to reasoning in hierarchic combinations FOL(T) of the free first-order logic FOL with a background theory T where the hierarchic calculus is refutationally complete or serves as a decision procedure. Particular hierarchic combinations covered in the thesis are the combinations of FOL and linear and non-linear arithmetic, LA and NLA resp.

Recent progress in automated reasoning has greatly encouraged numerous applications in soft- and hardware verification and the analysis of complex systems. The applications typically require to determine the validity/unsatisfiability of quantified formulae over the combination of the free first-order logic with some background theories. The hierarchic superposition leverages both (i) the reasoning in FOL equational clauses with universally quantified variables, like the standard superposition does, and (ii) powerful reasoning techniques in such theories as, e.g., arithmetic, which are usually not (finitely) axiomatizable by FOL formulae, like modern SMT solvers do. The thesis significantly extends previous results on SUP(T), particularly: we introduce new substantially more effective sufficient completeness and hierarchic redundancy criteria turning SUP(T) to a complete or a decision procedure for various FOL(T) fragments; instantiate and refine SUP(T) to effectively support particular combinations of FOL with the LA and NLA theories enabling a fully automatic mechanism of reasoning about systems formalized in FOL(LA) or FOL(NLA).

## Zusammenfassung

Diese Arbeit befasst sich mit dem hierarchischen Superpositionskalkül SUP(T) und seiner Anwendung auf hierarchischen Kombinationen FOL(T) der freien Logik erste Stufe FOL und einer Hintergrundtheorie T, deren hierarchischer Kalkül widerlegungsvollständig ist oder als Entscheidungsverfahren dient. Die behandelten hierarchischen Kombinationen sind im Besonderen die Kombinationen von FOL und linearer und nichtlinearer Arithmetik, LA bzw. NLA.

Die jüngsten Fortschritte in dem Bereich des automatisierten Beweisens haben zahlreiche Anwendungen in der Soft- und Hardwareverifikation und der Analyse komplexer Systeme hervorgebracht. Die Anwendungen erfordern typischerweise die Gültigkeit/Unerfüllbarkeit quantifizierter Formeln über Kombinationen der freien Logik erste Stufe mit Hintergrundtheorien zu bestimmen. Die hierarchische Superposition verbindet beides: (i) das Beweisen über FOL-Klauseln mit Gleichheit und allquantifizierten Variablen, wie in der Standardsuperposition, und (ii) mächtige Beweistechniken in Theorien, die üblicherweise nicht (endlich) axiomatisierbar durch FOL-Formeln sind (z. B. Arithmetik), wie in modernen SMT-Solvern. Diese Arbeit erweitert frühere Ergebnisse über SUP(T) signifikant, im Besonderen führen wir substantiell effektiverer Kriterien zur Bestimmung der hinreichenden Vollständigkeit und der hierarchischen Redundanz ein. Mit diesen Kriterien wird SUP(T) widerlegungsvollständig beziehungsweise ein Entscheidungsverfahren für verschiedene FOL(T)-Fragmente. Weiterhin instantiieren und verfeinern wir SUP(T), um effektiv die Kombinationen von FOL mit der LA- und der NLA-Theorie zu unterstützen, und erhalten eine vollautomatische Beweisprozedur auf Systemen, die in FOL(LA) oder FOL(NLA) formalisiert werden können.

# Contents

CHAPTER 1

# Introduction

*Small reasons lead to big consequences.*

*Look in the root!*

Kozma Prutkov

## Motivation

Recent progress in automated reasoning has greatly encouraged numerous applications in soft- and hardware verification and the analysis of complex systems. The applications typically require to determine the validity/unsatisfiability of formulae over the combination of the free first-order logic FOL with some background theories T. Often such formulae include quantifiers. Nowadays, there are efficient theorem provers for the full free first-order logic with equality and quantifiers, as well as techniques and tools realizing reasoning about ground formulae, or about formulae that can be effectively and efficiently reduced to a ground fragment, in various theories, such as linear and non-linear arithmetic, data structures, set theory, etc., and combinations thereof. Typically, reasoning in combinations of free first-order logic and background theories is weak: existing methods that are aimed at handling problems in FOL modulo T are usually deficient in universality and/or completeness, due to (i) their restriction to very specific axiomatizations of the background theory T or of the free enrichment of T, and/or (ii) partial prevention or poor treatment of quantifiers, resulting therefore in incomplete or very specialized procedures.

## Our Contribution

This dissertation evolves the methodology of universal and complete reasoning in general combinations FOL(T) of the free first-order logic FOL with a background theory T taking as a computational basis the Hierarchic Superposition approach [BGW92, BGW94] of Bachmair, Ganzinger, and Waldmann.

The Hierarchic Superposition calculus SUP(T) is aimed at reasoning in the full clausal FOL(T) fragment with equality. It combines the standard "flat" superposition SUP [HR91, BG94, NR01, Wei01] and a background theory solver in a modular fashion: while the hierarchic calculus SUP(T) maneuvers the overall proof search deriving new clauses similarly to how the flat SUP calculus does, it delivers pure theory formulae arising as side conditions to the background solver, invoked thus like a subroutine in a program. The hierarchic SUP(T) calculus greatly benefits from (i) a sufficient completeness criterion, according to which the hierarchic calculus is complete for all clause sets that possess a sufficient completeness property; and (ii) a hierarchic redundancy criterion which enables deletion and simplification of clauses deemed to be redundant in the hierarchic FOL(T) setting. Bachmair, Ganzinger, and Waldmann have proposed in [BG94] first versions of a sufficient completeness criterion and a hierarchic redundancy criterion, the former being rather weak, and the latter a straightforward lifting of the standard redundancy criterion to the hierarchic FOL(T) level, both loosely treating the background theory T.

Our main contributions presented in this dissertation comprise the following:

1. We have *significantly strengthened the sufficient completeness criterion* originally suggested by Bachmair, Ganzinger, and Waldmann. Our reformulation of the sufficient completeness criterion covers much larger class of clause sets for which SUP(T) is complete.

   The original sufficient completeness criterion enjoins all ground FOL(T)-terms of a background T-sort to be reducible to some ground T-terms *in all models* of the subset of all ground instances of a given clause set $N$, where all T-sort variables in clauses in $N$ are instantiated only with ground T-terms (the so-called set of all *simple ground instances*). Basically, we have relaxed the above requirement by (i) considering not all terms specified above, but only those which have an occurrence of a free function symbol ranging into a T-sort only at the top position ((*smooth*) *extension* terms, as we call them), and (ii) evaluating the terms not in all models of the set of all simple ground instances of $N$, but only in those which are monomorphic extensions of models of the background theory T (*weak algebras*, as we call them).

2. We have presented a *new sophisticated hierarchic redundancy criterion* which takes into account redundancy with respect to a background theory T, in contrast to the hierarchic redundancy criterion suggested by Bachmair, Ganzinger, and Waldmann. The new criterion enables substantially more effective redundancy elimination and simplification techniques.

3. In addition to elaborating a sufficient completeness criterion, we have also developed a noval *Local Sufficient Completeness Criterion*, according to which

for achieving completeness it suffices to sufficiently define only those extension terms that occur in a given clause set.

The local version of the sufficient completeness criterion suggests a transformation method of turning a given set $N$ of FOL(T) clauses, which is initially not sufficiently complete, to a modified equisatisfiable set $N'$, possessing the local sufficient completeness property, by adding extra clauses to $N$, that enforce reducibility of the required terms to base terms, which are otherwise not sufficiently defined by $N$ itself. In this dissertation, we have exploited such a technique to show that SUP(T) can decide or can be turned to a complete procedure for particular fragments of FOL(T).

4. We have developed two basic *hierarchic reduction rules*: Hierarchic Tautology Deletion and Hierarchic Subsumption Deletion, which are compliant with the hierarchic redundancy criterion. The hierarchic reduction rules, in contrast to their "flat" versions, take into account the hierarchic nature of the FOL(T) combination and can therefore detect redundancy with respect to a background theory T.

   These two rules may serve as a basis for defining hierarchic versions of other reduction rules traditionally used in modern superposition-based theorem provers, such as: Condensation, Unit Conflict, Matching Replacement Resolution, (Non-)Unit Rewriting[1], etc.

5. We have shown that the hierarchic superposition calculus SUP(T) is a *decision procedure* for the ground FOL(T) fragment. Thanks to the modular design of SUP(T), the decidability result can be easily extended beyond the ground case: we have defined the *Bernays-Schönfinkel Horn class with equality and ground base sort terms, BSHE(GBST)*, and proved SUP(T) to decide the class.

   Worthwhile to mention, the decidability results have been obtained leveraging the above *local sufficient completeness* notion: as sets of ground FOL(T) clauses are neither sufficiently complete, nor locally sufficiently complete, in general, completeness is achieved by *Basification*, a preprocessing procedure which expands an input clause set $N$ with extra clauses equating ground smooth extension terms, that appear in $N$, to unique fresh constants (parameters), yielding this way a clause set $N'$ which is locally sufficiently complete.

   The ground case decidability is not much surprising. Nevertheless, SUP(T) with Basification incorporated constitute a decision procedure for many non-trivial theory combinations, important in verification and reasoning in which can be reduced to the ground fragment, e.g. in Local Theory Extensions [SS05, IJSS08, ISS10]. The BSHE(GBST) class is strong enough to describe ontologies with arithmetical facts, and SUP(T) can hence be used for reasoning about and querying such kind of ontologies.

[1]Although not reported in this dissertation, we have implemented hierarchic versions of the listed reduction rules in a system combination SPASS(Z3) of the first-order theorem prover SPASS and SMT-solver Z3.

6. We have instantiated and refined the hierarchic superposition calculus SUP(T) for the *hierarchic combination FOL(LA)* of FOL and the theory of *linear arithmetic LA*.

   In particular, we have refined the hierarchic reduction rules and developed new effective algorithms for checking LA formulae arising as side conditions of inference and reduction rules of the obtained *Superposition Modulo Linear Arithmetic SUP(LA)* calculus.

   We have implemented the SUP(LA) calculus in SPASS(LA), a system combination of the automated first-order theorem prover SPASS [WDF$^+$09] and a background LA-solver. In place of an LA-solver we have used Simplex-based LP-solver QSopt[1], SMT-solver Z3 [dMB08c], QE-based computer-logic system Redlog [DS97], and Büchi-automata-based toolkit LIRA [BDEK07]. We have shown that SUP(LA) can be successfully used for practical applications; in particular, as case studies we have considered transition systems.

7. We have also investigated the hierarchic SUP(NLA) calculus for reasoning about the *hierarchic combination FOL(NLA)* of FOL and the theory of *non-linear arithmetic NLA* over the reals including transcendental functions, such as exponentiation and trigonometric functions.

   In contrast to the SUP(LA) approach, where T-related problems are all decidable, here this is not the case due to undecidability of NLA involving transcendental functions, yielding that the resulting calculus SUP(NLA) is not complete, in general, even if a given clause set is (locally) sufficiently complete. Nevertheless, we have strengthened the (local) sufficient completeness criterion to account for the NLA reasoning issues, gaining this way a complete (and sound) calculus for an attractive fragment of FOL(NLA).

   As an ingredient towards obtaining a practically working approach, we have developed a series of simplification and approximation techniques for arithmetic formulae arising as SUP(NLA) saturates a FOL(NLA) clause set. We have shown that a particular class of FOL(NLA) formulae is strong enough to formalize safety properties of hybrid non-linear systems.

   We have implemented the SUP(NLA) calculus in SPASS(iSAT), a system combination of SPASS and an SMT solver for non-linear arithmetic iSAT [2]. Applied to various scenarios of traffic collision avoidance protocols, we have shown by experiments that SPASS(iSAT) can fully automatically proof and disproof safety properties of such protocols using the very same formalization.

[1]See `http://www2.isye.gatech.edu/~wcook/qsopt/`.
[2]See `http://isat.gforge.avacs.org/`

## Related Work

There are now several calculi combining full first-order logic FOL with a background theory T, and systems (implementations) supporting reasoning in FOL(T) available. The main difference between all of them, on the one hand, and our approach, on the other hand, is that (i) they are restricted to very specific axiomatizations of the background theory T or of the free enrichment of T, and/or (ii) they partially prevent or poorly treat quantifiers, resulting therefore in incomplete or very specialized procedures. Thus, in contrast to SUP(T), these techniques are usually deficient in universality and/or completeness.

Below we provide a short description of approaches comparable to SUP(T).

**DPLL($\Gamma$ + T) (with Speculative Inferences).** DPLL($\Gamma$ + T) [BLdM09, BLdM11] of Bonacina, Lynch, and de Moura is a deep integration of (i) a generic superposition-based inference system $\Gamma$ and (ii) a DPLL(T)-based SMT-solver. DPLL($\Gamma$ + T) is aimed at reasoning in a combination of background theories $\mathrm{T} = \bigcup_i \mathrm{T}_i$ and an axiomatized theory $R$. The general format of an input problem is $R \cup P$, where $R$ is a set of non-ground clauses (axioms) without occurrences of T-symbols, and $P$ a set of ground clauses over T and $R$ (typically, $P$ is obtained by skolemizing a negated universal conjecture whose validity in $R$ modulo T needs to be (dis)proved).

DPLL($\Gamma$ + T) incorporates an SMT technology called DPLL(T) [NOT06, Seb07, BSST09]. DPLL(T) is aimed at determining the satisfiability of ground formulae a combination of background theories $\mathrm{T} = \bigcup_i \mathrm{T}_i$, and integrates a DPLL-based SAT-solver [DP60, DLL62, ZS00, MMZ+01] and satellite solvers for the theories $\mathrm{T}_1$, $\mathrm{T}_2$, ... Operationally, while the SAT-solver tries to build a candidate (propositional) model of a given set of ground T-clauses by maintaining and manipulating a set of propositional variables associated with ground theory literals, the $\mathrm{T}_i$-solvers check whether the propositional model is consistent with respect to each theory $\mathrm{T}_i$. The interaction between the $\mathrm{T}_i$-solvers is realized by different equality sharing methods; the model-based theory combination [dMB08b] method is one of the most efficient such methods existing (and underlies DPLL(T) in DPLL($\Gamma$ + T)).

The inference system $\Gamma$ serves in DPLL($\Gamma$ + T) as a satellite solver for the theory $R$. Thus, the DPLL(T) part performs on ground clauses and literals, whereas $\Gamma$ works on non-ground clauses and unit ground $R$-clauses ($R$-literals). For deducing new clauses, $\Gamma$ takes premises from the current clause set and $R$-literals from the current sequence of assigned literals (the sequence represents the candidate model built thus far). If (i) $\mathrm{T}_1$, $\mathrm{T}_2$, ... are pairwise-disjoint and stably-infinite [ABRS09, ABRS05], (ii) $R$ is variable-inactive [ABRS05, BE10, BGN+06] and disjoint from each theory $\mathrm{T}_i$, and (iii) $\Gamma$ follows a fair strategy, then DPLL($\Gamma$ + T) is refutationally complete on all input problems $R \cup P$, where $R$ and $P$ are as defined above. In order to deal with quantified formulae which do not comply with all the above requirements, DPLL($\Gamma$ + T)-based SMT-solvers rely on E-matching [dMB07, GBT07], a heuristic technique for instantiating quantified variables. The heuristic support for quantifiers is the source of DPLL(T)'s incompleteness, hence incompleteness of DPLL($\Gamma$ + T), on quantified formulae. One of the most efficient implementations of DPLL($\Gamma$ + T) existing is an SMT Solver Z3 [dMB08c].

DPLL(Γ + T) can be further extended with a reversible *speculative inference rule* [BLdM09, BLdM11] that adds an arbitrary “guess”-clause that is, in general, not entailed by the current clause set (whence *“speculative”* or *“unsound”*). This technique allows to get a decision procedure for some particular axiomatizations $R$, if DPLL(Γ + T) is guaranteed for certain sequences of speculative inferences to terminate in the UNSAT state, if the input formula $R \cup P$ is unsatisfiable, or to finitely saturate the input formula without deriving a contradiction, if $R \cup P$ is satisfiable.

A very good overview of DPLL, DPLL(T), DPLL(Γ + T), DPLL(Γ + T) coupled with Speculative Inferences, and related techniques can be found in [Bon10]. A good survey on the SMT technology and its applications is given in [DMB11].

**Model Evolution (with Equality) Modulo Built-In Theories.** Baumgartner and Tinelli have presented in [BFT08] a new calculus called *Model evolution extended with Linear Integer Arithmetic* $\mathcal{ME}$(LIA) dedicated to reasoning in a combination of FOL and the the theory of Linear Integer Arithmetic LIA.

The calculus is based on the Model Evolution calculus $\mathcal{ME}$ [BT03], an instantiation method, originally invented by the same authors as a first-order logic version of the propositional DPLL procedure. The $\mathcal{ME}$ calculus tries to build a Herbrand model of a given set of *non-equational* clauses by maintaining and manipulating a set of literals, called *context*, which is a finite and compact representation of a candidate model, in a DPLL fashion by applying unification-based versions of DPLL's rules lifted for FOL literals and utilizing the instantiation preorder on terms.

The extended calculus $\mathcal{ME}$(LIA) supports formulae over LIA enriched with extra arithmetic constants (parameters) and arbitrary uninterpreted predicates, where every variable is restricted to range over an integer interval bounded below, and every parameter is restricted to a finite interval over $\mathbb{Z}$. Thus, FOL(LIA) formulae containing variables over a free domain or free function symbols cannot be processed by $\mathcal{ME}$(LIA). The $\mathcal{ME}$(LIA) calculus operates in a manner similar to that of $\mathcal{ME}$, differing from it in the following aspects: (i) the maintained context may contain in addition LIA constraints, which can be any first-order LIA formulae possibly containing parameters; (ii) $\mathcal{ME}$(LIA) uses an LIA-validity decision procedure as a black-box for checking theory reasoning tasks; and (iii) the underlying ordering is based on the “strictly less” ordering $<$ on integers instead of the instantiation ordering on terms used by $\mathcal{ME}$, which coped with the above quantification restrictions on parameters and variables guarantees the existence of only finitely many minimal solutions to an LIA formula.

Model evolution with equality modulo built-in theories $\mathcal{ME}_{\mathrm{E}}$(T) [BT11] generalizes both $\mathcal{ME}$ and $\mathcal{ME}$(LIA) by supporting first-order *equational* logic, and reasoning in an abstract background theory T, which is not restricted in $\mathcal{ME}_{\mathrm{E}}$(T) to the theory of LIA as in $\mathcal{ME}$(LIA). For equational reasoning $\mathcal{ME}_{\mathrm{E}}$(T) adopts the Ordered Paramodulation inference rule [RW69, Wei01]. As the side premise the rule takes an equation only from the current context, whereas the main premise is always a clause from the current clause set; the equation used for the inference is kept with the conclusion to facilitate searching a model. Although $\mathcal{ME}_{\mathrm{E}}$(T) enables reasoning in the equational fragment of FOL(T), and therefore supports arbitrary free function symbols (in contrast to $\mathcal{ME}$(LIA)), it still enjoin all free functions of a non-zero arity to range into a free sort. Thus, $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ lacks treatment of free function symbols ranging into a base sort.

**Constraint Sequent Calculus for FOL with LIA.** The Constraint Sequent Calculus of Rümmer [Rüm08a, Rüm08b] is aimed at reasoning in a combination of the free first-order logic FOL and the theory of linear integer arithmetic LIA. The core of the calculus combines the free-variable tableaux with incremental closure [Gie01] and the Omega quantifier elimination procedure [Pug91]. The former is used to generate arithmetic constraints over variables and constants (parameters) by applying Gentzen-style sequent inference rules. The latter is invoked for deciding validity of the produced LIA constraints, which are conjunctions of equations over variables and constants with quantifier alternations. The calculus admits arbitrary formulae over Linear Integer Arithmetic (Presburger Arithmetic) enriched with arbitrary uninterpreted predicates, where all variables range into the integers $\mathbb{Z}$. Thus, neither free function symbols nor variables over a free domain are supported. The constraint calculus is sound and complete on the classes of purely universal and of purely existential formulae of the indicated FOL(NLA) fragment. The method is implemented in the theorem prover *Princess*[1].

**Linear Arithmetic Superposition Calculus LASCA.** The work [KV07] by Korovin and Voronkov extends the standard superposition calculus with two extra inference rules for reasoning about LA equations and inequations. The resulting calculus, called *LASCA*, is sound; moreover, the calculus is proved complete under a finiteness assumption on the number of rational coefficients appearing in LA-terms in clauses in a LASCA-derivation from an input set of FOL(LA) clauses. Thus, the finiteness assumption effectively prevents quantification over the arithmetic domain. Still, LASCA admits arbitrary clauses over FOL(LA), where the base signature of LA is enriched with arbitrary free sorts and free function symbols, but does not guarantee completeness, in general.

**Cancellative Superposition for (Divisible) Abelian Groups.** Waldmann's Cancellative Superposition and Chaining for totally ordered divisible abelian groups [Wal01] and Cancellative Superposition for cancellative abelian monoids[2] are two refutationally complete superposition calculi, refined for sets of clauses *containing the axiomatizations* of totally-ordered divisible abelian groups and cancellative abelian monoids, respectively. The supported language consists of full free FOL enrichments of the indicated theories.

**Spass+T.** SPASS+T [PW06] is a (rather) *parallel* system combination of the automated theorem prover SPASS [WDF$^+$09] and a generic SMT-procedure used as a black-box solver for ground formulae over arithmetic and free function symbols. SPASS+T admits arbitrary clause sets with variables over both arithmetic and free domains, equality, and free function and predicate symbols ranging into the arithmetic or the free sort. Operationally, SPASS+T consists of three modules: (i) control module on the top of (ii) SPASS and (iii) the SMT-procedure. SPASS

[1]The homepage of *Princess*: `http://www.philipp.ruemmer.org/princess.shtml`.

[2]The theory of the indicated abelian groups and monoids is studied, e.g. , in books [Gil84, Lan93].

passively communicates with the SMT-module via the control module: the controller collects all ground clauses produced by SPASS and transmits them to the SMT-module. Ground clauses passed to the SMT-solver are generated in SPASS by exploiting an additional theory instantiation rule. SPASS+T stops execution whenever: (i) SPASS finds a proof, (ii) the SMT-procedure proves a received formula unsatisfiable, or (iii) SPASS finitely saturates the input clause set without deriving an empty clause, and the SMT module reports satisfiability for all the received formulae. The first two cases signalize the input clause set to be unsatisfiable. Because SPASS+T is an *incomplete* procedure, the third case is inconclusive.

**Hierarchic Reasoning in Local Theory Extensions.** A hierarchic reasoning methodology for (combinations of) local theory extensions has been studied by Sofronie-Stokkermans [SS05] in cooperation with Ganzinger [GSSW06] and Ihlemann [ISS10] and Jacobs [IJSS08]. The distinguishing feature of local theory extensions is that reasoning in them can be reduced to reasoning in the ground fragment.

In the very basic setting, given a background theory T and a set $K$ of non-ground FOL(T) clauses (axioms), the theory extension $\mathrm{T} \cup K$ is said to satisfy the *locality condition*, if for every set $G$ of ground FOL(T) clauses ($G$ is typically obtained by skolemizing a negated universal conjecture whose validity in $K$ modulo T needs to be (dis)proved), a formula $K \cup G$ is unsatisfiable modulo T if and only if the ground formula $K[G] \cup G$ is unsatisfiable modulo T, where $K[G]$ is a set of ground instances of $K$, such that a term with a free top symbol occurs in $K[G]$ whenever the term already occurs in $K$ or $G$.

The theory extension $T \cup K$ is said to satisfy the *extended locality condition*, if for every *augmented clause* $\Gamma \vee G$, where $\Gamma$ is an arbitrary T-sentence (closed formula) and $G$ a set of ground FOL(T) clauses, the formula $K \cup \Gamma \cup G$ is unsatisfiable modulo T if and only if the ground formula $K[\Psi(K,G)] \cup \Gamma \cup G$ is unsatisfiable modulo T, where:

– $K[\Psi(K,G)]$ is a set of ground instances of $K$, such that a term with a free top symbol occurs in $K[\Psi(K,G)]$ whenever the term already occurs in $\Psi(K,G)$; and

– $\Psi$ is a closure operator on the sets of ground terms appearing in its arguments.

Thus, if $\Psi$ is the identity and $\Gamma$ is empty, the extended condition reduces to the basic one.

Hierarchic Reasoning in Local Theory Extensions is implemented in a tool called H-PILoT [ISS09], which tests satisfiability of an input formula $K \cup \Gamma \cup G$, where $K$, $\Gamma$, and $G$ are as specified above, by reducing it to a (ground) T-formula, whose satisfiability is checked by a specialized SMT-procedure invoked from H-PILoT as a black-box solver for T.

The algorithm underlying H-PILoT consists of the following three steps:

1. A given input formula $K \cup \Gamma \cup G$ is *instantiated* to $K[\Psi(K,G)] \cup \Gamma \cup G$.

2. All ground extension terms $t$ (ground terms with a free top symbol) appearing in $K[\Psi(K,G)] \cup \Gamma \cup G$ are *purified* in a bottom-up manner by replacing them with fresh constants $a$ (T-parameters), resulting in a formula

$K_0 \cup \Gamma_0 \cup G_0 \cup D$, where $D$ consists of all definitions $t \approx a$, and $K_0$, $\Gamma_0$, $G_0$ are T-formulae.

3. The obtained formula $K_0 \cup \Gamma_0 \cup G_0 \cup D$ is further reduced by replacing $D$ with a set of clauses

$$\mathsf{Con} = \{\bigwedge_i a_i \approx b_i \to a \approx b \mid (f(a_1,\dots,a_n) \approx a), (f(b_1,\dots,b_n) \approx b) \in D\},$$

whose purpose is to compute the congruence closure with respect to $D$. Satisfiability of the T-formula $K_0 \cup G_0 \cup \Gamma_0 \cup \mathsf{Con}$ is then checked by the background SMT-procedure used as a subroutine in a program.

By locality, if the theory extension $T \cup K$ is local, then the original formula $K \cup \Gamma \cup G$ is (un)satisfiable modulo T if and only if $K[\Psi(K,G)] \cup \Gamma \cup G$ is (un)satisfiable modulo T. Consequently, $K \cup \Gamma \cup G$ is (un)satisfiable modulo T if and only if the T-formula $K_0 \cup \Gamma_0 \cup G_0 \cup \mathsf{Con}$ is (un)satisfiable in T. The algorithm of H-PILoT is *complete for local theory extensions*.

**Hierarchic Superposition with Weak Abstraction.** The most recent work by Baumgartner and Waldmann [BW13] is dedicated to strengthening *Abstraction*, a preprocessing step which transforms every clause in a given set of $FOL(T)$ clauses to a format suitable for hierarchic superposition inference rules [BGW94].

Abstraction suggested in the original work [BGW94] by Bachmair, Ganzinger, and Waldmann is a recursive procedure which "purifies" every clause $C$ in a given clause set $N$ by (i) replacing all occurrences of a term $t = f(t_1,\dots,t_n)$ with a free (resp. theory) top operator symbol $f$, occurring in $C$ immediately below some theory (resp. free) operator symbol $g$ or in an atom $f(t_1,\dots,t_n) \approx g(s_1,\dots,s_m)$, with a fresh variable $x$ over the theory domain, and (ii) adding the negative literal $x \not\approx t$ to the clause, yielding thus a new clause $x \not\approx t \vee C[x/t]$, where $C[x/t]$ is obtained from $C$ by replacing an occurrence of $t$ with $x$. Having purified all literals in $C$ in this way, abstraction yields an equivalent clause $C'$ in which all literals are *pure*, meaning that each literal is built either over the theory signature, or over the free enrichment signature, and the only symbols shared by the literals are variables over the theory domain. The main disadvantage of [BGW94]'s abstraction mechanism is that it may actually destroy sufficient completeness of the initial clause set $N$, a sufficient condition for completeness of the hierarchic calculus SUP(T) on $N$.

The *weak abstraction* algorithm presented by Baumgartner and Waldmann resolves this shortcoming of the original "total" abstraction of Bachmair, Ganzinger, and Waldmann. The main idea of weak abstraction is to abstract out only pure non-variable non-constant T-terms. As extension terms are not abstracted out by weak abstraction, it has also to be performed on every clause derived, in contrast to "total" abstraction which is exhaustively applied only once to an input clause set. In [BW13] it is shown that the hierarchic superposition calculus modified in a straightforward way to fit the new abstraction algorithm is refutationally complete for all clause sets obtained by weak abstraction from sufficiently complete clause sets.

Besides, Baumgartner and Waldmann have proposed new sufficient completeness and hierarchic reduction criteria, which are stronger than the original respective criteria of [BGW94] as they take into account the background theory T. Still, the new criteria are not stronger than the ones developed in our work (see Section 3.4.4, page 76, and Section 3.4.7, page 96, for a detailed comparison of the respective criteria suggested by us and in [BGW94] and [BW13]).

Also, Baumgartner and Waldmann offer in [BW13] a new inference rule Define, capable of dynamically defining ground terms, generated in a SUP(T)-derivation. The inference rule is a generalization of the Basification mechanism developed in our work [KW11, KW12] (see Section 4.2.1 for a detailed exposition of Basification). The overall approach is implemented in a theorem prover called Beagle[1].

In [BW13] the authors claim weak abstraction coupled with the Define rule benefit in a new completeness result for the class of FOL(T) clauses in which all theory-sort terms are ground (the so-called *GBT class*). Unfortunately, the weak abstraction algorithm applied to a GBT clause set, exhaustively saturated with respect to the Define rule (which is required in [BW13] as one more preprocessing step to gain completeness), reduces to the original "total" abstraction procedure of [BGW94]. Moreover, the exhaustive application of the Define rule to a given GBT clause set $N$ emulates the application of Basification to $N$, and thus the both transformations produce the same clause set $N'$ out of the given $N$. In Chapter 4 we show that the SUP(T) calculus with [BGW94]'s "total" abstraction and our Basification algorithm also constitute a complete procedure for the GBT class (see discussion in Section 4.3, page 166.).

The SUP(T) calculus of Baumgartner and Waldmann, adjusted to comply with weak abstraction, has a disadvantage that weakly abstracted clauses may contain compound *impure* FOL(T) terms (these are terms which contain theory and free operators), which demands more complex implementational technicalities and might thus potentially decrease the overall effectiveness of the approach, because theory-simplification and elimination of theory-redundancy has to be done in this case with regard to such impure terms.

In this dissertation, abstraction is regarded simply as a *preprocessing step* and does not impose any particular restrictions on applicability of statements and methods developed here. For this reason, the weak abstraction mechanism of Baumgartner and Waldmann can be safely integrated into our framework instead of [BGW94]'s abstraction method, causing no failure regarding achieved theoretical results.

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

## Outline

The rest of the dissertation is organized as follows:

- In **Chapter 2** we give an overview of the most relevant foundations and notations used throughout the dissertation.
- **Chapter 3** is devoted to presenting the Hierarchic Superposition calculus SUP(T) and proving its completeness. In particular, we introduce our versions of a hierarchic sufficient completeness criterion and hierarchic redundancy criterion. Besides, we develop two basic effective reduction rules which comply with the requirements of our hierarchic redundancy criterion. The chapter ends with a presentation and investigation of a notion of local sufficient completeness criterion and a proof that SUP(T) is complete on locally sufficiently complete clause sets.
- In **Chapter 4** we show that SUP(T) can be turned into a decision procedure for the ground FOL(T) fragment and the non-ground Bernays-Schönfinkel Horn class BSHE(GBST) with equality and ground base sort terms. The decidability results are mainly due to a preprocessing procedure Basification, some extra syntactical modification techniques elaborated here, and the notion of local sufficient completeness criterion introduced in Chapter 3. The chapter ends with a discussion of an example application of the decidability result for the BSHE(GBST) class towards reasoning about and querying ontologies with arithmetical facts.
- **Chapter 5** describes the Hierarchic Superposition calculus modulo Linear Arithmetic SUP(LA), an instance of SUP(T) where the background theory T is linear arithmetic LA. In particular, we have refined the hierarchic reduction rules and develop new effective algorithms for checking LA formulae arising as side conditions of inference and reduction rules of SUP(LA). The chapter ends with a discussion of different implementations of the calculus in system combinations SPASS(LA) of the automated theorem prover SPASS with different satellite LA-solvers, and of application of SUP(LA) to reachability problems of transition systems.
- In **Chapter 6** we study the Hierarchic Superposition calculus modulo Non-Linear Arithmetic SUP(NLA), an instance of SUP(T) where the background theory T is non-linear arithmetic NLA over the reals including transcendental functions. The calculus is implemented in SPASS(iSAT), a system combination of SPASS and an SMT solver iSAT for non-linear arithmetic. Here, we elaborate the issues related to the hierarchic combination FOL(NLA) due to undecidability of NLA involving transcendental functions. Particularly, we present a series of simplification and approximation techniques for treating NLA formulae that eventually enables a complete automatic behavior of SPASS(iSAT) on various scenarios of collision avoidance protocols.
- In **Chapter 7** we conclude and discuss possible directions of future work.

# CHAPTER 2

# Preliminaries

In this chapter we introduce the basic notions and definitions exploited throughout the whole thesis. A more detailed introduction can be found in [Fit90], [BN98], and [Wei01]. Notions specific for particular topics will be given in the respective chapters.

## 2.1 Signatures

DEFINITION 2.1 ► Signature

A ***many-sorted signature*** $\Sigma$ is a pair

$$\Sigma \overset{\text{def}}{=\!=} (\mathcal{S}, \Omega),$$

consisting of a finite non-empty set $\mathcal{S}$ of ***sort symbols***, and a set $\Omega$ of ***operator symbols***, such that

(i) every operator symbol $f \in \Omega$ has a unique sort declaration $f : \mathsf{S}_1 \times \cdots \times \mathsf{S}_n \to \mathsf{S}$, indicating the sorts of arguments and the range sort of $f$, respectively, for some $\mathsf{S}_1, \ldots, \mathsf{S}_n, \mathsf{S} \in \mathcal{S}$ and $n \geq 0$; and

(ii) for every sort $\mathsf{S} \in \mathcal{S}$ there exists at least one operator symbol $f \in \Omega$ such that $f : \mathsf{S}_1 \times \cdots \times \mathsf{S}_n \to \mathsf{S}$, for some $\mathsf{S}_1, \ldots, \mathsf{S}_n \in \mathcal{S}$ and $n \geq 0$.

■

The number $n$ in a sort declaration $f : \mathsf{S}_1 \times \cdots \times \mathsf{S}_n \to \mathsf{S}$ is called the ***arity*** of $f$ and denoted by *arity*$(f)$. A function symbol $a : \to \mathsf{S}$ of the zero arity is called a ***constant***. We require that for every sort $\mathsf{S} \in \mathcal{S}$ there exists at least one ground term[1]

In addition to the signature $\Sigma$, we always assume a variable set $\mathcal{X}$, disjoint from $\Omega$, such that for every sort $\mathsf{S} \in \mathcal{S}$ there exists a countably infinite subset of $\mathcal{X}$ consisting of variables of the sort $\mathsf{S}$. Every variable $x \in \mathcal{X}$ has a unique sort declaration $x : \mathsf{S}$, for some $\mathsf{S} \in \mathcal{S}$, indicating the sort of $x$.

## 2.2 Terms. Formulae. Expressions

DEFINITION 2.2 ► Term

Given a signature $\Sigma = (\mathcal{S}, \Omega)$, a sort $\mathsf{S} \in \mathcal{S}$ and a variable set $\mathcal{Y} \subseteq \mathcal{X}$, where $\mathcal{X}$ is the variable set underlying the signature $\Sigma$, the set $T_\Omega(\mathsf{S}, \mathcal{Y})$ of all ***terms*** of the sort $\mathsf{S}$ built over the operator symbols $\Omega$ (signature $\Sigma$) and variables $\mathcal{Y}$ is recursively defined as the smallest set containing

(i) every variable $y$, such that $y \in \mathcal{Y}$ and $y : \mathsf{S}$,

(ii) all terms $f(t_1, \ldots, t_n)$, where

- $f \in \Omega$ and $f : \mathsf{S}_1, \ldots, \mathsf{S}_n \to \mathsf{S}$,
- $t_i \in T_\Omega(\mathsf{S}_i, \mathcal{Y})$ and $\mathsf{S}_i \in \mathcal{S}$, for every $i \in \{1, \ldots, n\}$.

■

The sort of a term $t$ is denoted by *sort*$(t)$. A term not containing a variable is called ***ground***. For the sake of simplicity we often write:

- $T_\Omega(\mathcal{S}', \mathcal{Y})$ for $\bigcup_{\mathsf{S} \in \mathcal{S}'} T_\Omega(\mathsf{S}, \mathcal{Y})$, the set of all terms of sorts in $\mathcal{S}' \subseteq \mathcal{S}$ built over $\Sigma$ and variables $\mathcal{Y} \subseteq \mathcal{X}$;
- $T_\Omega(\mathcal{Y})$ for $T_\Omega(\mathcal{S}, \mathcal{Y})$, the set of all terms over $\Sigma$ and variables $\mathcal{Y} \subseteq \mathcal{X}$;
- $T_\Omega(\mathcal{X})$ for $T_\Omega(\mathcal{S}, \mathcal{X})$, the set of all terms over $\Sigma$;
- $T_\Omega(\mathcal{S}')$ for $T_\Omega(\mathcal{S}', \emptyset)$, the set of all ground terms over $\Sigma$ of sorts $\mathcal{S}' \subseteq \mathcal{S}$;

[1]The notion of a (ground) term is defined below.

- $T_\Omega(\mathsf{S})$ for $T_\Omega(\{\mathsf{S}\}, \emptyset)$, the set of all ground terms over $\Sigma$ of a sort $\mathsf{S} \in \mathcal{S}$;
- $T_\Omega$ for $T_\Omega(\mathcal{S})$, the set of all ground terms over $\Sigma$.

◄ DEFINITION 2.3
Equation. Atom. Literal

An ***equation*** over the signature $\Sigma$ is a multiset of two terms $s, t \in T_\Omega(\mathsf{S}, \mathcal{X})$ of the same sort $\mathsf{S} = sort(s) = sort(t)$, written $s \approx t$. Any equation is an ***atom***. An atom or the negation of an atom is called a ***literal***. ■

A literal is ***positive*** if it is an atom, and ***negative***, otherwise. A negative literal $\neg(s \approx t)$, also called ***disequation***, is often written as $s \not\approx t$. When the matter of discussion does not depend on the sign of a literal, we write $s \mathrel{\dot\approx} t$, where $\dot\approx$ is either $\approx$ or $\not\approx$. Given a literal $L = (s \mathrel{\dot\approx} t)$, the ***complementary literal*** of $L$, denoted by $\overline{L}$, is defined as $\overline{L} = (s \not\approx t)$ if $\dot\approx = \approx$, and $\overline{L} = (s \approx t)$ otherwise. For a set $M = \{L_1, L_2, \ldots\}$ of literals, $\overline{M}$ denotes the set $\{\overline{L}_1, \overline{L}_2, \ldots\}$ consisting of $M$'s complementary literals.

The above notion of signature does not naturally support predicates. We override this problem by extending a given signature for every predicate symbol $P$ taking arguments of sorts $\mathsf{S}_1, \ldots, \mathsf{S}_n$ as follows: (i) add a distinct sort $\mathsf{S}_P$ to $\mathcal{S}$, (ii) introduce a fresh constant $true_P$ of the sort $\mathsf{S}_P$ to $\Omega$, and (iii) encode every predicate $P$ as a function $P : \mathsf{S}_1, \ldots, \mathsf{S}_n \to \mathsf{S}_P$. Thus, predicative atoms are turned into equations $P(t_1, \ldots, t_n) \approx true_P$, written usually as $P(t_1, \ldots, t_n)$.

◄ DEFINITION 2.4
Formula

The set $F_\Omega$ of ***formulae*** over the signature $\Sigma$ is the smallest set containing

- the logical constants $\top$ and $\bot$,
- every atom over $\Sigma$,
- applications of boolean connectives: negation $\neg F$, conjunction $F \wedge G$, disjunction $F \vee G$, implication $F \to G$, and equivalence $F \leftrightarrow G$, for any formulae $F, G \in F_\Omega$,
- quantifier applications: $\forall x.F$, $\exists x.F$, where $x \in \mathcal{X}$ and $F \in F_\Omega$.

■

A formula not containing a quantifier is called ***quantifier-free***. Implication $F \to G$ and equivalence $F \leftrightarrow G$ are abbreviations for $\neg F \vee G$ and $(F \to G) \wedge (G \to F)$, respectively.

An ***expression*** is either a term or a formula. It follows from the definitions of terms and formulae that expressions have a tree-like structure. For referring to certain subtrees of an expression, called subexpressions, we use sequences of natural numbers, called positions.

◄ DEFINITION 2.5
Subexpression
Position

***Subexpression*** $e/p$ at ***position*** $p$ in an expression $e \in F_\Omega \cup T_\Omega(\mathcal{X})$ is recursively defined as:

- $e/p = e$, if $p$ is empty;
- $e/p = e_i/q$, if $p = iq$, where $iq$ is the concatenation of the two positions $i$ and $q$, and either
    - $e = f(e_1, \ldots, e_n)$ and $i \in \{1, \ldots, n\}$, or
    - $e = \neg e_1$ and $i = 1$, or
    - $e = e_1 \vee e_2$ or $e = e_1 \wedge e_2$, and $i \in \{1, 2\}$, or

- $e = \forall\, e_1.e_2$ or $e = \exists\, e_1.e_2$, where $e_1 \in \mathcal{X}$ and $i \in \{1,2\}$,

The set of all positions in $e$, for each of which the respective subexpression is defined, is denoted by $\rho(e)$. ■

The length of a position $p$ is its length as a sequence, denoted by $|p|$. If $|p| = 0$, i.e. $p$ is empty, then it is called the top position and denoted by $p = \varepsilon$.

DEFINITION 2.6 ► Prefix Order
Any two positions $p, q$ can be related to each other with respect to the following ***prefix orders***:

– $p$ ***above*** $q$ (or equivalently, $q$ ***below*** $p$), denoted by $p \leq q$, if $q = pp'$ for some $p'$.

– $p$ is ***strictly above*** $q$ (or equivalently, $q$ ***strictly below*** $p$), denoted by $p < q$, if $p \leq q$ and not $q \leq p$. Note that $p < q$ iff $q = pp'$ for some $p' \neq \varepsilon$;

– $p$ and $q$ ***parallel***, denoted by $p \parallel q$, if neither $p \leq q$ nor $q \leq p$.

■

An expression $e$ is said to ***contain*** another expression $e'$ if $e'$ is a subexpression of $e$, and $e'$ is called a ***strict subexpression*** of $e$ if, in addition, $e' \neq e$; in the both cases, $e'$ is said to ***occur*** in $e$, which is denoted by $e[e']$. We also write $e[e']_p$ to emphasize that the subexpression $e'$ occurs in $e$ at a position $p \in \rho(e)$. Ambiguously, we write $e[e'']$ to denote the result of replacing the occurrence of $e'$ in $e$ with $e''$ (we also write $e[e'']_p$ to emphasize that the replacement is done at the position $p$). We call $e'$ a ***immediate subexpression*** of $e$ if $e' = e/p$ for some $p \in \rho(e)$ and $|p| = 1$. We call a subexpression $e'$ a ***subterm*** if $e'$ is a term, and a ***subformula*** if $e'$ is a formula.

The ***size*** of an expression $e$, written $|e|$, is the number of all subexpressions occurring in it; evidently, the size of $e$ equals the cardinality of the set $\rho(e)$, that is $|e| \overset{\text{def}}{=} |\rho(e)|$. The ***depth*** of an expression is the maximal length of a position in the expression: $depth(e) \overset{\text{def}}{=} max\{|p| \mid p \in \rho(e)\}$. The set of *all* variables occurring in an expression $e$ is denoted by $var(e)$ and formally defined as $var(e) \overset{\text{def}}{=} \{x \in \mathcal{X} \mid x = e/p,\ p \in \rho(e)\}$. The expression is ***ground*** if $var(e) = \emptyset$.

EXAMPLE 2.7 ► If $e = \neg P(f(x, g(a)))$, then:

– let $e' = P(f(x, g(a)))$; $e'$ is an immediate subformula of $e$ occurring at position $p = (1)$, and $x$ and $g(a)$ strict subterms of $e$ occurring at positions $q_1 = (1,1,1)$ and $q_2 = (1,1,2)$, respectively;

– the position $p$ is strictly above $q_1$ and $q_2$ as $q_1 = pp_1 = (1)(1,1) = (1,1,1)$ and $q_2 = pp_2 = (1)(1,2) = (1,1,2)$, where $p_1 = (1,1)$ and $p_2 = (1,2)$ are the positions of $x$ and $g(a)$, respectively, in the formula $e' = P(f(x, g(a)))$;

– the subterms $x$ and $g(a)$ are parallel;

– the subterms $x$ and $g(a)$ occur in the term $f(x, g(a))$ at positions (1) and (2), respectively, thus they are immediate subterms thereof;

– the formula $e[b]_{q_1} = \neg P(f(b, g(a)))$ is obtained from $e$ by replacing $x$ with $b$;

- $p, q_1, q_2 \in \rho(e)$ and $p_1, p_2 \in \rho(e')$, but also $p_1 \in \rho(e)$ and $p_2 \notin \rho(e)$, because $e/p_1$ is defined and equals $f(x, g(a))$, whereas $e/p_2$ is not defined;
- $|p| = 1$, $|q_1| = |q_2| = 3$, and $|p_1| = |p_2| = 2$;
- $\rho(e) = \{\varepsilon, (1), (1,1), (1,1,1), (1,1,2), (1,1,2,1)\}$ and $|e| = 6$;
- $var(e) = \{x\}$.

■

◄ DEFINITION 2.8
Equivalence Relation
Congruence Relation
Equivalence class

An ***equivalence*** relation $\sim$ on a term set $T_\Omega(\mathcal{X})$ is a reflexive, transitive, symmetric binary relation on $T_\Omega(\mathcal{X})$. Two terms $s$ and $t$ are called ***equivalent***, if $s \sim t$.

An equivalence $\sim$ is called a ***congruence*** if $s \sim t$ implies $u[s] \sim u[t]$, for all terms $s, t, u \in T_\Omega(\mathcal{X})$.

Given a term $t \in T_\Omega(\mathcal{X})$, the set of all terms equivalent to $t$ is called the ***equivalence class of** $t$ **by*** $\sim$, denoted by $[t]_\sim$:

$$[t]_\sim \overset{\text{def}}{=} \{t' \in T_\Omega(\mathcal{X}) \mid t' \sim t\}.$$

■

If the matter of discussion does not depend on a particular equivalence relation or it is unambiguously known from the context, we simply write $[t]$ instead of $[t]_\sim$.

◄ DEFINITION 2.9
Quotient

The set of all equivalence classes in $T_\Omega(\mathcal{X})$ defined by the equivalence relation $\sim$ is called a ***quotient by*** $\sim$, denoted by $T_\Omega/\sim$:

$$T_\Omega/\sim \overset{\text{def}}{=} \{[t]_\sim \mid t \in T_\Omega(\mathcal{X})\}.$$

■

We write $\sim_E$ to denote the smallest congruence relation "containing" a set of equations $E$, that is, $(l \approx r) \in E$ implies $l \sim_E r$. The equivalence class $[t]_{\sim_E}$ of a term $t$ by the equivalence (congruence) $\sim_E$ is usually denoted, for short, by $[t]_E$. Likewise, we usually write $T_\Omega(\mathcal{X})/E$ for the quotient $T_\Omega(\mathcal{X})/\sim_E$ of $T_\Omega(\mathcal{X})$ by the equivalence (congruence) $\sim_E$.

## 2.3 Substitutions. Unifiers

An occurrence of a variable $x$ in a formula $F$ is called ***bound***, if $p \in \rho(F)$ is the position of the occurrence of $x$ in $F$, and $p = qq'$, such that $F/q = Qx.G$, where $Q \in \{\forall, \exists\}$. The subformula $G$ is called the ***scope*** of the quantifier $Qx$. In other words, a variable $x$ is bound if it occurs in the scope of a quantifier $Qx$. Any other occurrence of a variable is called ***free***. A formula not containing a free occurrence of a variable is called ***closed***. If $x_1, \dots, x_n$ are the variables freely occurring in a formula $F$, then $\forall x_1.\cdots\forall x_n.F$ and $\exists x_1.\cdots\exists x_n.F$ are the ***universal*** and the ***existential closure*** of $F$, respectively.

◄ DEFINITION 2.10
Substitution

A ***substitution*** $\sigma$ is a mapping from variables $\mathcal{X}$ to terms $T_\Omega(\mathcal{X})$ such that

(i) $\sigma(x) \neq x$ for only finitely many variables $x$, and

(ii) every variable $x \in \mathcal{X}$ is mapped to a term $t \in T_\Omega(\mathsf{S}, \mathcal{X})$ of the same sort $\mathsf{S} = sort(x)$.

■

The application $\sigma(x)$ of a substitution $\sigma$ to a variable $x$ is often written in postfix notation as $x\sigma$. The variable set

$$dom(\sigma) \overset{\text{def}}{=} \{x \in \mathcal{X} \mid x\sigma \neq x\}$$

is called the ***domain*** of $\sigma$; the term set

$$im(\sigma) \overset{\text{def}}{=} \{x\sigma \mid x \in dom(\sigma)\}$$

is called the ***image*** of $\sigma$; and the variable set

$$cdom(\sigma) \overset{\text{def}}{=} \bigcup_{t \in im(\sigma)} var(t)$$

is called the ***co-domain*** of $\sigma$. From the above definition of a substitution it follows that $dom(\sigma)$ is finite for any substitution $\sigma$. The composition of two substitutions $\sigma$ and $\tau$ is written as a juxtaposition $\sigma\tau$, i.e. $t\sigma\tau = (t\sigma)\tau$.

We write $\sigma = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ if $dom(\sigma) = \{x_1, \dots, x_n\}$ and $x_i\sigma = t_i$ for every $i \in \{1, \dots, n\}$. The substitution $\sigma$ is also often written as $[t_1/x_1, \dots, t_n/x_n]$.

The ***modification*** of a substitution $\sigma$ at a variable $x$ is defined as follows:

$$\sigma[x \mapsto t](y) \overset{\text{def}}{=} \begin{cases} t, & \text{if } y = x \\ \sigma(y), & \text{otherwise.} \end{cases}$$

A substitution $\sigma$ is identified with its extension to expressions and defined as following:

- $\bot\sigma = \bot$,
- $\top\sigma = \top$,
- $(f(t_1, \dots, t_n))\sigma = f(t_1\sigma, \dots, t_n\sigma)$,
- $(s \approx t)\sigma = s\sigma \approx t\sigma$,
- $(\neg F)\sigma = \neg(F\sigma)$,
- $(F \circ G)\sigma = F\sigma \circ G\sigma$, where $\circ \in \{\vee, \wedge\}$,
- $(Qx.F)\sigma = Qz.(F\sigma[x \mapsto z])$, where $Q \in \{\forall, \exists\}$ and $z$ is a fresh variable.

The result $e\sigma$ of applying a substitution $\sigma$ to an expression $e$ is called an ***instance*** of $e$. The substitution $\sigma$ is called ***ground*** if it maps every domain variable to a ground term. If the application of a substitution $\sigma$ to an expression $e$ produces a ground expression $e\sigma$, then $e\sigma$ is called a ***ground instance*** of $e$. The ***set of all ground instances*** of $e$ is denoted by $gi(e)$. A ground substitution $\sigma$ is called ***grounding for an expression*** $e$, if $e\sigma$ is ground. A substitution $\sigma$ is called a ***variable renaming*** if $im(\sigma) \subseteq \mathcal{X}$ and for any $x, y \in \mathcal{X}$, if $x \neq y$ then $x\sigma \neq y\sigma$. The latter condition for a substitution to be a variable renaming is essentially the same as the injectivity property. Nevertheless, in the context of substitutions we do not use the notions like "injectivity" or "bijectivity" in order to avoid any confusion, because the notions of domain and co-domain of a substitution differ from those of a function, which are commonly used in such areas as *common mathematics* or *category theory*.

DEFINITION 2.11 ► Two terms $s$ and $t$ are said to be ***unifiable*** if there exists a substitution $\sigma$ such that $s\sigma = t\sigma$, the substitution $\sigma$ is called then a ***unifier*** of $s$ and $t$. The unifier $\sigma$

Unifier
Matcher

is called a ***most general unifier***, written $\sigma = mgu(s,t)$, if any other unifier $\tau$ of $s$ and $t$ can be represented as $\tau = \sigma\tau'$, for some substitution $\tau'$. A substitution $\sigma$ is called a ***matcher*** from $s$ to $t$ if $s\sigma = t$. ■

Notably, most general unifiers are unique up to variable renaming. Given $n$ pairs of terms $(s_i, t_i)$, a substitution $\sigma$ is called a ***simultaneous unifier*** of $(s_1, t_1)$, ..., $(s_n, t_n)$ if $s_i\sigma = t_i\sigma$ for every $i \in \{1,\dots,n\}$. Simultaneous most general unifiers and matchers are defined accordingly.

## 2.4 Multisets. Orderings. Rewrite Systems

◄ DEFINITION 2.12
Multiset

A ***multiset*** over a set $X$ (which can be a set of terms, atoms, literals, clauses, etc., or sets thereof) is a mapping $M$ from $X$ to the non-negative integer (natural) number. ■

Intuitively, $M(x)$ specifies the number of occurrences of $x$ in $M$. We say that $x$ is an element of $M$ if $M(x) > 0$. The union, intersection, and difference of multisets are defined by the identities:

- $(M_1 \cup M_2)(x) = M_1(x) + M_2(x)$,
- $(M_1 \cap M_2)(x) = min\{M_1(x), M_2(x)\}$,
- $(M_1 \setminus M_2)(x) = min\{0, M_1(x) - M_2(x)\}$.

A multiset $M$ is finite if the set $\{x \mid M(x) > 0\}$ is finite. If $M$ is a multiset and $S$ a set, we write $M \subseteq S$ to indicate that every element of $M$ is an element of $S$, and use $M \setminus S$ to denote the multiset $M'$ for which $M'(x) = 0$ for any $x$ in $S$, and $M'(x) = M(x)$, otherwise. We often use sequence- or set-like notation to denote multisets and write, for instance, $M, N$ instead- of $M \cup N$, or $M, L$ instead of $M \cup \{L\}$. For example, $\{F, G, G\}$ denotes the multiset $M$ over formulae for which $M(F) = 1$, $M(G) = 2$, and $M(H) = 0$, for any other formula $H$.

◄ DEFINITION 2.13
Ordering

A ***partial ordering*** $\succeq$ is a transitive, reflexive, and antisymmetric binary relation. A ***strict ordering*** $\succ$ is a transitive and irreflexive relation. ■

Every partial ordering $\succeq$ induces a strict ordering $\succ$ by $t \succ s$ iff $t \succeq s$ and $t \neq s$. A strict ordering is said to be ***total*** if for any two distinct elements $s$ and $t$ either $s \succ t$ or $t \succ s$. We often write $t \succ s_1, \dots, s_n$ to denote that $t \succ s_i$, for every $i \in \{1,\dots,n\}$. A strict ordering $\succ$ is said to be ***well-founded*** if there is no infinite descending chain $s_1 \succ s_2 \succ \dots$ of elements.

We say that an ordering $\succ$:

- is ***stable under substitutions***, if $e_1 \succ e_2$ implies $e_1\sigma \succ e_2\sigma$, for all expressions $e_1, e_2$ and substitutions $\sigma$;
- is ***compatible with contexts***, if $e_1 \succ e_2$ implies $e[e_1]_p \succ e[e_2]_p$, for all expressions $e, e_1, e_2$ and all positions $p \in \rho(e)$;
- has the ***subterm property***, if $e[e'] \succ e'$, for all expressions $e$ and strict subexpressions $e'$ of $e$;

DEFINITION 2.14 ► Rewrite Ordering / Reduction Ordering / Simplification Ordering

A ***rewrite ordering*** is a strict ordering that is stable under substitutions and compatible with contexts; a ***reduction ordering***, a well-founded rewrite ordering; and a ***simplification ordering***, a reduction ordering with the subterm property. ■

The ***lexicographic extension*** $\succ_{\text{lex}}$ on tuples of some strict ordering is defined by $(t_1,\dots,t_n) \succ_{\text{lex}} (s_1,\dots,s_n)$ if $t_i \succ s_i$ for some $i \in \{1,\dots,n\}$, and $t_j = s_j$ for every $1 \le j \le i$.

The ***multiset extension*** $\succ_{\text{mul}}$ of a partial ordering $\succ$ on a set $S$ is defined as follows: $M \succ_{\text{mul}} N$ if $M \neq N$ and for every $x$ such that $N(x) > M(x)$, there exists some $y$ such that $M(y) > N(y)$ and $y \succ x$. Given a multiset, any smaller multiset can be obtained by (repeatedly) replacing an element by zero or more occurrences of smaller elements. If an ordering $\succ$ is total (respectively, well-founded), so is its multiset extension [DM79].

DEFINITION 2.15 ► Lexicographic Path Ordering

Let $\Sigma = (\mathcal{S},\Omega)$ be a signature, where $\mathcal{S}$ is a set of sorts, $\Omega$ a set of operator symbols, and $\mathcal{X}$ the underlying set of variables.

Let $\succ$ be a total strict well-founded ordering on operator symbols in $\Omega$, called ***precedence***. ***Lexicographic path ordering*** $\succ_{\text{lpo}}$, ***LPO*** for short, is defined as follows: if $t, s$ are terms in $T_\Omega(\mathcal{X})$, then $t \succ_{\text{lpo}} s$ if

1. $s = x \in \mathcal{X}$, $x \in var(t)$, and $t \neq s$, or
2. $t = f(t_1,\dots,t_n)$, $s = g(s_1,\dots,s_m)$, and
   a) $t_i \succeq_{\text{lpo}} s$ for some $i \in \{1,\dots,n\}$, or
   b) $f \succ g$ and $t \succeq_{\text{lpo}} s_j$ for every $j \in \{1,\dots,m\}$, or
   c) $f = g$, $t \succeq_{\text{lpo}} s_j$ for every $j \in \{1,\dots,m\}$, and $(t_1,\dots,t_n)\ (\succ_{\text{lpo}})_{\text{lex}}\ (s_1,\dots,s_m)$.

■

The LPO is a reduction ordering. Moreover, if the precedence $\succ$ is total on $\Omega$, then LPO $\succ_{\text{lpo}}$ augmenting $\succ$ is total on the set of ground terms $T_\Omega$. Ambiguously, we will write $\succ$ instead of $\succ_{\text{lpo}}$.

DEFINITION 2.16 ► Rewrite Relation

A binary relation $\to$ on terms is called a ***rewrite relation*** if it is a rewrite ordering. ■

Thus, expanding the notion of a rewrite relation $\to$ it can be defined as follows: $s \to t$ implies $u[s\sigma]_p \to u[t\sigma]p$, for any terms $s$, $t$, and $u$, position $p \in \rho(u)$, and substitution $\sigma$.

Rewrite relations are denoted by arrows, possibly under- and/or overscripted. If $\to$ is a rewrite relation, we denote by $\to^0$ identity; by $\leftarrow$ its inverse; by $\to^+$ its transitive closure; by $\to^*$ its transitive-reflexive closure. We write $\leftrightarrow$, $\leftrightarrow^+$, and $\leftrightarrow^*$ for symmetric, transitive symmetric, and reflexive transitive symmetric closure, respectively.

Two terms $t_1$ and $t_2$ are called ***joinable*** if there exists a term $s$, to which $t_1$ and $t_2$ rewrite: $t_1 \to^* s \leftarrow^* t_2$; this is denoted by $t_1 \downarrow t_2$. A rewrite relation $\to$ is called

- ***confluent*** if $t_1 \leftarrow^* s \to^* t_2$ implies $t_1 \downarrow t_2$;
- ***terminating*** (or ***well-founded***) if there is no infinite sequence $t_1 \to t_2 \to \dots$;

– ***Church-Rosser*** if $t_1 \leftrightarrow^* t_2$ implies $t_1 \downarrow t_2$.

A term that cannot be rewritten is said to be in normal form or ***irreducible*** (with respect to $\rightarrow$). A ***normal form*** of $s$ is any irreducible term $t$ for which $s \rightarrow^* t$.

◄ DEFINITION 2.17 Rewrite Rules Term Rewrite System

Let $\succ$ be a strict ordering on terms. A ***rewrite rule*** is an equation $l \approx r$ between two terms $l$ and $r$, such that $l \succ r$, denoted by $l \rightarrow r$. A ***term rewrite system*** $R$, or a TRS for short, is a set of rewrite rules (with respect to the term ordering $\succ$). ■

Given a rewrite system $R$, the smallest rewrite relation containing $R$ is denoted by $\rightarrow_R$. The rewrite relation $\rightarrow_R$ is called the ***rewrite relation induced by*** $R$. Thus, we write $s \rightarrow_R t$ to indicate that there exist a rewrite rule $(l \approx r) \in R$, position $p \in \rho(s)$, and matcher $\sigma$, such that the subterm of $s$ at $p$ equals $l\sigma$, and replacing $l\sigma$ in $s$ yields $t$:

$$s \rightarrow_R t \overset{\text{def}}{\iff} s/p = l\sigma \text{ and } t = s[r\sigma]_p.$$

We write $t = s{\downarrow_R}$ to denote that $t$ is a normal form of $s$ with respect to the rewrite relation $\rightarrow_R$. Notions $\rightarrow^0_R$, $\rightarrow^+_R$, $\rightarrow^*_R$, etc., are defined accordingly.

◄ DEFINITION 2.18 Convergent TRS

A term rewrite system $R$ is called ***convergent*** if the rewrite relation $\rightarrow_R$ is confluent and terminating. ■

◄ LEMMA 2.19 [BN98]

*A rewrite relation $\rightarrow$ is Church-Rosser iff confluent. Convergent rewrite systems define unique normal forms.*

◄ THEOREM 2.20 [BN98]

*A rewrite system $R$ terminates iff there exists a reduction ordering $\succ$ such that $l \succ r$, for each rule $l \rightarrow r$ in $R$.*

A rewrite system $R$ is called ***right-reduced*** if for all rewrite rules $l \rightarrow r$ in $R$, the term $r$ is irreducible by $R$. A rewrite system $R$ is called ***left-reduced*** if for all rewrite rules $l \rightarrow r$ in $R$, the term $l$ is irreducible by $R \setminus \{l \rightarrow r\}$. A rewrite system is called ***reduced*** if it is left- and right-reduced.

◄ THEOREM 2.21 [Hue80]

*Left-reduced terminating rewrite systems are convergent.*

## 2.5 Clauses

◄ DEFINITION 2.22 Clause

A ***clause*** $C$ over a signature $\Sigma$ is a pair of multisets of $\Sigma$-atoms, written

$$C \overset{\text{def}}{=} \Gamma \rightarrow \Delta.$$

The multiset $\Gamma$ is called the ***antecedent***; the multiset $\Omega$, the ***succedent***. ■

Logically, the atoms in $\Gamma$ denote negative literals while the atoms in $\Delta$ denote the positive literals in the clause. All variables of a clause are implicitly universally quantified. Semantically, a clause

$$C = \{A_1, \ldots, A_m\} \rightarrow \{B_1, \ldots, B_n\}$$

represents an implication

$$A_1 \wedge \ldots \wedge A_m \rightarrow B_1 \vee \ldots \vee B_n,$$

or equivalently, a disjunction

$$\neg A_1 \vee \ldots \vee \neg A_m \vee B_1 \vee \ldots \vee B_n.$$

Sometimes, we interpret clauses as multisets of their literals; thus, given a clause

$$C = \{A_1, \ldots, A_m\} \rightarrow \{B_1, \ldots, B_n\},$$

we may write

$$C = \{L_1, \ldots, L_{m+n}\},$$

where

$$\{L_1, \ldots, L_{m+n}\} = \{\neg A_1, \ldots, \neg A_m, B_1, \ldots, B_n\}.$$

The ***class of all clauses*** over a signature $\Sigma$ is denoted by $Cl_\Sigma$.

We usually write $\Gamma_1, \Gamma_2$ instead of $\Gamma_1 \cup \Gamma_2$; $\Gamma, A$ instead of $\Gamma \cup \{A\}$; and $A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$ instead of $\{A_1, \ldots, A_m\} \rightarrow \{B_1, \ldots, B_n\}$.

The ***length*** of a clause $C = \Gamma \rightarrow \Delta$ is the sum of the numbers of atoms in the antecedent and succedent of $C$, denoted by $|C|$, i.e. $|C| \overset{\text{def}}{=} |\Gamma| + |\Delta|$. The ***depth*** of $C$ is the maximal depth of its literals: $depth(C) \overset{\text{def}}{=} max\{depth(L) \mid L \in C\}$.

A term $t$ is said to ***occur in a clause*** $C$, if there is a literal $L = (l \approx r)$ in $C$ such that $t$ is a subterm of $l$ or $r$. A term $t$ is said to ***occur in a clause set*** $N$, if there is a clause $C$ in $N$ such that $t$ occurs in $C$.

The clause $C$ is called

- ***negative*** (***positive***) if it contains only negative (positive) literals;
- ***Horn*** if its succedent contains at most one atom, i.e. $|\Delta| \leq 1$;
- ***unit*** if it consists of a single literal, i.e. $|C| = 1$;
- ***empty*** if it does not contain a literal, i.e. $|C| = 0$; we write $\square$ to denote an empty clause (which can be interpreted as an empty disjunction or an implication $\top \rightarrow \bot$).

In Section 3.2, we also introduce a special type of clauses, called *abstracted*, Definition 3.20, which are specific for the hierarchic superposition calculus.

The result of an application of a substitution $\sigma$ to a clause $C = \{L_1, \ldots, L_n\}$ is the clause $C\sigma$ obtained from $C$ by applying $\sigma$ to every literal $L_i$ of $C$, thus $C\sigma = \{L_1\sigma, \ldots, L_n\sigma\}$. The set of all ground instances of a clause $C$ is denoted by $gi(C)$. The set $gi(N)$ of all ground instances of a clause set $N$ is the union of sets of all ground instances of every clause in $N$, i.e. $gi(N) = \bigcup_{C \in N} gi(C)$.

DEFINITION 2.23 ► Selection Function

A ***selection function*** *Sel* assigns to each clause $C = \Gamma \rightarrow \Delta$ a multiset of atoms in the antecedent $\Gamma$ of the clause. The atoms (and corresponding negative literals in $C$) returned by $Sel(C)$ are called *selected* in the clause $C$. ■

A selection function *Sel* may select atoms *arbitrarily* irregardless of ordering[1] restrictions. Selection functions are aimed to reduce the search space of a calculus. The strategy, when all atoms are selected in the antecedent of a clause, is called ***eager selection***. Eager selection is an instance of a ***positive superposition strategy***, which is called this way because in this strategy only positive clauses can be

[1]Orderings are introduced in Section 2.4.

superposed into other clauses. In Chapter 4, we shall exploit eager selection as it allows to keep the length of Horn abstracted clauses bounded.

Any ordering on terms can be extended to clauses in the following way. We consider clauses as multisets of occurrences of equations.

◄ DEFINITION 2.24
Equation Occurrence

An occurrence of an equation $s \approx t$ in the antecedent is identified with the multiset $\{s, s, t, t\}$, the occurrence of an equation in the succedent is identified with the multiset $\{s, t\}$. ■

Now we overload $\succ$ on literal occurrences to be the multiset extension of $\succ$ on terms, and $\succ$ on clauses to be the multiset extension of $\succ$ on literal occurrences. Observe that an occurrence of an equation $s \approx t$ in the antecedent is strictly greater than an occurrence of $s \approx t$ in the succedent.

◄ DEFINITION 2.25
(Strictly) Maximal/Minimal Equation Occurrences/Literals

An antecedent or succedent occurrence of an equation $s \approx t$ is ***maximal/minimal*** in a clause $C$ if there is no occurrence of an equation in $C$ that is strictly greater/smaller then the occurrence $s \approx t$ with respect to $\succ$. An antecedent or succedent occurrence of an equation $s \approx t$ is ***strictly maximal/minimal*** in a clause $C$ if there is no occurrence of an equation $C$ that is greater/smaller then or equal to the occurrence $s \approx t$ with respect to the ordering $\succ$. A literal is (strictly) maximal/minimal if the corresponding equation occurrence is (strictly) maximal/minimal. ■

## 2.6 Semantics. Homomorphisms. Specifications

◄ DEFINITION 2.26
Algebra

Let $\Sigma = (\mathcal{S}, \Omega)$ be a signature with a set of sorts $\mathcal{S}$ and operator set $\Omega$. A $\Sigma$-***algebra*** $\mathcal{A}$, also called $\Sigma$-***interpretation***, is a mapping that assigns

- a non-empty carrier set $\mathcal{A}(\mathsf{S})$ to every sort $\mathsf{S} \in \mathcal{S}$, such that $\mathcal{A}(\mathsf{S}_1) \cap \mathcal{A}(\mathsf{S}_2) = \emptyset$ for any distinct sorts $\mathsf{S}_1, \mathsf{S}_2 \in \mathcal{S}$;
- a total function $\mathcal{A}(f) : \mathcal{A}(\mathsf{S}_1) \times \cdots \times \mathcal{A}(\mathsf{S}_n) \to \mathcal{A}(\mathsf{S})$ to every operator $f \in \Omega$, where $f : \mathsf{S}_1 \times \cdots \times \mathsf{S}_n \to \mathsf{S}$.

The set $U_{\mathcal{A}} = \mathcal{A}(\mathcal{S}) = \bigcup_{\mathsf{S} \in \mathcal{S}} \mathcal{A}(\mathsf{S})$ is called the ***universe*** of $\mathcal{A}$. ■

Very often, if the signature $\Sigma$ is irrelevant or clear from the context, we omit the prefix "$\Sigma$-" and use the term "algebra" ("interpretation") when referring to a $\Sigma$-algebra ($\Sigma$-interpretation). Also, we write $f_{\mathcal{A}}$ and $\mathsf{S}_{\mathcal{A}}$ in place of $\mathcal{A}(f)$ and $\mathcal{A}(\mathsf{S})$. The class of all $\Sigma$-algebras is denoted by $\mathscr{A}_{\Sigma}$.

A (variable) ***assignment*** for an algebra $\mathcal{A}$ is a function $v : \mathcal{X} \to U_{\mathcal{A}}$ such that $v(x) \in \mathsf{S}_{\mathcal{A}}$ for every variable $x \in \mathcal{X}$, where $\mathsf{S} = sort(x)$. A ***modification*** $v[x \mapsto e]$ of an assignment $v$ at a variable $x$, where $e \in \mathsf{S}_{\mathcal{A}}$ and $\mathsf{S} = sort(x)$, is the assignment defined as following:

$$v[x \mapsto e](y) \overset{\text{def}}{=} \begin{cases} e, & \text{if } y = x \\ v(y), & \text{otherwise.} \end{cases}$$

Informally speaking, the assignment $v[x \mapsto e]$ is identical to $v$ at every variable except the $x$, which is mapped by $v[x \mapsto e]$ to $e$.

The homomorphic extension $\mathcal{A}(v)$ of $v$ onto terms is a mapping $T_{\Omega}(\mathcal{X}) \to U_{\mathcal{A}}$ defined as

- $\mathcal{A}(v)(x) = v(x)$, where $x \in \mathcal{X}$,
- $\mathcal{A}(v)(f(t_1,\dots,t_n)) = f_{\mathcal{A}}(\mathcal{A}(v)(t_1),\dots,\mathcal{A}(v)(t_n))$, where $f \in \Omega$.

Given a term $t \in T_\Omega(\mathcal{X})$, the value $\mathcal{A}(v)(t)$ is called the ***interpretation*** of $t$ under $\mathcal{A}$ and $v$. If the term $t$ is ground, the value $\mathcal{A}(v)(t)$ does not depend on a particular choice of $v$, for which reason the interpretation of $t$ under $\mathcal{A}$ is denoted by $\mathcal{A}(t)$. Sometimes, for $\mathcal{A}(v)(t)$ and $\mathcal{A}(t)$ we write $t_{\mathcal{A}(v)}$ and $t_{\mathcal{A}}$, respectively.

An algebra $\mathcal{A}$ is called ***term-generated***, if every element $e$ of the universe $U_{\mathcal{A}}$ of $\mathcal{A}$ equals the interpretation $t_{\mathcal{A}}$ of some ground term $t$. Note, that for an arbitrary signature there might be no term-generated algebra; for instance, given a signature $\Sigma = (\mathcal{S} = \{\mathsf{S}_1, \mathsf{S}_2\}, \Omega = \{f : \mathsf{S}_1 \to \mathsf{S}_2, g : \mathsf{S}_2 \to \mathsf{S}_1\})$, no $\Sigma$-algebra is term-generated. In this work we consider only such signatures, for which there exists at least one term-generated algebra.

An algebra $\mathcal{A}$ and an assignment $v$ ***satisfy*** a formula $F \in F_\Omega$, denoted by $\mathcal{A}, v \models F$, if

- $F = \top$, or
- $F = s \approx t$ and $\mathcal{A}(v)(s) = \mathcal{A}(v)(t)$, where $s, t \in T_\Omega(\mathcal{X})$, or
- $F = \neg G$ and $\mathcal{A}, v \not\models G$, where $G \in F_\Omega$, or
- $F = F_1 \vee F_2$ and $\mathcal{A}, v \models F_1$ or $\mathcal{A}, v \models F_2$, where $F_1, F_2 \in F_\Omega$, or
- $F = F_1 \wedge F_2$ and $\mathcal{A}, v \models F_1$ and $\mathcal{A}, v \models F_2$, where $F_1, F_2 \in F_\Omega$, or
- $F = \exists x.G$ and $\mathcal{A}, v[x \mapsto e] \models G$ for some $e \in \mathsf{S}_{\mathcal{A}}$, where $\mathsf{S} = sort(x)$, or
- $F = \forall x.G$ and $\mathcal{A}, v[x \mapsto e] \models G$ for every $e \in \mathsf{S}_{\mathcal{A}}$, where $\mathsf{S} = sort(x)$.

A formula $F$ is called:

- ***satisfiable by*** $\mathcal{A}$ ***under*** $v$ (or ***valid in*** $\mathcal{A}$ ***under*** $v$) if $\mathcal{A}, v \models F$; in this case, $F$ is also called ***consistent***;
- ***satisfiable by*** $\mathcal{A}$ if $\mathcal{A}, v \models F$ for some assignment $v$;
- ***valid in*** $\mathcal{A}$, written $\mathcal{A} \models F$, if $\mathcal{A}, v \models F$ for any assignment $v$; in this case, $\mathcal{A}$ is called a ***model*** of $F$;
- ***satisfiable*** if $\mathcal{A}, v \models F$ for some algebra $\mathcal{A}$ and some assignment $v$;
- ***unsatisfiable*** if $\mathcal{A}, v \not\models F$ for any algebra $\mathcal{A}$ and any assignment $v$; in this case, $F$ is also called ***inconsistent***; we write $\perp$ to denote a ***contradiction***, an unsatisfiable formula like $F \wedge \neg F$ or an empty disjunction.
- ***valid***, written $\models F$, if $\mathcal{A}, v \models F$ for any algebra $\mathcal{A}$ and any assignment $v$; in this case, $F$ is also called ***tautology***.

Note, that if $F$ is a sentence, that is a formula not containing a free variable, it is valid in $\mathcal{A}$ if and only if it is satisfiable by $\mathcal{A}$.

Given two formulae $F$ and $G$, we say $F$ ***entails*** $G$, or $G$ is a ***consequence*** of $F$, written $F \models G$, if for any algebra $\mathcal{A}$ and assignment $v$, such that $\mathcal{A}, v \models F$, it holds that $\mathcal{A}, v \models G$. The formulae $F$ and $G$ are called ***equivalent***, written $F \models\!\!\!\!\!= G$, if $F \models G$ and $G \models F$. We call two formulae $F$ and $G$ ***equisatisfiable***, if $F$ is satisfiable iff $G$ is satisfiable (not necessarily in the same models). Note that if $F$ and $G$ are equivalent then they are equisatisfiable, but not other way round. The notions of “entailment”, “equivalence”, and “equisatisfiability” are naturally extended to sets of formulae, that are treated as conjunctions of single formulae. Thus, given formula sets $N$ and $M$, we say that $N$ entails $M$, written $N \models M$, if for any algebra $\mathcal{A}$ and assignment $v$, such that $\mathcal{A}, v \models F$ for every $F \in N$, it follows that $\mathcal{A}, v \models G$ for every $G \in M$. The sets $N$ and $M$ are equivalent, written $N \models\!\!\!| M$, if $N \models M$ and $M \models N$. Given an arbitrary formula $F$ and formula set $N$, we write $N \models F$ to denote $N \models \{F\}$; analogously, $F \models N$ stands for $\{F\} \models N$.

Since clauses are implicitly universally quantified disjunctions of quantifier-free literals, a clause $C$ is satisfiable by an algebra $\mathcal{A}$, if for the algebra $\mathcal{A}$ and every assignment $v$ there is a literal $L$ in $C$, which is satisfied by $\mathcal{A}$ under $v$. More formally, a clause $C$ defined as

$$\begin{aligned} C &= A_1, \dots, A_m \rightarrow B_1, \dots, B_n \\ &= \neg A_1 \vee \dots \vee \neg A_m \vee B_1 \vee \dots \vee B_n, \end{aligned}$$

is satisfiable by an algebra $\mathcal{A}$, written $\mathcal{A} \models C$, if and only if for any assignment $v$ it holds that $\mathcal{A}, v \not\models A_i$ or $\mathcal{A}, v \models B_j$, for some $i \in \{1, \dots, m\}$ or $j \in \{1, \dots, n\}$. Note, that if $C = \{L_1, \dots, L_k\}$ is a ground clause, where every $L_i$ is a ground literal, then $\mathcal{A} \models C$ if and only if there is a literal $L_j$ in $C$ such that $\mathcal{A} \models L_j$. A clause set $N$ is satisfiable iff all clauses $C \in N$ are satisfiable by the same algebra $\mathcal{A}$. Accordingly, if $N$ and $M$ are two clause sets, $N \models M$ iff every model $\mathcal{A}$ of $N$ is also a model of $M$.

◄ DEFINITION 2.27
Herbrand Interpretation

A ***Herbrand interpretation*** $\mathcal{I}$ is a $\Sigma$-algebra, whose universe $U_{\mathcal{I}}$ equals a quotient $T_\Omega/\!\sim$, for some equivalence relation $\sim$, and which interprets every ground term over $\Sigma$ by its equivalence class:

$$\mathcal{I} \text{ Herbrand} \overset{\text{def}}{\iff} \begin{array}{l} \text{(i) } U_{\mathcal{I}} = T_\Omega/\!\sim, \text{ and} \\ \text{(ii) } f_{\mathcal{I}}([t_1]_\sim, \dots, [t_n]_\sim) = [f(t_1, \dots, t_n)]_\sim, \text{ for all } f \in \Omega. \end{array}$$

■

Since a Herbrand interpretation $\mathcal{I}$ is uniquely defined by its universe $U_{\mathcal{I}} = T_\Omega/\!\sim$, we will often write $T_\Omega/\!\sim$ instead of $\mathcal{I}$. If the equivalence $\sim$ equals $\sim_E$, the smallest congruence containing a set of equations $E$, we simply write $T_\Omega/E$ for the quotient $T_\Omega/\!\sim_E$. Evidently, any Herbrand interpretation is term-generated.

The following theorem by Birkhoff states the equivalence between syntactic and semantic consequences.

◄ THEOREM 2.28
Birkhoff's Theorem

*Let $\mathcal{X}$ be a countably infinite set of variables, $E$ a set of equations. The following properties are equivalent for all terms $l, r \in T_\Omega(\mathcal{X})$:*

*(i)* $l \leftrightarrow_E^* r$,

*(ii)* $E \models \forall \vec{x}.(l \approx r)$,

*(iii)* $T_\Omega(\mathcal{X})/E \models \forall \vec{x}.(l \approx r)$,

*where* $\vec{x} = var(l) \cup var(r)$.

From Birkhoff's Theorem and Theorem 2.19 it follows that a *convergent* term rewrite system $R$ entails an equation $l \approx r$ if and only if the normal forms of the terms with respect to $R$ are equal:

$$R \models \forall \vec{x}.(l \approx r) \;\Leftrightarrow\; l{\downarrow_R} = r{\downarrow_R}\,.$$

DEFINITION 2.29 ► Construction of candidate interpretations [BG94] Rewrite System $R_N$ Candidate Interpretation $\mathcal{I}_N$

Let $\Sigma = (\mathcal{S}, \Omega)$ be a signature with sort and operator sets $\mathcal{S}$ and $\Omega$, respectively, $N$ a set of ground $\Sigma$-clauses and $\succ$ a reduction ordering total on ground $\Sigma$-terms.

We inductively define ground rewrite systems $E_C$ and $R_C$ and Herbrand interpretations $\mathcal{I}_C$ for all clauses $C \in N$. Assume $E_D$ and $R_D$ have been defined for all clauses $D \in N$ for which $D \prec C$, then

$$R_C = \bigcup_{\substack{D \in N \\ D \prec C}} E_D \quad \text{and} \quad \mathcal{I}_C = T_\Omega / R_C,$$

where $T_\Omega / R_C$ is a quotient for $T_\Omega$ by the smallest congruence containing $R_C$. Moreover, if $C = \Gamma \to \Delta, s \approx t$, where:

(i) $s \approx t$ is strictly maximal in $C$,

(ii) $s \succ t$,

(iii) $s$ irreducible by $R_C$, and

(iv) $\mathcal{I}_C \not\models \Gamma \to \Delta$,

then $E_C = \{s \approx t\}$. In this case, $C$ is said to ***produce*** the equation (rewrite rule) $s \approx t$, and called ***productive***. Otherwise $E_C = \emptyset$.

Finally, we define

$$R_N = \bigcup_{C \in N} E_C \quad \text{and} \quad \mathcal{I}_N = T_\Omega / R_N,$$

where $T_\Omega / R_N$ is a quotient for $T_\Omega$ by the smallest congruence containing $R_N$. The Herbrand interpretation $\mathcal{I}_N$ is called a ***candidate interpretation***. ■

The term rewrite system $R_N$ is constructed in such a way that it is left-reduced and terminating, hence convergent. Consequently, if $u \approx v$ is a ground equation over $\Sigma$, then $\mathcal{I}_N$ entails the equation iff the terms $u$ and $v$ are joinable or, equivalently, have the same normal form in $R_N$:

$$\mathcal{I}_N \models u \approx v \quad \Leftrightarrow \quad u \downarrow_{R_N} v \quad \Leftrightarrow \quad u\!\downarrow_{R_N} = v\!\downarrow_{R_N}.$$

The same holds for $R_C$ and $\mathcal{I}_C$.

DEFINITION 2.30 ► Homomorphism Homomorphic Algebras

Let $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ be two signatures such that $\Sigma \subseteq \Sigma'$, that is: $\mathcal{S} \subseteq \mathcal{S}'$, $\Omega \subseteq \Omega'$. Let $\mathcal{A}$ be a $\Sigma$-algebra, and $\mathcal{A}'$ a $\Sigma'$-algebra.

The algebra $\mathcal{A}$ is called ***homomorphic to*** $\mathcal{A}'$, if there exists a mapping $h : U_\mathcal{A} \to U_{\mathcal{A}'}$, such that

(i) $h(\mathsf{S}_\mathcal{A}) \subseteq \mathsf{S}_{\mathcal{A}'}$, for every sort $\mathsf{S} \in \mathcal{S}$; and

(ii) $h(f_\mathcal{A}(e_1, \ldots, e_n)) = f_{\mathcal{A}'}(h(e_1), \ldots, h(e_n))$, for every operator $f : \mathsf{S}_1 \times \cdots \times \mathsf{S}_n \to \mathsf{S}$ in $\Omega$, where $e_i \in (\mathsf{S}_i)_\mathcal{A}$ for every $i \in \{1, \ldots, n\}$.

The mapping $h$ is called a ***homomorphism from*** $\mathcal{A}$ ***to*** $\mathcal{A}'$. ■

DEFINITION 2.31 ► Monomorphism Monomorphic Algebras

Let $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ be two signatures such that $\Sigma \subseteq \Sigma'$, that is: $\mathcal{S} \subseteq \mathcal{S}'$, $\Omega \subseteq \Omega'$. Let $\mathcal{A}$ be a $\Sigma$-algebra, and $\mathcal{A}'$ a $\Sigma'$-algebra.

The algebra $\mathcal{A}$ is called ***monomorphic to*** $\mathcal{A}'$, if there exists a mapping $h : U_\mathcal{A} \to U_{\mathcal{A}'}$, such that

(i) $h$ is a homomorphism from $\mathcal{A}$ to $\mathcal{A}'$; and

(ii) $h$ is injective, that is: $\forall e_1, e_2 \in U_\mathcal{A} : e_1 \neq e_2 \Rightarrow h(e_1) \neq h(e_2)$.

The mapping $h$ is called a ***monomorphism from*** $\mathcal{A}$ ***to*** $\mathcal{A}'$. ■

From the two definitions above it follows that a homomorphism is a structure-preserving mapping, a monomorphism preserves, in addition, distinctness.

If $h$ is a homomorphism from $\mathcal{A}$ to $\mathcal{A}'$, and there exists an inverse mapping $h^{-1} : U_{\mathcal{A}'} \to U_{\mathcal{A}}$ that is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$, then $h^{-1}$ is called an ***inverse homomorphism from*** $\mathcal{A}'$ ***to*** $\mathcal{A}$.

◄ DEFINITION 2.32
Isomorphism
Isomorphic Algebras

Let $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ be two signatures such that $\Sigma \subseteq \Sigma'$, that is: $\mathcal{S} \subseteq \mathcal{S}'$, $\Omega \subseteq \Omega'$. Let $\mathcal{A}$ be a $\Sigma$-algebra, and $\mathcal{A}'$ a $\Sigma'$-algebra.

A homomorphism $h$ from $\mathcal{A}$ to $\mathcal{A}'$ is called ***isomorphism*** if there exists an inverse homomorphism $h^{-1}$ from $\mathcal{A}'$ to $\mathcal{A}$. Algebras $\mathcal{A}$ and $\mathcal{A}'$ are called ***isomorphic*** if there exists an isomorphism from one to another. ■

In other words, $\mathcal{A}$ and $\mathcal{A}'$ are isomorphic if there exists a bijective mapping $h : U_{\mathcal{A}} \to U_{\mathcal{A}'}$, such that $h$ is a homomorphism from $\mathcal{A}$ to $\mathcal{A}'$, and its inverse $h^{-1}$ is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$.

◄ DEFINITION 2.33
Specification

A ***specification*** is a tuple

$$\mathsf{Sp} \overset{\text{def}}{=} (\Sigma, \mathscr{C}),$$

where $\Sigma$ is a signature and $\mathscr{C}$ a non-empty class of term-generated $\Sigma$-algebras. ■

The algebras in $\mathscr{C}$ are called ***models of the specification*** $\mathsf{Sp}$. If $\mathscr{C}$ is the class of all term-generated $\Sigma$-models of a certain set of $\Sigma$-formulae $N$, then we write $(\Sigma, N)$ instead of $(\Sigma, \mathscr{C})$.

The specification $\mathsf{Sp}$ is ***compact*** if every set of formulae over $\Sigma$ is satisfiable in $\mathscr{C}$ whenever each of its finite subsets is satisfiable in $\mathscr{C}$, or equivalently, if every set of formulae over $\Sigma$ is unsatisfiable in $\mathscr{C}$ whenever some of its finite subsets is unsatisfiable in $\mathscr{C}$.

CHAPTER 3

# Hierarchic Refutational Theorem Proving

In this chapter we introduce the hierarchic superposition calculus SUP(T) which enables the hierarchic combination of an abstract theory T with first-order logic. The main goal of the chapter is to show that SUP(T) is a complete calculus. Also, we define and investigate the notion of a local sufficient completeness criterion and prove that SUP(T) is complete on locally sufficiently complete clause sets. The chapter ends with a presentation of two basic effective hierarchic reduction rules.

## 3.1 Refutational Theorem Proving

### 3.1.1 Inference, Reduction, and Splitting Rules

The technical part, "mechanics", of automated deduction is realized by means of special rules, called ***deduction rules***, that manipulate a given formula set $N$ by adding to or deleting formulae from $N$. The rules are described in a fraction-like manner. For a rule $R$, the formulae above the fraction bar are called the ***premises*** of $R$, denoted by $prem(R)$, and the formulae below the bar the ***conclusions*** of $R$, denoted by $concl(R)$. There are three basic types of rules:

– ***inference rules***

$$\mathcal{I}\,\frac{F_1 \dots F_n}{G}$$

that add the conclusion $G$ to the current formulae set $N$ yielding a formula set $N' = N \cup \{G\}$, provided $F_1, \dots, F_n \in N$;

– ***reduction rules***

$$\mathcal{R}\,\frac{F_1 \dots F_n}{G_1 \dots G_m}$$

that replace in $N$ the premises $F_1 \dots F_n$ by the conclusions $G_1 \dots G_m$ yielding a formula set $N' = N \setminus \{F_1, \dots, F_n\} \cup \{G_1, \dots, G_m\}$;

– and ***splitting rules***

$$\mathcal{S}\,\frac{F}{G \;\mid\; G'}$$

that replace the whole set $N$ by disjunction of two sets $N_1 = N \cup \{G\}$ and $N_2 = N \cup \{G'\}$, provided $F \in N$.

Deduction rules are usually constrained by a list of conditions that have to be satisfied in order to apply the rules and help to reduce the search space. An application of an inference, reduction, or splitting rule is called ***inference***, ***reduction***, or ***splitting***, respectively.

Given a formula set $N$, inference rules are aimed to derive new formulae (the conclusion $G$) from already existing ones (the premises $F_1 \dots F_n$) in $N$, whereas reduction rules are devoted to "simplify" $N$ by replacing formulae in it (the premises $F_1 \dots F_n$) by some "simpler" ones (the conclusions $G_1 \dots G_m$). As a special case, if the number of conclusions of a reduction rule is smaller than the number of its premises, the rule can be actually used to delete formulae from $N$, reducing thus the size of $N$. Splitting rules have a dual nature: on one hand, they replace the current formula set $N$ by two sets and extend them by new formulae $G$ and $G'$, respectively, serving this way like an inference rule; on the other hand, the new formulae $G$ and $G'$ are "simpler" then the premise $F$ and usually subsume $F$, so that $F$ can be deleted right after the splitting rule application, which yields formula sets $N_1' = N_1 \setminus \{F\}$ and $N_2' = N_2 \setminus \{F\}$, respectively, reducing thus the original problem to "simpler" subproblems, and serving this way like a reduction rule. The aim of splitting is to split a clause set into two simpler independent clause sets, such that they can be considered separately from each other.

A set of inference rules is called an ***inference system***. A set of deduction rules is called a ***deduction system***. An inference/reduction rule is ***sound*** (with respect to an entailment relation $\models$) if it yields a formula set $N'$ entailed (with respect to $\models$) by the original set $N$, i.e. $N \models N'$. A splitting rule is sound if the disjunction of the resulting formula sets $N_1$ and $N_2$ is entailed by the original set $N$, i.e. $N \models N_1 \vee N_2$. A deduction system is called ***complete*** if every formula $G$ entailed by a given formula set $N$ is derivable from $N$ by a sequence of application of the deduction system's rules. A deduction system is called ***refutationally complete***, if a contradiction $\perp$ is derivable by the deduction system's rules from any unsatisfiable formula set. A contradiction might be any unsatisfiable formula, for instance $P \wedge \neg P$. In this work we stick to the clausal fragment (we consider only formula sets that consist of clauses), so the only contradictory formula in our case is the empty clause $\square$.

Most modern refutational theorem proving techniques exhaustively apply inference rules to a given formula set $N$ until a closed set $N_\infty$, called ***saturated***[1], is reached, for which conclusion of every inference possible on $N_\infty$ is already contained in the set $N_\infty$. And if the inference system used is refutationally complete, the original set $N$ is unsatisfiable if and only if the saturated set $N_\infty$ contains the empty clause $\square$. But in practice saturated sets tend to be very large, very often even infinite (actually, saturated sets are finite only for a few classes of formulae), nonetheless most of formulae in a saturated set are not needed for deriving a contradiction. For this reason, in order to obtain a practically useful result (particularly, to ensure termination) it is critical to develop methods for determining redundant formulae and eliminating inferences with redundant premises or producing such superfluous conclusions, and, therefore, to introduce a weaker notion of saturation.

In the subsequent sections, we formally describe the notion of a "redundancy criterion" — an abstract apparatus aimed to define redundancy for a given class of formulae and inference system. Thereafter, based on the introduced notion of redundancy criterion we give definitions of a derivation relation, saturation, theorem proving calculus, and approximation of theorem proving calculus. In Sections 3.3, 3.4.2, 3.4.4 we instantiate this formalisation for the standard ground superposition calculus and hierarchic superposition calculus; the main purpose of this instantiation is showing refutational completeness of the hierarchic calculus.

### 3.1.2 Redundancy Criterion

A redundancy criterion usually consists of two parts: a redundant formula criterion $\mathcal{R}_F$, and a redundant inference criterion $\mathcal{R}_I$. Given a formula set $N$, the set $\mathcal{R}_F(N)$ contains all formulae, which are redundant for $N$ (for instance, formulae subsumed by ones in $N$, or tautologies), and can be therefore deleted from $N$ (if certain conditions are satisfied). Likewise, the set $\mathcal{R}_I(N)$ contains all redundant inferences, which are not needed to be performed (under certain conditions). Note, that $\mathcal{R}_F(N)$ may also contain formulae which do not appear in $N$, and $\mathcal{R}_I(N)$ may contain inferences with premises beyond $N$. Here we give a definition of an abstract redundancy criterion, which we later instantiate with con-

[1]Here we discuss the topic of saturation informally — for a formal definition of a saturated set see Section 3.1.3

crete criteria for standard ground and hierarchic superposition calculi. The usage of an abstract notion of a redundancy criterion benefits from establishing properties which hold generally for all derivations regardless of a particular choice of an inference system or a redundancy criterion. For a specific criterion, it has to be shown to comply to the abstract notion, so that the general derivation properties could be exploited without proving them to hold in that particular context.

There are only few works which abstractly define the notion of a redundancy criterion [BGW94, BG01, Wal02] differring from each other in minor aspects. Next we give a definition of the abstract notion which better suits the SUP(T) framework (the same definition appears also in [Wal02]).

DEFINITION 3.1 ► Redundancy Criterion

A pair $\mathcal{R} = (\mathcal{R}_I, \mathcal{R}_F)$ 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 $M$:

(i) $N \setminus \mathcal{R}_F(N) \models \mathcal{R}_F(N)$;

(ii) if $N \subseteq M$, then $\mathcal{R}_F(N) \subseteq \mathcal{R}_F(M)$ and $\mathcal{R}_I(N) \subseteq \mathcal{R}_I(M)$;

(iii) if $I \in \mathcal{I}(M)$ and $concl(I) \in N \cup \mathcal{R}_F(N)$, then $I \in \mathcal{R}_I(N)$;

(iv) if $M \subseteq \mathcal{R}_F(N)$, then $\mathcal{R}_F(N) \subseteq \mathcal{R}_F(N \setminus M)$ and $\mathcal{R}_I(N) \subseteq \mathcal{R}_I(N \setminus M)$.

Inferences in $\mathcal{R}_I(N)$ and formulae in $\mathcal{R}_F(N)$ are said to be ***redundant*** for (with respect to) $N$. ■

The conditions of a redundancy criterion $\mathcal{R}$ have the following meaning:

- condition (i) says that, given a formulae set $N$, the formulae redundant for $N$ logically follow from the non-redundant subset $N \setminus \mathcal{R}_F(N)$ of $N$;
- the second condition expresses the monotonicity property of $\mathcal{R}$ with respect to the set inclusion;
- according to condition (iii), all inferences with a conclusion that is redundant for $N$ or already contained in $N$ are redundant;
- and the last condition states that the formulae/inferences redundant for $N$ remain so if redundant formulae (all or some of them) are deleted from $N$.

Very often we shall write just $\mathcal{R}$ instead of $\mathcal{R}_F$ and $\mathcal{R}_I$, whenever the matter of the context's discussion can be applied for both $\mathcal{R}_F$ and $\mathcal{R}_I$; for instance, the monotonicity property of $\mathcal{R}$ (condition (ii) in Definition 3.1) can be alternatively expressed as '$N \subseteq M$ implies $\mathcal{R}(N) \subseteq \mathcal{R}(M)$'.

Note that, conditions (i) and (iv) of Definition 3.1 imply that if $M \subseteq \mathcal{R}_F(N)$, then $\mathcal{R}(N) = \mathcal{R}(N \setminus M)$, for any formula sets $M$ and $N$. Indeed, $N \setminus M \subseteq N$, hence, by condition (ii), $\mathcal{R}(N \setminus M) \subseteq \mathcal{R}(N)$, which together with condition (iv) gives $\mathcal{R}(N) = \mathcal{R}(N \setminus M)$.

## 3.1.3 Derivations. Saturation and Refutation

DEFINITION 3.2 ► Saturated Set

A set of formulae $N$ is called ***saturated*** (with respect to an inference system $\mathcal{I}$ and reduction criterion $\mathcal{R}$), if all inferences with non-redundant premises from $N$ are redundant:

$$N \textit{ saturated} \overset{\text{def}}{\iff} \mathcal{I}(N \setminus \mathcal{R}_F(N)) \subseteq \mathcal{R}_I(N).$$

■

Alternatively, the set $N$ can be called $(\mathcal{I},\mathcal{R})$-*saturated*, or just *saturated*, if $\mathcal{I}$ and $\mathcal{R}$ are clear from the context.

An inference system $\mathcal{I}$, a redundancy criterion $\mathcal{R}$, and a consequence relation $\models$ define a derivation relation $\vdash$ which determines all possible formula set derivable from a given formulae set $N$.

◄ DEFINITION 3.3 Derivation Relation

A binary relation $\vdash$ on sets of formulae is ***derivation relation*** (with respect to an inference system $\mathcal{I}$, redundancy criterion $\mathcal{R}$, and entailment relation $\models$), if the following properties are satisfied for all sets of formulae $N$ and $M$:

(i) if $I \in \mathcal{I}(N)$, then $N \vdash N \cup \{concl(I)\}$.

(ii) if $N \vdash M$, then $N \setminus M \subseteq \mathcal{R}_F(M)$;

(iii) if $N \vdash M$, then $N \models M$;

■

Condition (i) enables adding to the formula set $N$ the conclusion of any inference $I$ with premises from $N$, whereas condition (ii) enables deleting from $N$ formulae which are redundant for $N$. Condition (iii) reflects soundness of the two above actions.

Next we prove that if $M$ is derived from $N$, then $M$ is equivalent to $N$. First we show that $N$ is a logical consequence of $M$.

◄ LEMMA 3.4

*Let $N$ and $M$ be arbitrary formulae sets, then $N \vdash M$ implies $M \models N$.*

◄ PROOF

Assume $N \vdash M$. We split the set $N$ into two parts $N_1$ and $N_2$ defined as follows:

$$\begin{aligned} N_1 &= N \cap M, \\ N_2 &= N \setminus M. \end{aligned}$$

Evidently, $N = N_1 \cup N_2$. We are to show that $M \models N_1$ and $M \models N_2$. Obviously, $N_1 \subseteq M$, hence $M \models N_1$. On other hand,

$$\begin{array}{rrcll} & N \setminus M & \subseteq & \mathcal{R}_F(M) & \text{// by cond. (ii) of Def. 3.3} \\ \Rightarrow & \mathcal{R}_F(M) & \models & N_2 & \text{// as } N_2 = N \setminus M, \\ \Rightarrow & M \setminus \mathcal{R}_F(M) & \models & N_2 & \text{// as } M \setminus \mathcal{R}_F(M) \models \mathcal{R}_F(M), \\ & & & & \quad \text{by cond. (i) of Def. 3.1} \\ \Rightarrow & M & \models & N_2 & \end{array}$$

Thus, $M \models N_1$ and $M \models N_2$, hence $M \models N_1 \cup N_2 = N$. ■

◄ LEMMA 3.5

*Let $N$ and $M$ be arbitrary formulae sets, then $N \vdash M$ implies $M \models\!\!\dashv N$.*

◄ PROOF

By condition (iii) of Definition 3.3 and Lemma 3.4. ■

◄ DEFINITION 3.6 Derivation

A (finite or countably infinite) sequence

$$N_0 \vdash N_1 \vdash N_2 \vdash \ldots$$

is called an $(\mathcal{I},\mathcal{R})$-***derivation*** (or simply ***derivation***, if $\mathcal{I}$ and $\mathcal{R}$ are clear from the context). ■

DEFINITION 3.7 ► Derivation Closure, Derivation Limit, Persisting formulae

Let $N_0 \vdash N_1 \vdash \ldots$ be a derivation. We write $N^*$ to denote the set of all derived formulae in the derivation:

$$N^* \overset{\text{def}}{=} \bigcup_i N_i$$

and call it the ***closure*** of the derivation. We write $N_\infty$ to denote the set of all ***persisting*** formulae in the derivation:

$$N_\infty \overset{\text{def}}{=} \bigcup_i \bigcap_{j \geq i} N_j$$

and call it the ***limit*** of the derivation. ■

DEFINITION 3.8 ► Fair Derivation

A ***derivation*** $N_0 \vdash N_1 \vdash \ldots$ is called ***fair***, if every inference with non-redundant premises from the limit of the derivation is redundant with respect to some $N_i$:

$$N_0 \vdash N_1 \vdash \ldots \textit{ fair} \overset{\text{def}}{\iff} \mathcal{I}(N_\infty \setminus \mathcal{R}_F(N_\infty)) \subseteq \bigcup_i \mathcal{R}_I(N_i).$$

■

The main feature of a fair derivation is that its limit is always saturated. Next, we formally prove this property, as in the literature there are different definitions of a fair derivation, and the combination of definitions of a redundancy criterion, a fair derivation and a saturated set presented here does not appear anywhere else including the original work by Bachmair, Ganzinger, and Waldmann on hierarchic superposition [BGW94].

LEMMA 3.9 ► *Let $N_0 \vdash N_1 \vdash \ldots$ be a derivation. Then:*

*(i) $\mathcal{R}(N^*) = \mathcal{R}(N_\infty)$;*

*(ii) if $N_0 \vdash N_1 \vdash \ldots$ is fair, then $N_\infty$ is saturated.*

PROOF ► Statement (i). On one hand, $N_\infty \subseteq N^*$, therefore $\mathcal{R}(N_\infty) \subseteq \mathcal{R}(N^*)$, by Condition (ii) of Definition 3.1. On the other hand, the set $N^* \setminus N_\infty$ consists of all non-persisting formulae in $N^*$, i.e. for every $C \in N^* \setminus N_\infty$, there exists an $i \geq 0$ such that $C \in N_i \setminus N_{i+1}$. Consider the set $N_i \setminus N_{i+1}$, for an arbitrary $i \geq 0$:

$$\begin{array}{rrcll}
 & N_i \setminus N_{i+1} & \subseteq & \mathcal{R}_F(N_{i+1}) & \text{// as } N_i \vdash N_{i+1} \\
 & & & & \quad \text{and by cond. (ii) of Def. 3.3} \\
 & & \subseteq & \mathcal{R}_F(N^*) & \text{// as } N_{i+1} \subseteq N^*, \\
 & & & & \quad \text{and by cond. (ii) of Def. 3.1} \\
\Rightarrow & N^* \setminus N_\infty & \subseteq & \mathcal{R}_F(N^*) & \\
\Rightarrow & \mathcal{R}(N^*) & \subseteq & \mathcal{R}(N^* \setminus (N^* \setminus N_\infty)) & \text{// by cond. (iv) of Def. 3.1} \\
 & & = & \mathcal{R}(N_\infty) & \text{// as } N_\infty \subseteq N^*
\end{array}$$

Thus, $\mathcal{R}(N_\infty) \subseteq \mathcal{R}(N^*)$ and $\mathcal{R}(N^*) \subseteq \mathcal{R}(N_\infty)$, consequently $\mathcal{R}(N_\infty) = \mathcal{R}(N^*)$.

Statement (ii). Let $I \in \mathcal{I}(N_\infty \setminus \mathcal{R}_F(N_\infty))$ be an arbitrary inference with non-redundant premises from the limit of the derivation, then

$$\begin{array}{rrcll}
 & I & \in & \mathcal{R}_I(N_i) & \text{// by Def. 3.8,} \\
 & & & & \quad \text{as } N_0 \vdash N_1 \vdash \ldots \text{ fair,} \\
 & & & & \quad \text{for some } i \geq 0 \\
 & & \subseteq & \mathcal{R}_I(N^*) & \text{// as } N_i \subseteq N^*, \\
 & & & & \quad \text{and by cond. (ii) of Def. 3.1} \\
 & & \subseteq & \mathcal{R}_I(N_\infty) & \text{// by (i) of Lemma 3.9} \\
\Rightarrow & \mathcal{I}(N_\infty \setminus \mathcal{R}_F(N_\infty)) & \subseteq & \mathcal{R}_I(N_\infty) &
\end{array}$$

From Definition 3.2, we conclude that $N_\infty$ is saturated. ■

◂ LEMMA 3.10

*Let $\vdash_1$ and $\vdash_2$ be two derivation relations defined with respect to the same entailment relation $\models$, inference systems $\mathcal{I}$ and $\mathcal{I}'$, and redundancy criteria $\mathcal{R}$ and $\mathcal{R}'$, respectively. If $N \vdash_1 N_1 \vdash_1 N_2 \vdash_1 \ldots$ and $N \vdash_2 N_1' \vdash_2 N_2' \vdash_2 \ldots$ are two derivations, and $N_\infty$ and $N_\infty'$ are their respective limits, then $N_\infty \models\!\mid N_\infty'$.*

◂ PROOF

By soundness of a derivation relation (condition (iii) of Definition 3.3) we know $N \models N_\infty$. On other hand,

$$\begin{aligned} N \setminus N_\infty &\subseteq N^* \setminus N_\infty && \text{// as in the proof of Lemma 3.9(i)} \\ &\subseteq \mathcal{R}_F(N^*) && \text{// as in the proof of Lemma 3.9(i)} \\ &= \mathcal{R}_F(N_\infty) && \text{// by Lemma 3.9(i)} \\ \Rightarrow \quad N_\infty &\models N \end{aligned}$$

Therefore, $N_\infty \models\!\mid N$. Analogously, we obtain $N_\infty' \models\!\mid N$. Consequently, $N_\infty \models\!\mid N_\infty'$. ■

Note that from Lemma 3.5 it inductively follows that for two derivations $N \vdash_1 N_1 \vdash_1 N_2 \vdash_1 \ldots$ and $N \vdash_2 N_1' \vdash_2 N_2' \vdash_2 \ldots$, where $\vdash_1$ and $\vdash_2$ are as defined in Lemma 3.10, it also holds that $N_i \models\!\mid N_j'$, for any two formula sets $N_i$ and $N_j'$ from the respective derivations, $i, j \geq 1$.

◂ DEFINITION 3.11
Theorem Proving Calculus

A triple $(\mathcal{I}, \mathcal{R}, \vdash)$ consisting of an inference system $\mathcal{I}$, a redundancy criterion $\mathcal{R}$, and a derivation relation $\vdash$ is called a ***theorem proving calculus***, or simply ***calculus***. We often write $(\mathcal{I}, \mathcal{R})$ instead of $(\mathcal{I}, \mathcal{R}, \vdash)$ if the matter of discussion does not depend on the particular choice of a derivation relation. ■

To any theorem proving calculus $(\mathcal{I}, \mathcal{R})$, there always correspond (i) a class $\mathcal{N}$ of formula sets, saturating which the calculus is aimed at, and (ii) the underlying entailment relation $\models$, with respect to which the redundancy criterion $\mathcal{R}$ is defined and regarding to which the inference system $\mathcal{I}$ is sound.

◂ DEFINITION 3.12
Refutational Completeness

A theorem proving calculus $(\mathcal{I}, \mathcal{R})$ is called ***refutationally complete*** (for the underlying formula set class $\mathcal{N}$), if for every saturated formula set $N \in \mathcal{N}$ it holds, that if $N$ is unsatisfiable (with respect to the underlying entailment relation $\models$) then the contradiction $\square$ is contained in $N$. ■

### 3.1.4 Approximation of Theorem Proving Calculi

To show refutational completeness of hierarchic superposition calculus SUP(T) for clauses over a combination of first-order logic with a background theory T, we shall basically follow a schema exploited to obtain a completeness result for the case of the standard superposition for *general first-order* clauses, namely – by lifting the completeness result of the standard superposition for *ground* clauses. To this end we introduce a concept of approximation between theorem proving calculi [BGW94] – an abstract technique that allows to establish the completeness property for a given calculus by relating it to another calculus, that is a-priori known to be refutationally complete.

DEFINITION 3.13 ► Calculi Approximation

Let $(\mathcal{I}_1, \mathcal{R}_1)$ and $(\mathcal{I}_2, \mathcal{R}_2)$ be two theorem proving calculi with respective underlying formula set classes $\mathcal{N}_1$ and $\mathcal{N}_2$ and entailment relations $\models_1$ and $\models_2$. We say that $(\mathcal{I}_1, \mathcal{R}_1)$ ***approximates*** $(\mathcal{I}_2, \mathcal{R}_2)$ via a mapping $\alpha$ from $\mathcal{N}_1$ to $\mathcal{N}_2$, if the following conditions are satisfied for every formula set $N \in \mathcal{N}_1$:

(i) if $N$ is $(\mathcal{I}_1, \mathcal{R}_1)$-saturated, then $\alpha(N)$ is $(\mathcal{I}_2, \mathcal{R}_2)$-saturated;

(ii) if $N \models_1 \perp_1$, then $\alpha(N) \models_2 \perp_2$;

(iii) if $\perp_2 \in \alpha(N)$, then $\perp_1 \in N$,

where the symbol $\perp_i$ denotes a contradiction for $\models_i$. ■

THEOREM 3.14 ► Calculi Approximation Theorem

*Let $(\mathcal{I}_1, \mathcal{R}_1)$ and $(\mathcal{I}_2, \mathcal{R}_2)$ be two theorem proving calculi with respective underlying formula set classes $\mathcal{N}_1$ and $\mathcal{N}_2$ and entailment relations $\models_1$ and $\models_2$. If $(\mathcal{I}_1, \mathcal{R}_1)$ approximates $(\mathcal{I}_2, \mathcal{R}_2)$ via $\alpha$, and if $(\mathcal{I}_2, \mathcal{R}_2)$ is refutationally complete for $\mathcal{N}_2$, then $(\mathcal{I}_1, \mathcal{R}_1)$ is refutationally complete for $\mathcal{N}_1$.*

PROOF ► Let $\perp_1$ and $\perp_2$ denote contradictions for $\models_1$ and $\models_2$, respectively. Assume $N \in \mathcal{N}_1$, $N \models_1 \perp_1$, and $N$ is saturated with respect to $(\mathcal{I}_1, \mathcal{R}_1)$. Then, by condition (i) of Definition 3.13, $\alpha(N)$ is saturated with respect to $(\mathcal{I}_2, \mathcal{R}_2)$; moreover, by condition (ii) of the definition, $\alpha(N) \models_2 \perp_2$. By the assumption, $(\mathcal{I}_2, \mathcal{R}_2)$ is refutationally complete for $\mathcal{N}_2$, hence $\perp_2 \in \alpha(N)$. Consequently, by condition (iii), we have $\perp_1 \in N$. As $N$ has been picked arbitrarily from $\mathcal{N}_1$, we conclude $(\mathcal{I}_1, \mathcal{R}_1)$ is refutationally complete for $\mathcal{N}_1$. ■

For many readers, it may be more intuitive to consider the approximation technique in the contrapositive way. By definition, a refutationally complete calculus has the property that a saturated formula set is unsatisfiable if and only if it contains a contradictory formula. Equivalently, a saturated set is satisfiable if and only if it does not contain a contradictory formula. So, in practice (for instance, in the case of standard superposition for general FOL clauses) the model existence of the saturated (clause) set not containing a contradictory formula (an empty clause) is a kind of argument exploited for showing refutational completeness. To this end, the conditions in definition of calculi approximation can be equivalently reformulated as follows: the first condition of Definition 3.13 remains the same; the second becomes 'if $\alpha(N)$ is satisfiable with respect to $\models_2$, then $N$ is satisfiable with respect to $\models_1$'; and the third condition becomes 'if $\perp_1 \notin N$, then $\perp_2 \notin \alpha(N)$'. The last two conditions are contrapositive reformulations of the original ones. Under such formulation, refutational completeness of the approximating calculus $(\mathcal{I}_1, \mathcal{R}_1)$ can be argued as follows. Assume the approximated calculus $(\mathcal{I}_2, \mathcal{R}_2)$ is refutationally complete, and $N$ an arbitrary clause set from $\mathcal{N}_1$, which is saturated with respect to $(\mathcal{I}_1, \mathcal{R}_1)$ and does not contain a contradictory formula $\perp_1$, then:

1. by the first condition, from the saturation of $N$ it follows that $\alpha(N)$ is saturated with respect to $(\mathcal{I}_2, \mathcal{R}_2)$;

2. $\perp_1 \notin N$ implies by the third condition that $\perp_2 \notin \alpha(N)$;

3. from the assumption that $(\mathcal{I}_2, \mathcal{R}_2)$ is refutationally complete, and the facts that $\alpha(N)$ is saturated and does not contain a contradictory formula $\perp_2$, it follows that $\alpha(N)$ is satisfiable with respect to $\models_2$;

4. according to the second condition, $N$ satisfiable with respect to $\models_1$.

Thus, any $(\mathcal{I}_1, \mathcal{R}_1)$-saturated set $N$ not containing a contradictory formula is satisfiable, hence $(\mathcal{I}_1, \mathcal{R}_1)$ is refutationally complete.

For proving refutational completeness of the hierarchic superposition SUP(T) we basically stick to the formulation of calculi approximation given in Definition 3.13; nevertheless, we also informally discuss the completeness property of SUP(T) from the point of view of the more intuitive contrapositive formulation of the calculi approximation concept.

## 3.2 Hierarchic Specification

DEFINITION 3.15 ► Let $\mathsf{Sp} = (\Sigma, \mathscr{C})$ be a specification consisting of a signature $\Sigma = (\mathcal{S}, \Omega)$ with sort set $\mathcal{S}$ and operator set $\Omega$, and a class $\mathscr{C}$ of term-generated algebras closed under isomorphism. Let $\mathsf{Sp}' = (\Sigma', Ax')$ be a specification, where

Hierarchic specification
Base specification
Body
Enrichment

- $\Sigma' = (\mathcal{S}', \Omega')$ is a signature augmenting $\Sigma$, that is:
  - $\mathcal{S} \subseteq \mathcal{S}'$,
  - $\Omega \subseteq \Omega'$;
- $Ax'$ is an axiom (formula) set over $\Sigma'$.

A pair $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is called a ***hierarchic specification***, $\mathsf{Sp}$ is called a ***base specification***, and $\mathsf{Sp}'$ is called the ***body*** of the hierarchic specification. The triple

$$(\mathcal{S}'', \Omega'', Ax'),$$

consisting of

- sorts $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$,
- operator symbols $\Omega'' = \Omega' \setminus \Omega$,
- the set of axioms $Ax'$,

is called the ***enrichment*** of the hierarchic specification $\mathsf{HSp}$.

The signatures $\Sigma$ and $\Sigma'$ come with the underlying disjoint variable sets $\mathcal{X}$ and $\mathcal{X}'$, respectively, such that $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$, where every variable in $\mathcal{X}$ is of a sort $\mathsf{S} \in \mathcal{S}$, and every variable in $\mathcal{X}''$ is of a sort $\mathsf{S}'' \in \mathcal{S}''$; thus, the variable sets $\mathcal{X}$ and $\mathcal{X}''$ are disjoint. For every sort $\mathsf{S} \in \mathcal{S}$ there exists a countably infinite subset of $\mathcal{X}$ consisting of variables of the sort $\mathsf{S}$; for every sort $\mathsf{S}'' \in \mathcal{S}''$ there also exists a countably infinite subset of $\mathcal{X}''$ consisting of variables of the sort $\mathsf{S}''$. ■

In this work we stipulate the enrichment's axiom set $Ax'$ to consist of first-order clauses. We call sorts in $\mathcal{S}$ and $\mathcal{S}''$ (operators in $\Omega$ and $\Omega''$) ***base*** and ***free sorts*** (***operators***), respectively. We call variables in $\mathcal{X}$ and $\mathcal{X}''$ ***base*** and ***non-base variables***, respectively.

From now on we *always* assume a hierarchic specification $\mathsf{HSp}$ consisting of the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and body $\mathsf{Sp}' = (\Sigma', Ax')$, and write

- $\Sigma$, $\mathscr{C}$, $\mathcal{S}$, $\Omega$, $\mathcal{X}$ to denote the base signature, the class of base term-generated algebras closed under isomorphism, the set of all base sorts, the set of all base operator symbols, and the set of all base variables (these are all variables of the base sorts $\mathcal{S}$), respectively;
- $\mathcal{S}''$, $\Omega''$, $\mathcal{X}''$ to denote the set of all free sorts, the set of all free operator symbols, and the set of all non-base variables (these are all variables of the free sorts $\mathcal{S}''$), respectively;
- $\Sigma'$, $\mathcal{S}'$, $\Omega'$, $\mathcal{X}'$ to denote the body signature, the set of all body sorts, the set of all body operator symbols, and the set of all body variables (these are all variables of the base or the free sorts), respectively;
- $\mathcal{A}$ and $\mathcal{A}'$ to denote $\Sigma$- and $\Sigma'$-algebras, respectively.
- $\mathscr{A}_{\Sigma}$ and $\mathscr{A}_{\Sigma'}$ to denote the classes of all $\Sigma$- and $\Sigma'$-algebras, respectively.

## 3.2.1 Syntax

We distinguish between different types of terms according to the sets of symbols they are built on.

**Types of Terms**

◄ DEFINITION 3.16
Pure Term
Base Term
Free Term
Non-Base Term

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

A $\Sigma'$-term $t$ is called ***pure*** if all operator symbols occurring in it exclusively come either from $\Omega$, or $\Omega''$:

$$t \text{ pure} \overset{\text{def}}{\iff} t \in T_\Omega(\mathcal{X}') \cup T_{\Omega''}(\mathcal{X}')$$

Pure terms built over base variables and base operators are called ***base*** terms:

$$t \text{ base} \overset{\text{def}}{\iff} t \in T_\Omega(\mathcal{X})$$

Pure terms built over variables (base or non-base) and free operators, and which are not single base variables, are called ***free***:

$$t \text{ free} \overset{\text{def}}{\iff} t \in T_{\Omega''}(\mathcal{X}') \setminus \mathcal{X}$$

A $\Sigma'$-term that is not a base term is called ***non-base***:

$$t \text{ non-base} \overset{\text{def}}{\iff} t \in T_{\Omega'}(\mathcal{X}') \setminus T_\Omega(\mathcal{X})$$

■

The definitions of pure, base, free, and non-base terms are naturally extended to all expressions, including (dis)equations, atoms, literals, and clauses. According to the given definition, base expressions consist only of base operators $\Omega$ and base variables $\mathcal{X}$, however free terms, in contrast, may contain free and base variables, but if a free term contains a base variable then it must also contain a free symbol different from the equality $\approx$.

Among non-base terms there are special terms which have a free top function symbol ranging into a base sort. We call such terms *extension terms* as they extend the base sorts with non-base terms.

◄ DEFINITION 3.17
Extension Symbols
Extension Terms
Extension-Free Terms
Smooth Terms

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

A free function symbol $f \in \Omega''$ is called an ***extension symbol*** if it ranges into a base sort $\mathsf{S} \in \mathcal{S}$:

$$f \text{ extension symbol} \overset{\text{def}}{\iff} f \in \Omega'', sort(f) \in \mathcal{S}$$

A $\Sigma'$-term $t$ is called an ***extension term*** if its top symbol $top(t)$ is an extension symbol. The set of all extension terms over $\Sigma'$ is denoted by $T^E_{\Omega'}(\mathcal{X}')$:

$$T^E_{\Omega'}(\mathcal{X}') \overset{\text{def}}{=} \{t \in T_{\Omega'}(\mathcal{S}, \mathcal{X}') \mid top(t) \in \Omega''\}$$

A $\Sigma'$-term $t$ is called ***extension-free*** if no subterm of $t$ is an extension term:

$$t \textit{ extension-free} \overset{\text{def}}{\iff} \forall p \in \rho(t) : t/p \notin T^E_{\Omega'}(\mathcal{X}')$$

A $\Sigma'$-term $t$ is called ***smooth*** if no strict subterm of $t$ is an extension term:

$$t \textit{ smooth} \overset{\text{def}}{\iff} \forall p \in \rho(t) : p > \varepsilon \Rightarrow t/p \notin T^E_{\Omega'}(\mathcal{X}')$$

■

Thus, a smooth term $t$ may contain an extension symbol $f$ only if $t$ is an extension term, and hence the extension symbol is the top one $f = \top(t)$. The notion of an extension-free term is naturally extended to all expressions, including (dis)equations, atoms, literals, and clauses.

EXAMPLE 3.18 ► Let $\Sigma = (\mathcal{S}, \Omega)$ be the signature of linear arithmetic over rationals, with operator set $\Omega = \{+,-,\cdot,\leq,<,\ldots\} \cup \mathbb{Q}$, where $\mathbb{Q}$ is the set of all rational numbers, and a single sort $\mathcal{S} = \{\mathsf{S}\}$. Let $\Sigma' = (\mathcal{S}', \Omega')$ be a body signature with sorts $\mathcal{S}' = \mathcal{S} \cup \{\mathsf{S}''\}$ and operators $\Omega' = \Omega \cup \{a, b, f, g, h\}$, where $a, b : \mathsf{S}''$, $f : \mathsf{S}'' \times \mathsf{S} \to \mathsf{S}$, $g : \mathsf{S}'' \to \mathsf{S}$, and $h : \mathsf{S}'' \to \mathsf{S}''$. Let $\mathcal{X} = \{x, y, \ldots\}$ and $\mathcal{X}'' = \{z, u, \ldots\}$ be base and non-base variable sets, respectively. Then function symbols $f$ and $g$ are *extension symbols*, and the following expressions are:

- *base*: terms $x$ and $1$, atoms $x \approx y$ and $3{\cdot}x{+}2{\cdot}y < 1$, clause $(x \leq 0, -2{\cdot}x < 3 \to)$;
- *free*: terms $z$ and $g(a)$, literals $z \approx u$ and $f(a,x) \not\approx x$, clause $a \approx b \to f(a,x) \approx g(b)$;
- *non-base*: term $x + 2 \cdot f(a, 3 \cdot y)$, atom $g(a) \leq 2 \cdot g(b)$, and any free expression;
- *extension-free*: $a$, $b$, $x$, $z$, $h(b)$, $h(z)$, and any base expression;
- *extension terms*: $f(a,0)$, $f(z, 1 + 2 \cdot g(a))$, $g(a)$, and $g(b)$;
- *smooth terms*: $a$, $x$, $z$, $h(b)$, $f(h(a),0)$, $f(z, 1 + 2 \cdot x)$;
- *smooth extension terms*: $f(a,0)$, $f(z, 1 + 2 \cdot x)$, $g(a)$, and $g(b)$.

■

Any non-base term of a base sort contains an extension subterm. When discussing completeness of the hierarchic superposition calculus SUP(T), it will be of a particular importance to ensure that all non-base terms of base sorts can be reduced to base terms in models of a given clause set $N$ (so-called "*sufficient completeness criterion*", Section 3.4.7). The following lemma shows, that it is sufficient to ensure the reduction property only for smooth extension terms.

LEMMA 3.19 ► Extension Terms Lemma

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ *and* $\Omega'' = \Omega' \setminus \Omega$ *be the enrichment sorts and operators, respectively.*

*Let* $\mathcal{A}'$ *be an arbitrary* $\Sigma'$*-algebra. Then each ground non-base term* $t$ *of a base sort* $\mathsf{S} \in \mathcal{S}$ *is equal under* $\mathcal{A}'$ *to some ground base term* $s$*, if and only if each ground smooth extension term* $t'$ *is equal under* $\mathcal{A}'$ *to some ground base term* $s'$*:*

$$\forall t \in T_{\Omega'}(\mathcal{S}) \setminus T_{\Omega}.\exists s \in T_{\Omega} : \mathcal{A}'(t) = \mathcal{A}'(s)$$

$$\Leftrightarrow$$

$$\forall t' \in T^E_{\Omega'}.\exists s' \in T_\Omega : t' \textit{ smooth} \Rightarrow \mathcal{A}'(t') = \mathcal{A}'(s').$$

◄ PROOF

**The "⇒" direction** is straightforward as $T^E_{\Omega'} \subseteq T_{\Omega'}(\mathcal{S}) \setminus T_\Omega$.
**The "⇐" direction.** Let $t \in T_{\Omega'}(\mathcal{S}) \setminus T_\Omega$ be an arbitrary ground non-base term of a base sort. We proceed by induction on the number $occ_E(t)$ of occurrences of extension subterms in $t$, formally defined as follows:

$$occ_E(t) \overset{\text{def}}{=} |\{p \in \rho(t) \mid t/p \in T^E_{\Omega'}\}|.$$

*Induction base.* Assume $occ_E(t) = 1$. Let $t'$ be the extension subterm in $t$. Without loss of generality, $t' = t/p$ occurs in $t$ at a position $p \in \rho(t)$. Any base-sorted symbol, occurring in $t$ at a position $q \in \rho(t)$ above or parallel to $p$, if any, is base:

$$\forall q \in \rho(t) : q \parallel p \text{ or } q < p \Rightarrow top(t/q) \in \Omega.$$

The term $t'$ is smooth and, therefore, $\mathcal{A}'(t') = \mathcal{A}'(s')$ for some ground base term $s' \in T_\Omega$. The term $s = t[s']_p$, obtained from $t$ by replacing $t'$ with $s'$, contains only base symbols, hence it is a base term. Thus, $\mathcal{A}(t) = \mathcal{A}(s)$, where $s \in T_\Omega$.

*Induction hypothesis.* Suppose the assertion holds for all terms $t \in T_{\Omega'}(\mathcal{S}) \setminus T_\Omega$ such that $occ_E(t) \leq n$, for some $n \geq 1$.

*Induction step.* Assume $occ_E(t) = n + 1$. Let $t'$ be a lowest extension subterm appearing in $t$, i.e. $t' = t/p$, for some position $p \in \rho(t)$ such that

$$\forall q \in \rho(t) : q > p \Rightarrow t/q \notin T^E_{\Omega'}.$$

The term $t'$ is smooth and, therefore, $\mathcal{A}'(t') = \mathcal{A}'(s')$ for some ground base term $s' \in T_\Omega$. The term $t[s']_p$ obtained from $t$ by replacing the occurrence of $t'$ at the position $p$ with $s'$ contains by one occurrence of an extension term less than $t$, hence, $occ_E(t[s']_p) = n$, and, by Induction hypothesis, $\mathcal{A}'(t[s']_p) = \mathcal{A}'(s)$, for some ground base term $s \in T_\Omega$. Consequently, $\mathcal{A}'(t) = \mathcal{A}'(s)$. ■

## Abstracted Clauses

◄ DEFINITION 3.20
Abstracted Clause

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

A clause $C$ is called ***abstracted*** if every literal in $C$ is pure. An abstracted clause $C$ is usually written in form

$$C = \Lambda \parallel \Gamma \to \Delta,$$

where $\Lambda$ consists of base *literals*, and $\Gamma$, $\Delta$ consist of free *atoms*.

Semantically, an abstracted clause $C = \Lambda \parallel \Gamma \to \Delta$ is an implication

$$\bigwedge \Lambda \wedge \bigwedge \Gamma \to \bigvee \Delta$$

between the conjunction of literals in $\Lambda$ and atoms $\Gamma$ on one hand side, and disjunction of atoms in $\Delta$, on the other, which is equivalent to the disjunction[1]

$$\bigvee \overline{\Lambda} \vee \bigvee \overline{\Gamma} \vee \bigvee \Delta.$$

The conjunctions $\bigwedge \Lambda$ and $\bigwedge \Gamma$, and disjunction $\bigvee \Delta$ of an abstracted clause $C$ are called the ***constraint***, the ***antecedent*** and the ***succedent*** of $C$, respectively.

[1]Recall, that for a (multi)set $M$ of literals, $\overline{M}$ denotes the (multi)set consisting of $M$'s literals negated.

The disjunctions $\bigvee\overline{\Lambda}$ and $\bigvee\overline{\Gamma}\vee\bigvee\Delta$ are called the ***base*** and ***free parts*** of $C$, respectively[1]. ■

In an abstracted clause $C = \Lambda \parallel \Gamma \to \Delta$, the double bar $\parallel$ symbol is used to separate the base part of a clause from the free one and does not carry itself any logical meaning. For the sake of conciseness, we usually write $\Lambda$, $\Gamma$ and $\Delta$ for the constraint, antecedent and succedent of $C$, respectively. Very often, the free part of $C$ is equivalently denoted as an implication $\Gamma \to \Delta$. As usual, $\Lambda$, $\Gamma$ and $\Delta$ may be empty and are then interpreted as *true*, *true*, *false*, respectively.

Any given disjunction of literals can be transformed into an abstracted clause of the form $\Lambda \parallel \Gamma \to \Delta$, where $\Lambda$ only contains base literals and all base terms in $\Gamma$, $\Delta$ are just base variables by the following transformation [BGW94], called ***abstraction***, or ***purification***. Whenever a subterm $t$, whose top symbol is a base symbol from $\Omega$, occurs immediately below a free operator symbol, it is replaced by a new base sort variable $x$ ("abstracted out") and the equation $x \approx t$ is added to $\Gamma$. Analogously, if a subterm $t$, whose top symbol is an extension symbol, occurs immediately below a base operator symbol, it is replaced by a new base variable $x$ and the equation $x \approx t$ is added to $\Gamma$. This transformation is repeated until all terms in the clause are pure; then all base atoms are moved to $\Lambda$ and all free ones to $\Gamma, \Delta$, respectively. Recall that $\Gamma, \Delta$ are sequences of atoms whereas $\Lambda$ holds theory literals. Moreover, we need to "purify" clauses only once — just before saturating the clauses, since if the premises of an inference are abstracted clauses, then the conclusion is also abstracted, Theorem 3.33.

Obviously, any abstracted clause is a member of the class $Cl_{\Sigma'}$ of all $\Sigma'$-clauses. The ***class of all base clauses*** (these are clauses with empty free part) is denoted by $Cl_{\Sigma}$. Clauses in $Cl_{\Sigma'} \setminus Cl_{\Sigma}$ are all non-base (these are clauses with non-empty free part).

Any ordering $\succ$ on terms can be extended to abstracted clauses in the following way. Like in the flat case, we consider clauses as multisets of occurrences of (dis)equations (recall that the antecedent and the succedent of a abstracted clause consist of equations, whereas the constraint may contain also disequations). An occurrence of an equation $s \approx t$ in the antecedent is identified with the multiset $\{s, s, t, t\}$, the occurrence of an equation $s \approx t$ in the succedent is identified with the multiset $\{s, t\}$; analogously, an occurrence of an equation $s \approx t$ in the constraint is identified with $\{s, s, t, t\}$, and an occurrence of a disequation $s \not\approx t$ in the constraint is identified with $\{s, t\}$. Now we overload $\succ$ on (dis)equation occurrences to be the multiset extension of $\succ$ on terms, and $\succ$ on clauses to be the multiset extension of $\succ$ on (dis)equation occurrences.

An antecedent or succedent occurrence of an equation $s \approx t$ is ***maximal*** (***minimal***) in an abstracted clause $C = \Lambda \parallel \Gamma \to \Delta$ if there is no occurrence of an equation in $\Gamma \to \Delta$ that is strictly greater (smaller) then the occurrence $s \approx t$ with respect to $\succ$. An antecedent or succedent occurrence of an equation $s \approx t$ is ***strictly maximal*** (***strictly minimal***) in the clause $C$ if there is no occurrence of an equation in $\Gamma \to \Delta$ that is greater (smaller) then or equal to the occurrence $s \approx t$ with respect to $\succ$. We do not consider occurrences of (dis)equations in the constraint of a clause for our maximality criteria. As clauses are abstracted, this is justified by

[1]Note the difference between the constraint (conjunction of $\Lambda$'s literals) and the base part (disjunction of $\Lambda$'s literals negated) of an abstracted clause.

considering a suitable path ordering, like LPO, where the operator symbols from $\Omega$ are smaller than all symbols in $\Omega'' = \Omega' \setminus \Omega$.

Given an abstracted clause $C = \Lambda \parallel \Gamma \rightarrow \Delta$, a selection function[1] *Sel* may assigns to $C$ a multiset of atoms only in the antecedent $\Gamma$ of the clause (literals in the constraint $\Lambda$ are not allowed being selected by any selection function).

**Weak Abstraction [BW13].** As already discussed in the Introduction chapter, the abstraction mechanism of [BGW94] is deficient in that it may actually destroy sufficient completeness of an initial clause set $N$, a sufficient condition for completeness of the hierarchic calculus SUP(T) on $N$.

The *weak abstraction* algorithm of Baumgartner and Waldmann [BW13] resolves this shortcoming of the original abstraction of Bachmair, Ganzinger, and Waldmann. The main idea of weak abstraction is to abstract out only pure non-variable non-constant T-terms. In [BW13] it is shown that the hierarchic superposition calculus modified in a straightforward way to fit the new abstraction algorithm is refutationally complete for all clause sets obtained by weak abstraction from sufficiently complete clause sets.

All results presented in this chapter are either formulated for general FOL(T) clauses, no matter abstracted or not, or do not depend on a particular form of abstracted clauses, as abstraction is regarded here simply as a *preprocessing step*. For instance, the Hierarchic Lifting Lemma (Lemma 3.71) and the Hierarchic Saturation Theorem (Theorem 3.72) are formulated for abstracted clauses, because the inference system $\mathcal{H}$ is defined for abstracted clauses. These Lemma and Theorem are also valid if simply reformulated for weakly abstracted clauses and for the variant of $\mathcal{H}$ inference system of [BW13], which captures the special form of weakly abstracted clauses in a straightforward way. Thus, the weak abstraction mechanism of Baumgartner and Waldmann can be safely integrated into our framework instead of [BGW94]'s abstraction method, causing no failure regarding achieved results.

### Simple Substitutions and Instances

◄ DEFINITION 3.21
Simple Substitution
Simple Instance

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

A substitution $\sigma$ is called ***simple*** if every base variable $x$ in the domain of $\sigma$ is mapped to a base term:

$$\sigma \ simple \overset{\text{def}}{\iff} \forall x \in dom(\sigma) : x \in \mathcal{X} \Rightarrow x\sigma \in T_{\Omega}(\mathcal{X}).$$

A term $t'$ is called a ***simple instance*** of a term $t$, if $t' = t\sigma$ for some simple substitution $\sigma$. The term $t'$ is a ***simple ground instance*** of the $t$ if $t'$ is ground. The set of all simple ground instances of $t$ is denoted by $sgi(t)$:

$$sgi(t) \overset{\text{def}}{=} \{t\sigma \mid \sigma \text{ simple}, t\sigma \in T_{\Omega'}\}.$$

■

[1]The notion of a selection function is introduced in Section 2.5, page 21.

Note that a term $t'$ that is a simple instance of some term $t$ is not necessarily base. Moreover, $t'$ is base if and only if the term $t$ is so. In other words, a simple instance of a base term is always base, whereas a simple instance $s'$ of a free term $s$ is always non-base, but not necessarily free. The notion of simple (ground) instance is naturally lifted to all expressions, particularly to literals, clauses, and sets thereof.

## 3.2.2 Semantics

DEFINITION 3.22 ► $\Sigma$-restriction $\mathcal{A}'|_\Sigma$

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively.

Given a $\Sigma'$-algebra $\mathcal{A}'$, the ***restriction of*** $\mathcal{A}'$ ***to*** $\Sigma$, written $\mathcal{A}'|_\Sigma$, is a $\Sigma$-algebra $\mathcal{A}$ that agrees with $\mathcal{A}'$ on base sorts and base operator symbols:

$$\mathcal{A}'|_\Sigma \overset{\text{def}}{=} \mathcal{A} \in \mathscr{A}_\Sigma : \big(\forall \mathsf{S} \in \mathcal{S}. \forall f \in \Omega : \mathsf{S}_\mathcal{A} = \mathsf{S}_{\mathcal{A}'}, f_\mathcal{A} = f_{\mathcal{A}'}\big)$$

■

Roughly speaking, the restriction $\mathcal{A}'|_\Sigma$ of a $\Sigma'$-algebra $\mathcal{A}'$ to $\Sigma$ is a $\Sigma$-algebra that is obtained from $\mathcal{A}'$ by leaving only the base part of the domain and interpretation of base operator symbols, or equivalently, by removing all free carrier sets $\mathsf{S}''_{\mathcal{A}'}$, for every free sort $\mathsf{S}'' \in \mathcal{S}''$, and all functions $g_{\mathcal{A}'}$, for every free operator symbol $g \in \Omega''$. As $\mathcal{A}'$ is total and $\Sigma \subseteq \Sigma'$, the $\Sigma$-algebra $\mathcal{A} = \mathcal{A}'|_\Sigma$ always exists.

Note that $\mathcal{A}'|_\Sigma$ is not necessarily term-generated, even if $\mathcal{A}'$ is term-generated. Indeed: if, for instance, there exists some element $e' \in \mathsf{S}_{\mathcal{A}'}$ in the universe of $\mathcal{A}'$, such that all ground terms $t$, whose interpretations $t_{\mathcal{A}'}$ under $\mathcal{A}'$ equal $e'$, have top symbols $g = top(t)$, and all such $g$'s come from the set of free operator symbols $\Omega''$, then after removing from $\mathcal{A}'$ all functions $g_{\mathcal{A}'}$, there is no term whose interpretation under $\mathcal{A}'|_\Sigma$ equals $e'$.

DEFINITION 3.23 ► Hierarchic algebra

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

A $\Sigma'$-algebra $\mathcal{A}'$ is called ***hierarchic*** if its restriction to $\Sigma$ is a base algebra. The class of all hierarchic algebras is denoted by $\mathscr{H}_{\mathsf{HSp}}$:

$$\mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}} \overset{\text{def}}{\iff} \mathcal{A}'|_\Sigma \in \mathscr{C}.$$

Given a $\Sigma'$-formula $F \in F_{\Omega'}$ and a $\Sigma'$-algebra $\mathcal{A} \in \mathscr{A}_{\Sigma'}$, we write $\mathcal{A}' \models_\mathscr{C} F$ if $\mathcal{A}'$ is a hierarchic model of $F$:

$$\mathcal{A}' \models_\mathscr{C} F \overset{\text{def}}{\iff} \mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}}, \mathcal{A}' \models F.$$

■

Informally speaking, a $\Sigma'$-algebra $\mathcal{A}'$ is hierarchic, if $\mathcal{A}'$ extends some base model $\mathcal{A} \in \mathscr{C}$ and neither collapses any its sort nor adds new elements to it.

DEFINITION 3.24 ► Model of HSp

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

A hierarchic algebra $\mathcal{A}'$ is called a ***model of a hierarchic specification*** $\mathsf{HSp}$, if it is a model of the axiom set $Ax'$:

$$\mathcal{A}' \text{ a model of } \mathsf{HSp} \overset{\text{def}}{\iff} \mathcal{A}' \models_{\mathscr{C}} Ax'.$$

■

◄ DEFINITION 3.25
$\mathscr{C}$-consistent
$\mathscr{C}$-entailment

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

Let $N$ and $M$ be two sets of $\Sigma'$-clauses. We call $N$ ***consistent relative to*** $\mathscr{C}$ (or, shortly, $\mathscr{C}$***-consistent***, or ***theory-consistent***), if there exists a hierarchic model of $N$:

$$N\ \mathscr{C}\text{-consistent} \overset{\text{def}}{\iff} \exists \mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}} : \mathcal{A}' \models N,$$

and otherwise we call it ***inconsistent relative to*** $\mathscr{C}$ (or $\mathscr{C}$***-inconsistent***, or ***theory-inconsistent***), written $N \models_{\mathscr{C}} \perp$.

We say $N$ ***implies/ entails*** $M$ ***relative to*** $\mathscr{C}$, written $N \models_{\mathscr{C}} M$, if every hierarchic model of $N$ is also a model of $M$:

$$N \models_{\mathscr{C}} M \overset{\text{def}}{\iff} \forall \mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}} : \mathcal{A}' \models N \Rightarrow \mathcal{A}' \models N.$$

■

Alternatively, we can say $N \models_{\mathscr{C}} M$ if given a base specification $\mathsf{Sp}$ and a body signature $\Sigma'$, the hierarchic specification $\mathsf{HSp} = (\mathsf{Sp}, (\Sigma', \{N \to M\}))$ has a model. If $N$ is a set of base clauses, it is $\mathscr{C}$-consistent if and only if some base algebra $\mathcal{A}$ in $\mathscr{C}$ is a model of $N$. Note that for any clause sets $N$ and $M$, $N \models_{\mathscr{C}} M$ follows from $N \models M$, but not other way around, in general.

Refutational theorem proving for hierarchic theories is aimed at answering one the following questions: given a clause set $N$ over the body signature $\Sigma'$,

- is $N$ false in all models of the hierarchic specification $\mathsf{HSp}$?
- does a hierarchic specification $(\mathsf{Sp}, (\Sigma', Ax' \cup N))$ have no model? or
- is $Ax' \cup N$ inconsistent relative to $\mathscr{C}$?

Each of the questions is a reformulation of the others, and, hence, they are all equivalent. One can think of a more general problem, namely: whether the set of axioms $Ax'$ implies a closed $\Sigma'$-formula $F$ (or a set thereof) relative to $\mathscr{C}$:

$$Ax' \models_{\mathscr{C}} F,$$

or in other words, whether $F$ holds in every model of the hierarchic specification $\mathsf{HSp}$. If $\neg F$ transformed to an equisatisfiable clause set $N$ (meaning, $N$ is satisfiable iff $\neg F$ is so), then solving the problem of $Ax' \models_{\mathscr{C}} F$ is equivalent to answering any of the above three questions.

## 3.3 SUP(T) Calculus

Assume HSp = (Sp, Sp′) is a hierarchic specification with the base specification Sp = $(\Sigma, \mathscr{C})$ and the body Sp′ = $(\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

In Definitions 3.26-3.31 we present the inference rules constituting the Superposition Modulo Theory SUP(T) calculus. All rules are defined for abstracted clauses and with respect to the hierarchic specification HSp.

DEFINITION 3.26 ► Hierarchic Splitting

The ***Hierarchic Splitting*** rule is

$$\mathcal{S}\,\frac{\Lambda_1, \Lambda_2 \parallel \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2}{\Lambda_1 \parallel \Gamma_1 \to \Delta_1 \quad \Big| \quad \Lambda_2 \parallel \Gamma_2 \to \Delta_2}$$

where

(i) $var(\Lambda_1 \parallel \Gamma_1 \to \Delta_1) \cap var(\Lambda_2 \parallel \Gamma_2 \to \Delta_2) = \emptyset$,

(ii) $\Delta_1 \neq \emptyset$ and $\Delta_2 \neq \emptyset$.

■

DEFINITION 3.27 ► Hierarchic Superposition Left

The ***Hierarchic Superposition Left*** rule is

$$\mathcal{I}\,\frac{\Lambda_1 \parallel \Gamma_1 \to \Delta_1, l \approx r \qquad \Lambda_2 \parallel s[l'] \approx t, \Gamma_2 \to \Delta_2}{(\Lambda_1, \Lambda_2 \parallel s[r] \approx t, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2)\sigma}$$

where

(i) $\sigma = mgu(l, l')$ and $\sigma$ is simple,

(ii) $l'$ is not a variable,

and for some simple grounding substitution $\psi$

(iii) $l\sigma\psi \succ r\sigma\psi$,

(iv) $s\sigma\psi \succ t\sigma\psi$,

(v) $(l \approx r)\sigma\psi$ is strictly maximal in $(\Gamma_1 \to \Delta_1, l \approx r)\sigma\psi$,

(vi) $(s \approx t)\sigma\psi$ is maximal in $(s \approx t, \Gamma_2 \to \Delta_2)\sigma\psi$ and no literal in $\Gamma_2$ is selected
or
$s \approx t$ is selected,

(vii) no literal in $\Gamma_1$ is selected.

■

We do not consider the terms in the theory part of the clauses for our maximality criteria: as clauses are abstracted, this is justified by considering a suitable path ordering, like LPO, where precedence is defined in such a way that the operator symbols from $\Omega$ are smaller than all symbols in $\Omega'' = \Omega' \setminus \Omega$, or transfinite KBO [LW07] with weights of base symbols equal to some natural numbers and weights of free symbols equal to some ordinal numbers. In such an ordering, any simple ground instance of a base literal, which is again base, is strictly smaller than any simple ground instance of a free literal, which is non-base.

Also, in this work we use reduction orderings, which are enjoin to orient all extension-terms greater than any extension-free term. Such an orderings can be, for instance, an LPO, where precedence is defined in such a way that all extension function symbols are greater than any non-extension symbol, or a transfinite KBO [LW07] with appropriately chosen weights.

◄ DEFINITION 3.28
Hierarchic Superposition Right

The ***Hierarchic Superposition Right*** rule is

$$\mathcal{I}\frac{\Lambda_1 \parallel \Gamma_1 \rightarrow \Delta_1, l \approx r \qquad \Lambda_2 \parallel \Gamma_2 \rightarrow \Delta_2, s[l'] \approx t}{(\Lambda_1, \Lambda_2 \parallel \Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2, s[r] \approx t)\sigma}$$

where

(i) $\sigma = mgu(l, l')$ and $\sigma$ is simple,

(ii) $l'$ is not a variable,

and for some simple grounding substitution $\psi$

(iii) $l\sigma\psi \succ r\sigma\psi$,

(iv) $s\sigma\psi \succ t\sigma\psi$,

(v) $(l \approx r)\sigma\psi$ is strictly maximal in $(\Gamma_1 \rightarrow \Delta_1, l \approx r)\sigma\psi$,

(vi) $(s[l'] \approx t)\sigma\psi$ is strictly maximal in $(s[l'] \approx t, \Gamma_2 \rightarrow \Delta_2)\sigma\psi$,

(vii) $(s[l'] \approx t)\sigma\psi \succ (l \approx r)\sigma\psi$,

(viii) no literal in $\Gamma_1$, $\Gamma_2$ is selected.

■

An application of Hierarchic Superposition Left/Right rule may produce a variable assignment $(s[r] \approx t)\sigma$, if $l'$ coincides with $s$, and $r\sigma$ and $t\sigma$ are variables. If in this case $r$ and $t$ are base variables, the resulting variable assignment $(s[r] \approx t)\sigma$ is a base atom and, therefore, is moved to the constraint of the inference's conclusion, i.e. the superposition left inference yields then the clause $(\Lambda_1, \Lambda_2, s[r] \approx t \parallel \Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2)\sigma$ and superposition right — the clause $(\Lambda_1, \Lambda_2, s[r] \not\approx t \parallel \Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2)\sigma$. It was not needed in the previous formulation of the calculus [BGW94] because there no strict separation between base terms and free terms of a clause was made. However, it is very useful to separate the base from the non-base literals in order to explore the reasoning mechanisms needed for the base specification.

Let $C_1$ and $C_2$ be the left and right premises of the superposition left/right rules, respectively. From the conditions of Definitions 3.27, 3.28 it follows that $C_1\sigma \not\succ C_2\sigma$ (and $C_1\sigma \not\succeq C_2\sigma$ in the case of the superposition right rule).

◄ DEFINITION 3.29
Hierarchic Equality Resolution

The ***Hierarchic Equality Resolution*** rule is

$$\mathcal{I}\frac{\Lambda \parallel \Gamma, s \approx t \rightarrow \Delta}{(\Lambda \parallel \Gamma \rightarrow \Delta)\sigma}$$

where

(i) $\sigma = mgu(s, t)$ and $\sigma$ is simple,

and for some simple grounding substitution $\psi$

(ii) $(s \approx t)\sigma\psi$ is maximal in $(\Gamma, s \approx t \rightarrow \Delta)\sigma\psi$ and no literal in $\Gamma$ is selected
or
$s \approx t$ is selected.

■

DEFINITION 3.30 ► Hierarchic Equality Factoring

The ***Hierarchic Equality Factoring*** rule is

$$\mathcal{I}\frac{\Lambda \parallel \Gamma \rightarrow \Delta, l \approx r, l' \approx r'}{(\Lambda \parallel \Gamma, r \approx r' \rightarrow \Delta, l' \approx r')\sigma}$$

where

(i) $\sigma = mgu(l, l')$ and $\sigma$ is simple,

and for some simple grounding substitution $\psi$

(ii) $l\sigma\psi \succ r\sigma\psi$,

(iii) $(l \approx r)\sigma\psi$ is maximal in $(\Lambda \parallel \Gamma \rightarrow \Delta, l \approx r, l' \approx r')\sigma\psi$,

(iv) no literal in $\Gamma$ is selected.

■

Standard factoring of non-equality literals is a special variant of the equality factoring rule where $r = r' = true_P$, for some predicate symbol $P$ which is the top symbol of $l$ and $l'$, and hence the literal $(r \approx r')\sigma$ is actually $true_P \approx true_P$ and can be eliminated.

DEFINITION 3.31 ► Constraint Refutation

The ***Constraint Refutation*** rule is

$$\mathcal{I}\frac{\Lambda_1 \parallel \rightarrow \quad \dots \quad \Lambda_n \parallel \rightarrow}{\Box}$$

where $(\Lambda_1 \parallel \rightarrow), \dots, (\Lambda_n \parallel \rightarrow) \models_{\mathscr{C}} \bot$. ■

The set of premises of the Constraint Refutation rule consists of clauses $C_1 = (\Lambda_1 \parallel \rightarrow)$ to $C_n = (\Lambda_n \parallel \rightarrow)$, each of which is *base*, i.e. the free part of every $C_i$ is empty. The rule is performed if no base algebra $\mathcal{A} \in \mathscr{C}$ is a simultaneous model of the base premises $C_1, \dots, C_n$; more formally, $(\Lambda_1 \parallel \rightarrow), \dots, (\Lambda_n \parallel \rightarrow) \models_{\mathscr{C}} \bot$, iff for every base algebra $\mathcal{A} \in \mathscr{C}$ it holds that

$$\begin{aligned} & \mathcal{A} \not\models \bigwedge_i \forall \vec{x}^i.\neg\Lambda_i && \text{// where } \vec{x}^i = var(\Lambda_i) \\ \Leftrightarrow \quad & \mathcal{A} \models \bigvee_i \exists \vec{x}^i.\Lambda_i \end{aligned}$$

Thus, the rule applies iff the existential closure $\exists\vec{x}^i.\Lambda_i$ of the constraint of *at least one base clause* among $C_1 = (\Lambda_1 \parallel \rightarrow)$, …, $C_n = (\Lambda_n \parallel \rightarrow)$ is valid in the base theory (satisfiable by every base algebra). Equivalently, the rule is not applied, iff there exists a base algebra $\mathcal{A} \in \mathscr{C}$ such that

$$\begin{aligned} & \mathcal{A} \not\models \bigvee_i \exists \vec{x}^i.\Lambda_i \\ \Leftrightarrow \quad \forall i \in \{1, \dots, n\} : \; & \mathcal{A} \not\models \exists \vec{x}^i.\Lambda_i \\ \Leftrightarrow \quad & \mathcal{A} \models \bigwedge_i \forall \vec{x}^i.\neg\Lambda_i \\ & = \bigwedge_i C_i \end{aligned}$$

i.e. the rule is not applicable iff there exists a base algebra $\mathcal{A} \in \mathscr{C}$ satisfying all the base clauses $C_1, \ldots, C_n$, or equivalently, falsifying the existential closure $\exists \vec{x}^i.\Lambda_i$ of the constraint of *every base clause* in $C_1, \ldots, C_n$.

Recall that we encode predicates as functions, which is needed to compactly define the calculus. According to the definition of the SUP(T) inference rules, only free literals are superposed, therefore encoding base predicates as functions is not strictly necessary. Still, we do it as it helps to compress the theoretical discussion by saving one extra case (of considering predicative atoms). Whenever the Constraint Refutation rule is performed and the background T-solver is invoked, every such base predicative (dis)equation $P(t_1,\ldots,t_n) \mathrel{\dot{\approx}} true_P$, where $\dot{\approx} \in \{\approx, \not\approx\}$, is passed to the solver as a corresponding predicate $P(t_1,\ldots,t_n)$ or $\neg P(t_1,\ldots,t_n)$, respectively.

◄ DEFINITION 3.32
Inference system $\mathcal{H}$

The inference rules Hierarchic Equality Resolution, Hierarchic Equality Factoring, Hierarchic Superposition Left/Right, and Constraint Refutation constitute the ***inference system*** $\mathcal{H}$. ■

◄ THEOREM 3.33
Abstracted Conclusion
[BGW94]

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*If N is a set of abstracted clauses, then every clause* $C \in concl(\mathcal{H}(N))$*—a conclusion of an* $\mathcal{H}$*-inference with premises in N—is abstracted.*

Discarding separation of base and free parts of clauses and simple unifiers, the rules Hierarchic Splitting, Hierarchic Superposition Left/Right, and Hierarchic Equality Resolution are obvious instances of the respective standard superposition inference rules [BG94, Wei01], therefore the hierarchic rules are sound with respect to the general entailment relation $\models$, hence sound with respect to the hierarchic entailment relation $\models_{\mathscr{C}}$ (recall that $N \models M$ implies $N \models_{\mathscr{C}} M$). The rule Constraint Refutation is not sound with respect to the general entailment relation $\models$ (as for a set of base clauses inconsistent relative to the theory T, there might exist a non-standard model of T, – a model out of the base model class $\mathscr{C}$, – that actually satisfies the given set of base clauses), but the rule is evidently sound with respect to $\models_{\mathscr{C}}$. Thus, the overall inference system consisting of the rules Hierarchic Splitting, Hierarchic Superposition Left/Right, Hierarchic Equality Resolution, and Constraint Refutation is sound with respect to $\models_{\mathscr{C}}$.

◄ LEMMA 3.34
$\mathcal{H}$ Soundness

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*The rules Hierarchic Splitting, Hierarchic Superposition Left/Right, Hierarchic Equality Resolution, and Constraint Refutation are sound with respect to the hierarchic entailment relation* $\models_{\mathscr{C}}$.

Besides soundness, we need in addition to show refutational completeness of the SUP(T) calculus. The next section is entirely devoted to this topic.

## 3.4 Completeness of SUP(T)

This section is devoted to proving refutational completeness of the hierarchic superposition calculus SUP(T). The completeness for SUP(T) will be primarily obtained by lifting Bachmair and Ganzinger's completeness result [BG91, BG94] of the flat (standard) superposition for ground clauses.

For the rest of the section, we consider a hierarchic specification $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ is the base signature and $\Sigma' = (\mathcal{S}', \Omega')$ is the body signature (please, recall Definition 3.15 of a hierarchic specification). We put $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ to be the set of enrichment sorts, and $\Omega'' = \Omega' \setminus \Omega$ the set of enrichment operators. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

### 3.4.1 Overview

**Long story short**. The idea of lifting completeness of the standard superposition for ground clauses is motivated by well-studied clear application of lifting towards establishing completeness of the standard superposition SUP for general clauses. So, let us first recall how it is done for the standard superposition. Assume, we are given a set $N$ of general FOL clauses, saturated with respect to SUP and not containing an empty clause. Obviously, the set $gi(N)$ of all ground instances of clauses in $N$ also does not contain an empty clause. From the assumption that $N$ is saturated it follows that $gi(N)$ is also saturated, which is a consequence of so-called *Lifting Lemma*. Roughly speaking, the lemma states that if there exists a non-redundant ground inference $I'$ with premises from $gi(N)$, then there exists a non-redundant inference $I$ with premises from $N$, such that the former is a ground instance of the latter: $I' = I\sigma$, for some ground substitution $\sigma$. Any SUP-saturated ground clause set that does not contain an empty clause has a (Herbrand) model. Thus, $gi(N)$ has a model, consequently, the set $N$ is satisfiable (under the model satisfying $gi(N)$).

We shall lift completeness of the flat superposition for the ground clauses to hierarchic superposition for general abstracted clauses following a lifting schema similar to the one exploited for the case of the standard superposition for general FOL clauses. However, we have to take care of several specialties that are due to the combination with the background theory T, which does not allow us to directly use exactly the same lifting mechanism.

- First of all, in contrast to the flat case, where a clause set $N$ is satisfiable if and only if there exists *any* algebra that is a model of the set $gi(N)$ of all ground instances of $N$, in the hierarchic case we are interested only in those models of $gi(N)$ that are *hierarchic* algebras[1], i.e. algebras whose restriction to the base signature $\Sigma$ is a base algebra from the class $\mathscr{C}$.
- Second, the principle of modularity underlying the hierarchic superposition calculus dictates a necessity of using only *simple* substitutions[2] to find unifiers of terms in the inference rules of SUP(T), which is essential for keep-

[1]The notion of a *hierarchic algebra* is introduced in Definition 3.23, page 44.
[2]The notion of a *simple substitution* is introduced in Definition 3.21, page 43.

ing constraints of abstracted clauses pure, as otherwise, if non-simple unifiers admitted, the constraint of an inference's conclusion may become non-base, which makes it impossible to use the background theory solver to deal with such constraints (for instance, in the Constraint Refutation rule and in the reduction rules Hierarchic Tautology and Subsumption Deletion). For this reason, we cannot rely on the Lifting Lemma in its standard formulation anymore, because saturation of clauses in $N$ with respect to the hierarchic calculus with simple mgu's, in general, does not imply saturation of all their ground instances $gi(N)$ with respect to the flat calculus. Maximum to what we can pretend in this situation is the saturation of the set $sgi(N)$ of all *simple* ground instances of $N$.

– But, then we face a problem of mutual hierarchic model existence: not every model of the set $sgi(N)$ is a hierarchic model of $N$, moreover, it might not even be a "flat" model of $N$.

For these reasons, merely correlating $N$ to $sgi(N)$, certainly, is not sufficient (in contrast to the flat case, where correlating $N$ to $gi(N)$ does the job). To this end, we need to answer a question:

> 'To what ground set $N'$ should we extend/modify the set $sgi(N)$ such that $N'$ meets the above requirements of *mutual saturation* and *mutual hierarchic model existence* relative to $N$?'

Next, we informally discuss our approach to resolving the question stated. But first, let us understand the following notation: we use the pairs $(\mathcal{H},\mathcal{R}^{\mathcal{H}})$ and $(\mathcal{F},\mathcal{R}^{\mathcal{F}})$ to denote the hierarchic and the flat superposition calculi, respectively, where

– $\mathcal{H}$ is the hierarchic inference system, presented in Section 3.3;

– $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_I,\mathcal{R}^{\mathcal{H}}_F)$ hierarchic redundancy criterion specifying redundant inferences $\mathcal{R}^{\mathcal{H}}_I$ and redundant formula $\mathcal{R}^{\mathcal{H}}_F$, respectively; we define $\mathcal{R}^{\mathcal{H}}$ formally later on in Section 3.4.4;

– $\mathcal{F}$ and $\mathcal{R}^{\mathcal{F}} = (\mathcal{R}^{\mathcal{F}}_I,\mathcal{R}^{\mathcal{F}}_F)$ stand for the flat inference system and ground redundancy criterion specifying redundant inferences $\mathcal{R}^{\mathcal{F}}_I$ and redundant formula $\mathcal{R}^{\mathcal{F}}_F$, respectively; we introduce the notions $\mathcal{F}$ and $\mathcal{R}^{\mathcal{F}}$ formally in Section 3.4.2.

The mutual saturation requirement can be simply gained by setting the hierarchic redundancy criterion accounting for the simple substitutions condition. So, we set $C \in \mathcal{R}^{\mathcal{H}}_F(N)$ if $sgi(C) \subseteq \mathcal{R}^{\mathcal{F}}_F(sgi(N))$, i.e. a clause is redundant if all its simple ground instances are redundant for the set of all simple ground instances of $N$. Similarly we define a redundant inference: $I \in \mathcal{R}^{\mathcal{H}}_I(N)$ if $sgi(N) \subseteq \mathcal{R}^{\mathcal{F}}_I(sgi(N))$. The latter does not apply to the constraint refutation inference rule, as there is no "flat" version of it in the standard SUP-calculus, therefore[1] we say that a constraint refutation inference is redundant if an empty clause is already in $N$.

[1]It should be noted that this formulation of $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_F,\mathcal{R}^{\mathcal{H}}_I)$ is too weak. We shall refine it in Section 3.4.4, and for the current informal overview of the overall completeness proof we stick to the above rough definition of $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_F,\mathcal{R}^{\mathcal{H}}_I)$. Actually, Bachmair, Ganzinger, and Waldmann suggested in their work [BGW94] precisely this definition of $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_F,\mathcal{R}^{\mathcal{H}}_I)$.

Realization of the requirement of mutual hierarchic model existence needs more effort. For a term-generated algebra $\mathcal{A}'$ to be a hierarchic model of $N$, it has to (i) satisfy the set $gi(N)$ of all ground instances of $N$, and (ii) its restriction $\mathcal{A}'|_\Sigma$ to the base signature has to be in $\mathscr{C}$. So, for (i) we need to somehow ensure that every clause in $gi(N)$ is a logical consequence of clauses in the extended ground set $N'$, and for (ii) we have to ensure then that every model $\mathcal{A}'$ of $N'$ is hierarchic. A possibility to satisfy the first requirement is to restrict our attention only to such clause sets $N$, whose set $sgi(N)$ of all simple ground instances has the property, that in every model of $sgi(N)$ every ground non-base term $t'$ of a base sort is interpreted the same as some ground base term $t$, i.e. $sgi(N) \models t' \approx t$. Then, in every model $\mathcal{A}'$ of the set $sgi(N)$ of all simple ground instances of a clause set $N$ possessing this property, every ground substitution is equivalent to some simple ground substitution, therefore $\mathcal{A}'$ is also a model of the set $gi(N)$ of all ground instances of $N$, hence a model of $N$. Since this is an integral property of a clause set (which is the only one we stipulate for an input), it is called *sufficient completeness with respect to simple ground instances* [BGW94], or simply *sufficient completeness*.

Now, we need to ensure $\mathcal{A}'$ to be a hierarchic algebra, wherefore $\mathcal{A}'$ must extend some base algebra $\mathcal{A} \in \mathscr{C}$ and neither delete elements from, nor add new ones to the universe $U_\mathcal{A}$ of $\mathcal{A}$. Speaking more formally, $\mathcal{A}'$ is hierarchic, if its restriction $\mathcal{A}'|_\Sigma$ to the base signature is isomorphic to $\mathcal{A} \in \mathscr{C}$. Let $\tilde{\mathsf{E}}_\mathcal{A}$ be the set of all positive unit base clauses, each of which consists of a single equation between two distinct ground base terms that are equal under $\mathcal{A}$, or, put it other way round, if two distinct ground base terms have the same interpretation in $\mathcal{A}$ then an equation between them is in $\tilde{\mathsf{E}}_\mathcal{A}$. If $\mathcal{A}'$ is a model of $\tilde{\mathsf{E}}_\mathcal{A}$, then $\mathcal{A}'$ is a homomorphic extension of $\mathcal{A}$. But, this is not sufficient yet, because $\mathcal{A}'$ might still add "junks" into the base sorts, which constitute the universe of $\mathcal{A}$, or delete some elements from them. Let $\tilde{\mathsf{D}}_\mathcal{A}$ be the set of all negative unit base clauses, each of which consists of a single disequation between two ground base terms that are not equal under $\mathcal{A}$, or in other words, if two ground base terms have different interpretation under $\mathcal{A}$ then a disequation between them is in $\tilde{\mathsf{D}}_\mathcal{A}$. If $\mathcal{A}'$ also satisfies $\tilde{\mathsf{D}}_\mathcal{A}$ – in addition to $\tilde{\mathsf{E}}_\mathcal{A}$, – then $\mathcal{A}'$ does not collapse any of base sorts, because, for otherwise, as every $\mathcal{A} \in \mathscr{C}$ is term-generated, there would exist two ground base terms with different interpretation under $\mathcal{A}$, but equally interpreted under $\mathcal{A}'$, which cannot happen if $\mathcal{A}' \models \tilde{\mathsf{D}}_\mathcal{A}$. It is easy to see, that if $\mathcal{A}'$ is a model of $\tilde{\mathsf{E}}_\mathcal{A}$ and $\tilde{\mathsf{D}}_\mathcal{A}$, then $\mathcal{A}'$ is a monomorphic extension of $\mathcal{A}$. The only thing left is preventing $\mathcal{A}'$ from adding extra elements to the universe of $\mathcal{A}$. This can happen only if the model $\mathcal{A}'$ of $sgi(N) \cup \tilde{\mathsf{E}}_\mathcal{A} \cup \tilde{\mathsf{D}}_\mathcal{A}$ evaluates some ground non-base term $t'$ of a base sort to some value that no base term $t$ has under $\mathcal{A}'$. But this is already fixed by the *sufficient completeness* property of $N$, considered in the previous paragraph, according to which $sgi(N) \models t' \approx t$.

Putting everything together, we obtain that if the set $N$ enjoys the sufficient completeness property, then every model $\mathcal{A}'$ of the set $N' = sgi(N) \cup \tilde{\mathsf{E}}_\mathcal{A} \cup \tilde{\mathsf{D}}_\mathcal{A}$, where $\mathcal{A} \in \mathscr{C}$ a base algebra, is a hierarchic model of $N$. Since it is not clear a-priori, if $sgi(N)$ has a hierarchic model, a concern is, 'what base algebra $\mathcal{A}$ should be taken to construct a set $N' = sgi(N) \cup \tilde{\mathsf{E}}_\mathcal{A} \cup \tilde{\mathsf{D}}_\mathcal{A}$?' If $N$ contains base clauses (those are all clauses that are built solely over base symbols) then for $\mathcal{A}$ we pick any base algebra satisfying the base subset of $N$. If the base subset of $N$ is theory-inconsistent, then due to compactness of the base specification there exist finitely many base clauses in $N$ that are theory-inconsistent, hence the Constraint Refutation rule applied to these clauses produces an empty clause, which is then also contained in the set *sgi*($N$), as $N$ is saturated, therefore *sgi*($N$) is unsatisfiable anyway. If $N$ does not contain base clauses, then we pick any base algebra $\mathcal{A} \in \mathscr{C}$. Later on, we show that for any $\mathcal{A}$ picked up this way, the set $N' = sgi(N) \cup \tilde{\mathsf{E}}_{\mathcal{A}} \cup \tilde{\mathsf{D}}_{\mathcal{A}}$ has a hierarchic model if and only if $N$ does.

**Refinements**. Evidently, the set $\tilde{\mathsf{E}}_{\mathcal{A}}$ contains a huge number of redundant clauses which are entailed by other clauses in $\tilde{\mathsf{E}}_{\mathcal{A}}$. So, we set $\mathsf{E}_{\mathcal{A}}$ to be the minimal (with respect to the underlying reduction ordering $\succ$) subset of $\tilde{\mathsf{E}}_{\mathcal{A}}$ that entails every clause in $\tilde{\mathsf{E}}_{\mathcal{A}}$. In the presence of $\mathsf{E}_{\mathcal{A}}$, the set $\tilde{\mathsf{D}}_{\mathcal{A}}$ also contains much redundancy, therefore we let $\mathsf{D}_{\mathcal{A}}$ be the minimal subset of $\tilde{\mathsf{D}}_{\mathcal{A}}$ such that $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ entails every clause in $\tilde{\mathsf{D}}_{\mathcal{A}}$ (we shall give a formal definition of $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$ later on in Section 3.4.3, and for the present we stick to the above definitions as they suffice for an informal discussion of SUP(T)'s completeness).

The solution to consider only those clause sets $N$ possessing the property

$$sgi(N) \models t' \approx t,$$

for all ground non-base terms $t'$ of a base sort and some ground base terms $t$, is too strong, meaning that the class of all such clause sets $N$ is quite small and not really interesting[1]. Moreover, the formulation above is a bit superfluous, because, first, we do not need this property to be defined with respect to *every* model of *sgi*($N$), but rather to every model satisfying, in addition, the sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$; and, second, according to the Extension Terms Lemma (Lemma 3.19) it suffices to sufficiently define only smooth extension terms[2], rather than all non-base terms of a base sort. For this reason, we consider only those clause sets $N$ for which the following holds[3]:

$$\forall \mathcal{A} \in \mathscr{C} : sgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \models t' \approx t,$$

for all smooth extension terms $t'$ and some ground base terms $t$. As it turns out, every $\Sigma'$-algebra $\mathcal{A}'$ satisfying the sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, for some base algebra $\mathcal{A}$, is actually a monomorphic extension of $\mathcal{A}$. As every such $\mathcal{A}'$ is *potentially* a hierarchic algebra, we call such algebras *weak with respect to* $\mathscr{C}$, or simply *weak*. We give a formal definition of a weak algebra in Definition 3.73 and discuss the notion in Section 3.4.6. The final formulation of the sufficient completeness property will be given with respect to weak algebras in Definition 3.78, and in Section 3.4.7 we shall discuss the notion in detail.

**Outline**. In Section 3.4.2 we recall the standard superposition calculus $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$ for ground clauses, the completeness result of which we shall later on lift to the hierarchic calculus $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$ for general clauses. In Section 3.4.3 we give formal definitions of the sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, and show some useful properties thereof, particularly, we prove that any ground theory-consistent base clause is a logical consequence of a *finite subset* of $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, and precisely identify clauses in such

[1]Actually, Bachmair, Ganzinger, and Waldmann suggested in their work [BGW94] precisely this solution.

[2]Please, recall Definition 3.17 of an extension term.

[3]Please, note that base algebras $\mathcal{A} \in \mathscr{C}$ are only used for construction of the clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$. The entailment relation "$\models$" above is a subject to *all* $\Sigma'$*-algebras*, in contrast to "$\models_{\mathscr{C}}$".

subsets. Thereafter, we introduce notions of $R^{\approx}_{\mathcal{A}}$-reduced terms, substitutions, and instances. $R^{\approx}_{\mathcal{A}}$ is a rewrite system consisting of rewrite rules $l \rightarrow r$, for each of which there is a clause $(\rightarrow l \approx r)$ in $\mathsf{E}_{\mathcal{A}}$. The main property of $R^{\approx}_{\mathcal{A}}$-reduced instances is that every simple ground non-reduced instance $C\sigma$ of a clause $C$ is a logical consequence of the respective $R^{\approx}_{\mathcal{A}}$-reduced instance $C\sigma'$ and *finitely many* clauses from $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$. This property is indispensable for presentation of hierarchic redundancy criterion and proving the Hierarchic Lifting Lemma. In Section 3.4.4 we give a formal definition of the hierarchic redundancy criterion $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_{F}, \mathcal{R}^{\mathcal{H}}_{I})$, and prove that $\mathcal{R}^{\mathcal{H}}$ does really confirm with the notion of a redundancy criterion. In Section 3.4.5 we prove the *Hierarchic Lifting Lemma* – a variant of the Lifting Lemma formulated for the hierarchic superposition calculus $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$, and based on it show in the *Hierarchic Saturation Theorem* that saturation of a set of abstracted clauses $N$ with respect to $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$ implies saturation of the set $sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ with respect to $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$, where $sgi_{\mathcal{A}}(N)$ stands for the set of all simple ground $R^{\approx}_{\mathcal{A}}$*-reduced* instances of $N$. In Section 3.4.6 we formally define the notion of a weak algebra, and provide sufficient and necessary conditions that determine a weak algebra. In Section 3.4.7 we define the *Sufficient Completeness* property that describes the class of clause sets for which the hierarchic superposition is refutationally complete. In *Hierarchic Model Lemma* we prove that a sufficiently complete clause set has a model if and only if the set $sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ is satisfiable, for some base algebra $\mathcal{A}$. In Section 3.4.8 we finally prove that SUP(T) is refutationally complete mainly based on the $\mathcal{H} - \mathcal{F}$ *Approximation Theorem* where we assert that the hierarchic superposition calculus $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$ approximates the flat superposition for ground clauses $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$, and in the *Hierarchic Completeness Theorem* we assert the completeness of SUP(T).

## 3.4.2 Standard Superposition for Ground Clauses SUP

For the purpose of completeness lifting we only need the ground version of Bachmair and Ganzinger's superposition calculus [BG91, BG94]. Here we recall the set $\mathcal{F}$ of basic inference rules (Definitions 3.35-3.38) constituting the standard superposition calculus SUP for ground clauses. Then, we give a definition of the standard redundancy criterion $\mathcal{R}^{\mathcal{F}}$ (Definition 3.40) for ground clauses, based on which later on in Section 3.4.4 we define the hierarchic redundancy criterion $\mathcal{R}^{\mathcal{H}}$. In the sequel we will often refer to the standard SUP calculus as the *flat calculus*.

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$. Premises and conclusions of all SUP-rules, presented below, are ground $\Sigma'$-clauses.

DEFINITION 3.35 ► Ground Superposition Left

The ***ground superposition left*** rule is

$$\mathcal{I}\frac{\Gamma_1 \rightarrow \Delta_1, l \approx r \qquad s[l] \approx t, \Gamma_2 \rightarrow \Delta_2}{s[r] \approx t, \Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2}$$

where

(i) $l \succ r$,

(ii) $s \succ t$,

(iii) $l \approx r$ is strictly maximal in $\Gamma_1 \rightarrow \Delta_1, l \approx r$,

(iv) $s[l] \not\approx t$ is maximal in $s[l] \approx t, \Gamma_2 \to \Delta_2$ and no literal in $\Gamma_2$ is selected,
or
$s \approx t$ is selected,

(v) no literal in $\Gamma_1$ is selected.

■

For the sake of compatibility with the hierarchic calculus, we impose one further restriction on selection functions, namely: only non-base[1] negative literals can be selected in a clause.

Note that from the given definition of the ground superposition left rule it follows that the right premise of the rule is strictly greater than the left one. Indeed, $l \succ r$ and $s \succ t$, and $l$ is a subterm of $s$, therefore $(s[l] \not\approx t) \succ (l \approx r)$; moreover, the literal $l \approx r$ is strictly maximal in the left premise, hence any other literal in the left premise is also smaller than $s[l] \not\approx t$.

◄ DEFINITION 3.36
Ground Superposition Right

The ***ground superposition right*** rule is

$$\mathcal{I}\,\frac{\Gamma_1 \to \Delta_1, l \approx r \qquad \Gamma_2 \to \Delta_2, s[l] \approx t}{\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2, s[r] \approx t}$$

where

(i) $l \succ r$,

(ii) $s \succ t$,

(iii) $l \approx r$ is strictly maximal in $\Gamma_1 \to \Delta_1, l \approx r$,

(iv) $s[l] \approx t$ is strictly maximal in $\Gamma_2 \to \Delta_2, s[l] \approx t$,

(v) $(s[l] \approx t) \succ (l \approx r)$,

(vi) no literal in $\Gamma_1$, $\Gamma_2$ is selected.

■

Condition (v) of the above definition ensures the right premise to be strictly greater than the left one.

◄ DEFINITION 3.37
Ground Equality Resolution

The ***ground equality resolution*** rule is

$$\mathcal{I}\,\frac{\Gamma, t \approx t \to \Delta}{\Gamma \to \Delta}$$

where

(i) $t \approx t$ is maximal in $\Gamma, t \approx t \to \Delta$ and no literal in $\Gamma$ is selected, or

(ii) $t \approx t$ is selected.

■

◄ DEFINITION 3.38
Ground Equality Factoring

The ***ground equality factoring*** rule is

$$\mathcal{I}\,\frac{\Gamma \to \Delta, s \approx t, s \approx t'}{\Gamma, t \approx t' \to \Delta, s \approx t'}$$

where

[1]The notions of pure, base, non-base, and free terms are given in Definition 3.16 on page 39.

(i) $s \succ t$,

(ii) $s \approx t$ is maximal in $\Gamma \rightarrow \Delta, s \approx t, s \approx t'$,

(iii) no literal in $\Gamma$ is selected.

■

DEFINITION 3.39 ► Inference system $\mathcal{F}$

The inference rules Ground Equality Resolution, Ground Equality Factoring, and Ground Superposition Left/Right constitute the ***inference system*** $\mathcal{F}$. ■

Next, we recall the definition of redundant ground clauses and inferences for the flat case. The definition is adopted for the hierarchic case by sticking to clauses over the body signature $\Sigma'$ of a hierarchic specification HSp.

DEFINITION 3.40 ► $\mathcal{F}$-Redundant Clauses $\mathcal{F}$-Redundant Inferences

Assume HSp = (Sp, Sp′) is a hierarchic specification with the base specification Sp = $(\Sigma, \mathscr{C})$ and the body Sp′ = $(\Sigma', Ax')$.

Let $N$ be a set of ground $\Sigma'$-clauses. We define $\mathcal{R}_F^{\mathcal{F}}(N)$ to be the set of all clauses $C$ such that there exist clauses $C_1, \ldots, C_n \in N$ which are smaller than $C$ with respect to $\succ$ and entail $C$:

$$\mathcal{R}_F^{\mathcal{F}}(N) \overset{\text{def}}{=\!=} \{C \in Cl_{\Sigma'} \mid \exists C_1, \ldots, C_n \in N : C_1, \ldots, C_n \prec C, \text{ and } C_1, \ldots, C_n \models C\}.$$

We define $\mathcal{R}_I^{\mathcal{F}}(N)$ to be the set of all inferences $I$ such that either a premise of $I$ is redundant for $N$ or else there exist clauses $C_1, \ldots, C_n \in N$ which are smaller than the maximal premise of $I$ with respect to $\succ$ and entail the conclusion of $I$:

$$\mathcal{R}_I^{\mathcal{F}}(N) \overset{\text{def}}{=\!=} \{I \in \mathcal{F}(N) \mid \text{(i) } prem(I) \cap \mathcal{R}_F^{\mathcal{F}}(N) \neq \emptyset; \text{ or (ii) } \exists C_1, \ldots, C_n \in N : C_1, \ldots, C_n \prec max_{\succ}(prem(I)), \text{ and } C_1, \ldots, C_n \models concl(I)\}.$$

■

Bachmair and Ganzinger have shown that any set of ground clauses $N$, which is saturated with respect to $\mathcal{F}$ and does not contain an empty clauses, has a (Herbrand) model $\mathcal{I}_N$ (construction of $\mathcal{I}_N$ is presented in Definition 2.29). Also, they have shown that $\mathcal{R}^{\mathcal{F}} = (\mathcal{R}_I^{\mathcal{F}}, \mathcal{R}_F^{\mathcal{F}})$ is a redundancy criterion with respect to the inference system $\mathcal{F}$ and the general entailment relation $\models$, and that the calculus $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$ is refutationally complete [BG94]. We put together these results in the following lemma and theorem.

LEMMA 3.41 ► *If a set $N$ of ground clauses is saturated with respect to $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$ and does not contain an empty clause, then the candidate interpretation*[1] *$\mathcal{I}_N$ is a model of $N$.*

THEOREM 3.42 ► *The inference system $\mathcal{F}$ and the redundancy criterion $\mathcal{R}^{\mathcal{F}} = (\mathcal{R}_I^{\mathcal{F}}, \mathcal{R}_F^{\mathcal{F}})$ satisfy the following properties:*

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

*(ii) $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$ is a refutationally complete calculus.*

[1]Please, recall Definition 2.29 of a candidate interpretation $\mathcal{I}_N$, page 26.

In Section 3.4.4 we formulate a general hierarchic redundancy criterion that is obtained by lifting the ground criterion to the general case. For this reason, we shall often show different properties of general clauses/inferences by considering their sets of all simple ground instances, reducing thereby the original problem to the ground case. To this end, we shall heavily use results obtained for the flat redundancy criterion $\mathcal{R}^{\mathcal{F}}$, particularly, we shall need a further property of $\mathcal{R}^{\mathcal{F}}$, stated in Lemma 3.43, which is a generalisation of the monotonicity property (ii) in Definition 3.1 of an abstract redundancy criterion.

◄ LEMMA 3.43

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $N$ and $M$ be two sets of ground $\Sigma'$-clauses. If for every clause $C \in N$ there exist clauses $C_1, \ldots, C_n \in M$ such that $\bigcup_i C_i \models C$ and $C_1, \ldots, C_n \preceq C$, then:*

*(i)* $\mathcal{R}_F^{\mathcal{F}}(N) \subseteq \mathcal{R}_F^{\mathcal{F}}(M)$,

*(ii)* $\mathcal{R}_I^{\mathcal{F}}(N) \subseteq \mathcal{R}_I^{\mathcal{F}}(M)$.

◄ PROOF

**Statement (i)**. Consider an arbitrary clause $C' \in \mathcal{R}_F^{\mathcal{F}}(N)$. By Definition 3.40, there exist clauses $C'_1, \ldots, C'_m \in N$ such that

$$\bigcup_{1 \le i \le m} C'_i \models C',$$

and $C'_i \prec C'$ for every $i \in \{1, \ldots, m\}$. By assumption, for every such a clause $C'_i$ there exist clauses $C^i_1, \ldots, C^i_{n_i} \in M$ such that

$$\bigcup_{1 \le j \le n_i} C^i_j \models C'_i,$$

and $C^i_j \preceq C'_i$ for every $j \in \{1, \ldots, n_i\}$. By transitivity of the entailment relation $\models$ and ordering $\prec$, we obtain

$$\bigcup_{1 \le i \le m} \bigcup_{1 \le j \le n_i} C^i_j \models C',$$

and $C^i_j \prec C'$ for every $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n_i\}$. Hence, by Definition 3.40 of $\mathcal{R}_F^{\mathcal{F}}$, we have $C' \in \mathcal{R}_F^{\mathcal{F}}(M)$.

Since $C'$ has been picked from $\mathcal{R}_F^{\mathcal{F}}(N)$ arbitrarily, we conclude $\mathcal{R}_F^{\mathcal{F}}(N) \subseteq \mathcal{R}_F^{\mathcal{F}}(M)$.

**Statement (ii)**. Consider an arbitrary inference $I \in \mathcal{R}_I^{\mathcal{F}}(N)$. By the definition of $\mathcal{R}_I^{\mathcal{F}}$, either

- $C' \in \mathcal{R}_F^{\mathcal{F}}(N)$ for some $C' \in \mathit{prem}(I)$; in this case, as we have just shown, $C' \in \mathcal{R}_F^{\mathcal{F}}(M)$, thus $I \in \mathcal{R}_I^{\mathcal{F}}(M)$; or
- there exist clauses $C'_1, \ldots, C'_m \in N$ such that

  $$\bigcup_{1 \le i \le m} C'_i \models \mathit{concl}(I),$$

  and $C'_i \prec \mathit{max}_{\succ}(\mathit{prem}(I))$ for every $i \in \{1, \ldots, m\}$. By the same argumentation that we have used to show (i), we assert that there exist clauses $C^i_1, \ldots, C^i_{n_i} \in M$, where $i$ ranges from 1 to $m$, such that

  $$\bigcup_{1 \le i \le m} \bigcup_{1 \le j \le n_i} C^i_j \models \mathit{concl}(I)$$

and $C^i_j \prec max(prem(I))$ for every $i \in \{1,\ldots,m\}$ and $j \in \{1,\ldots,n_i\}$. Therefore, by Definition 3.40 of $\mathcal{R}^{\mathcal{F}}_I$, we have $I \in \mathcal{R}^{\mathcal{F}}_I(M)$.

Since $I$ has been picked from $\mathcal{R}^{\mathcal{F}}_I(N)$ arbitrarily, and for all possible cases of $I$ we have shown that $I \in \mathcal{R}^{\mathcal{F}}_I(M)$, it follows $\mathcal{R}^{\mathcal{F}}_I(N) \subseteq \mathcal{R}^{\mathcal{F}}_I(M)$. ■

The $\mathcal{F}$-inference rules are defined such that the conclusion of an inference is smaller than its largest premise with respect to any reduction ordering [BG90, BG91, BG94]. In the context of a hierarchic specification, this property has a stronger formulation: the non-base part[1] of the conclusion of an $\mathcal{F}$-inference with non-base premises is smaller than the non-base part of its largest premise. The stronger property has a useful implication that any ground non-base clause $C = \Pi,\Gamma \to \Upsilon,\Delta$, whose non-base part $\Gamma \to \Delta$ is smaller than or equal to the non-base part $\Gamma' \to \Delta'$ of the conclusion $D = concl(I) = \Pi',\Gamma' \to \Upsilon',\Delta'$ of an $\mathcal{F}$-inference $I$, is smaller than the inference's largest premise $max_{\succ}(prem(I))$, *even if* the whole clause $C$ is larger than the whole conclusion $D$ (which is the case if $\Gamma \to \Delta = \Gamma' \to \Delta'$ and $\Pi \to \Upsilon \succ \Pi' \to \Upsilon'$), with respect to any reduction ordering $\succ$, under which all base operator symbols $\Omega$ are smaller than non-base ones $\Omega' \setminus \Omega$. We will exploit this property in Section 3.3 when discussing the applicability of the hierarchic reduction rules.

PROPOSITION 3.44 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp},\mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma,\mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*The non-base part of the maximal premise of an $\mathcal{F}$-inference with non-base premises is greater than the non-base part of the inference's conclusion.*

PROOF ► First, let us make the following very important observation. Let $C$ be an arbitrary non-base clause. If some literal $L$ is (strictly) maximal in $C$ or selected, then $L$ is non-base: first, $\mathcal{F}$ admits selection only of *non-base* negative literals, and, second, any ground non-base literal is strictly greater than any base one.

Assume $I$ is a ***Ground Equality Resolution*** inference (Definition 3.37)

$$\mathcal{I}\frac{\Pi,\Gamma,t\approx t \to \Upsilon,\Delta}{\Pi,\Gamma \to \Upsilon,\Delta}$$

where $\Pi$ and $\Upsilon$ consist of base atoms, and $\Gamma$ and $\Delta$ of non-base ones. The atom $t \approx t$ is non-base. Obviously, $\Gamma, t\approx t \to \Delta \succ \Gamma \to \Delta$.

Assume $I$ is a ***Ground Equality Factoring*** inference (Definition 3.38)

$$\mathcal{I}\frac{\Pi,\Gamma \to \Upsilon,\Delta,s\approx t,s\approx t'}{\Pi,\Gamma,t\approx t' \to \Upsilon,\Delta,s\approx t'}$$

where $\Pi$ and $\Upsilon$ consist of base atoms, and $\Gamma$ and $\Delta$ of non-base ones. The atoms $s \approx t$, $s \approx t'$ are non-base, whereas $t \approx t'$ may be base. Since $s \approx t$ is maximal in the premise, and $s \succ t$, we conclude $t \succeq t'$. Consequently,

$$s\approx t \succeq s\approx t' \succ t \not\approx t'.$$

[1]Recall, that the (non-)base part of a clause is a clause consisting of all (non-)base literals occurring in the given clause. For an abstracted clause, the base and the non-base parts coincide with the base and free parts of the clause, respectively. For a simple instance of an abstracted clause, the base part consists of all instantiated base literals, and the non-base part consists of all instantiated free literals of the given clause.

Therefore, $s \approx t$ is greater than or equal to any literal in the conclusion of $I$. Besides, $s \approx t$ occurs in the premise more often than in the conclusion. Hence,

$$\begin{aligned}\Gamma \to \Delta, s \approx t, s \approx t' &\succ \Gamma, t \approx t' \to \Delta, s \approx t' \\ &\succ \Gamma \to \Delta, s \approx t'.\end{aligned}$$

Assume $I$ is a ***Ground Superposition Left*** inference (Definition 3.35)

$$\mathcal{I}\frac{\Pi_1, \Gamma_1 \to \Upsilon_1, \Delta_1, l \approx r \qquad \Pi_2, s[l] \approx t, \Gamma_2 \to \Upsilon_2, \Delta_2}{\Pi_1, \Pi_2, s[r] \approx t, \Gamma_1, \Gamma_2 \to \Upsilon_1, \Upsilon_2, \Delta_1, \Delta_2}$$

where $\Pi_i$ and $\Upsilon_i$ consist of base atoms, and $\Gamma_i$ and $\Delta_i$ of non-base ones, for each $i \in \{1,2\}$. The atoms $l \approx r$ and $s[l] \approx t$ are non-base, whereas $s[r] \approx t$ may be base. Since $l$ is a subterm of $s$, we may infer that $s \not\approx t \succ l \approx r$. From the fact that $l \approx r$ is strictly maximal in the left premise, we conclude that $s \not\approx t$ is greater than any literal in the left premise. Hence,

$$\begin{aligned}s[l] \approx t, \Gamma_2 \to \Delta_2 &\succ s[r] \approx t, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2 \\ &\succ \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2\end{aligned}$$

Assume $I$ is a ***Ground Superposition Right*** inference (Definition 3.36)

$$\mathcal{I}\frac{\Pi_1, \Gamma_1 \to \Upsilon_1, \Delta_1, l \approx r \qquad \Pi_2, \Gamma_2 \to \Upsilon_2, \Delta_2, s[l] \approx t}{\Pi_1, \Pi_2, \Gamma_1, \Gamma_2 \to \Upsilon_1, \Upsilon_2, \Delta_1, \Delta_2, s[r] \approx t}$$

where $\Pi_i$ and $\Upsilon_i$ consist of base atoms, and $\Gamma_i$ and $\Delta_i$ of non-base ones, for each $i \in \{1,2\}$. The atoms $l \approx r$ and $s[l] \approx t$ are non-base, whereas $s[r] \approx t$ may be base. Since $l \succ r$, it follows that $s[l] \approx t \succ s[r] \approx t$. Since the involved literals are strictly maximal in the respective clauses, and $s[l] \approx t \succ l \approx r$, it follows that $s[l] \approx t$ is greater than any literal in the left premise. Hence,

$$\begin{aligned}\Gamma_2 \to \Delta_2, s[l] \approx t &\succ \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2, s[r] \approx t \\ &\succ \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2\end{aligned}$$

Thus, the non-base part of the maximal premise of any $\mathcal{F}$-inference with non-base premises is greater than the non-base part of its conclusion. ■

### 3.4.3 Reduced Instances and Extended Clause Sets

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

In the overview to the present section, we have reduced the problem of SUP(T) completeness to hierarchic model existence of the ground clause set $N' = sgi(N) \cup \tilde{\mathsf{E}}_{\mathcal{A}} \cup \tilde{\mathsf{D}}_{\mathcal{A}}$, where $\mathcal{A} \in \mathscr{C}$ is a base algebra, $\tilde{\mathsf{E}}_{\mathcal{A}}$ a set of positive unit clauses, each of which consists of an equation between two distinct ground base terms that are equal under the base algebra $\mathcal{A}$, and $\tilde{\mathsf{D}}_{\mathcal{A}}$ is a set of negative unit clauses, each of which consists of a disequation between two ground base terms that have different interpretation under the $\mathcal{A}$.

Evidently, the set $N'$ might contain many clauses that are redundant with respect to the ground redundancy criterion $\mathcal{R}^{\mathcal{F}}$. In order to identify those redundant clauses in $N'$, we define the sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$ of irredundant clauses contained in $\tilde{\mathsf{E}}_{\mathcal{A}}$ and $\tilde{\mathsf{D}}_{\mathcal{A}}$ , such that $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \models \tilde{\mathsf{E}}_{\mathcal{A}} \cup \tilde{\mathsf{D}}_{\mathcal{A}}$, Definition 3.49.

In the presence of clauses in $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, a considerable bunch of clauses in $sgi(N)$ may be also redundant. Thus, any ground base clause $C \notin \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ that is satisfiable under some base algebra $\mathcal{A} \in \mathscr{C}$ is a logical consequence of finitely many clauses from $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, each of which is smaller than $C$, Lemma 3.51, therefore any such clause $C$ is redundant.

A simple ground substitution $\sigma$ allows arbitrary ground base terms to appear in its image $im(\sigma)$. Given a base algebra $\mathcal{A}$, any ground base term $t$ represents an equivalence class $[t]_{\mathcal{A}}$ consisting of ground base terms that have the same interpretation in $\mathcal{A}$ as $t$, Definition 3.45. Each equivalence class $[t]_{\mathcal{A}}$ has a minimal element with respect to $\succ$, which we denote by $m^{\succ}_{\mathcal{A}}(t)$. Any equation between two representatives of such equivalence class is entailed by a finite subset of $\mathsf{E}_{\mathcal{A}}$ (clauses in $\mathsf{D}_{\mathcal{A}}$ play a role only for validating *disequations*). Then we can show that every simple ground instance $C\sigma$ of a non-base clause $C \in N$ is implied by a finite subset of $\mathsf{E}_{\mathcal{A}}$ and another simple ground instance $C\sigma'$ of $C$ smaller than (or equal to) $C$, Proposition 3.59, where $\sigma'$ is a simple ground substitution obtained from $\sigma$ by replacing every base (sub)term $t$ in its image $im(\sigma)$ with the minimal element $m^{\succ}_{\mathcal{A}}(t)$ of the equivalence class $[t]_{\mathcal{A}}$, Definition 3.54. We use notation $sgi_{\mathcal{A}}(C)$ to denote the set of all simple ground instances $C\sigma'$ of a clause $C$, where every simple grounding substitution $\sigma'$ is reduced in the above sense; and we write $N_{\mathcal{A}}$ to denote the clause set $sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, Definition 3.56. Thus, each clause $C$ in $sgi(N) \setminus N_{\mathcal{A}}$ is entailed by finitely many clauses in $N_{\mathcal{A}}$, which all are smaller than $C$, Lemma 3.61, hence redundant.

The correctness of term/substitution reduction discussed just above is shown using the apparatus of the term rewrite systems theory. Particularly for this purpose, we introduce rewrite systems $\tilde{R}^{\approx}_{\mathcal{A}}$ and $R^{\approx}_{\mathcal{A}}$ containing all rewrite rules $l \to r$ for each of which there is a clause $(\to l \approx r)$ in $\tilde{\mathsf{E}}_{\mathcal{A}}$ or in $\mathsf{E}_{\mathcal{A}}$, respectively, Definition 3.46.

The current subsection is devoted to a formal definition of the sets $R^{\approx}_{\mathcal{A}}$, $\mathsf{E}_{\mathcal{A}}$, $\mathsf{D}_{\mathcal{A}}$, $sgi_{\mathcal{A}}(N)$, $N_{\mathcal{A}}$, and proving the properties briefly discussed above. The properties will become *particularly important* in the section on lifting and saturation, where they are used to show that all *non-liftable* ground inferences with premises in $N_{\mathcal{A}}$ are actually redundant, which is indispensable to ensure the mutual saturation condition[1].

DEFINITION 3.45 ► $[t]_{\mathcal{A}}$ $m^{\succ}_{\mathcal{A}}(t)$ Assume $\mathsf{Sp} = (\Sigma, \mathscr{C})$ is a base specification, with a base signature $\Sigma = (\mathcal{S}, \Omega)$ and a class of base algebras $\mathscr{C}$.

Let $\mathcal{A} \in \mathscr{C}$ be a base algebra. For every ground base term $t \in T_\Omega$ we define the ***equivalence class*** $[t]_{\mathcal{A}}$ ***of*** $t$ ***in*** $\mathcal{A}$ as the set consisting of all ground base terms $s \in T_\Omega$ that have the same interpretation in $\mathcal{A}$ as $t$:

$$[t]_{\mathcal{A}} \overset{\text{def}}{=} \{s \in T_\Omega \mid s_{\mathcal{A}} = t_{\mathcal{A}}\}.$$

Assume $\succ$ is a reduction ordering total on ground $\Sigma$-terms. We write $m^{\succ}_{\mathcal{A}}(t)$ to denote the smallest term in $[t]_{\mathcal{A}}$ with respect to the ordering ordering $\succ$:

$$m^{\succ}_{\mathcal{A}}(t) \overset{\text{def}}{=} min_{\succ}([t]_{\mathcal{A}}).$$

[1]The mutual saturation condition is discussed in the overview to Section 3.4, pages 50-51.

■

Since $\succ$ is a reduction ordering total on ground terms, for any equivalence class $[t]_{\mathcal{A}}$ the term $m^{\succ}_{\mathcal{A}}(t)$ is unique.

Next we define two rewrite systems which given a base algebra $\mathcal{A} \in \mathscr{C}$ induce the same equality relation on ground base terms as $\mathcal{A}$. The rewrite systems' properties discussed below will be of a particular use when presenting the hierarchic redundancy criterion.

◄ DEFINITION 3.46 Rewrite System $\tilde{R}^{\approx}_{\mathcal{A}}$ Rewrite System $R^{\approx}_{\mathcal{A}}$

Assume $\mathsf{Sp} = (\Sigma, \mathscr{C})$ is a base specification, with a base signature $\Sigma = (\mathcal{S}, \Omega)$ and a class of base algebras $\mathscr{C}$. Let $\succ$ be a reduction ordering total on ground $\Sigma$-terms.

Let $\mathcal{A} \in \mathscr{C}$ be a base algebra. We define a ***rewrite system*** $\tilde{R}^{\approx}_{\mathcal{A}}$ as the set of all rewrite rules between every ground base term $t \in T_\Omega$ and the smallest term in its equivalence class $[t]_{\mathcal{A}}$ different from $t$:

$$\tilde{R}^{\approx}_{\mathcal{A}} \stackrel{\text{def}}{=} \{t \to m^{\succ}_{\mathcal{A}}(t) \mid t \in T_\Omega, t \neq m^{\succ}_{\mathcal{A}}(t)\}.$$

We define a ***rewrite system*** $R^{\approx}_{\mathcal{A}}$ to be the reduced subset of $\tilde{R}^{\approx}_{\mathcal{A}}$, that is, the set of all rules $(l \to r) \in \tilde{R}^{\approx}_{\mathcal{A}}$ such that $l$ is not reducible by $\tilde{R}^{\approx}_{\mathcal{A}} \setminus \{l \to r\}$:

$$R^{\approx}_{\mathcal{A}} \stackrel{\text{def}}{=} \{(l \to r) \in \tilde{R}^{\approx}_{\mathcal{A}} \mid l = l{\downarrow}_{\tilde{R}^{\approx}_{\mathcal{A}} \setminus \{l \to r\}}\}.$$

■

Obviously, $\tilde{R}^{\approx}_{\mathcal{A}}$ is terminating, confluent, and right-reduced. Next we show that $R^{\approx}_{\mathcal{A}}$ induces the same equality relation on ground base terms as $\tilde{R}^{\approx}_{\mathcal{A}}$.

◄ LEMMA 3.47

*Assume* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *is a base specification, with a base signature* $\Sigma = (\mathcal{S}, \Omega)$ *and a class of base algebras* $\mathscr{C}$*.*

*Let* $\mathcal{A} \in \mathscr{C}$ *be a base algebra. The rewrite systems* $\tilde{R}^{\approx}_{\mathcal{A}}$ *and* $R^{\approx}_{\mathcal{A}}$ *define the same set of normal forms.*

◄ PROOF

By contradiction. Let $t$ be an arbitrary term in $T_\Omega$. Let

$$s' = t{\downarrow}_{\tilde{R}^{\approx}_{\mathcal{A}}} \text{ and } s = t{\downarrow}_{R^{\approx}_{\mathcal{A}}}$$

be the normal forms of $t$ with respect to $\tilde{R}^{\approx}_{\mathcal{A}}$ and $R^{\approx}_{\mathcal{A}}$, respectively. Suppose $s \neq s'$.

Since $R^{\approx}_{\mathcal{A}} \subseteq \tilde{R}^{\approx}_{\mathcal{A}}$ and $\tilde{R}^{\approx}_{\mathcal{A}}$ is confluent, we have

$$s \to^*_{\tilde{R}^{\approx}_{\mathcal{A}}} s'' \text{ and } s' \to^*_{\tilde{R}^{\approx}_{\mathcal{A}}} s'',$$

for some $s'' \in T_\Omega$. Since $s'$ is the normal form of $t$ with respect to $\tilde{R}^{\approx}_{\mathcal{A}}$, we have actually $s'' = s'$. Moreover, by assumption, $s \neq s'$, hence

$$s \to^+_{\tilde{R}^{\approx}_{\mathcal{A}}} s'.$$

Let $\{s_1, \ldots, s_n\}$ be the set of all terms $s_i \in T_\Omega$ that are reachable by a single $\tilde{R}^{\approx}_{\mathcal{A}}$-rewrite step from $s$ in a rewriting $s \to^+_{\tilde{R}^{\approx}_{\mathcal{A}}} s'$. In other words, any rewriting $s \to^+_{\tilde{R}^{\approx}_{\mathcal{A}}} s'$ has form

$$s \to_{\tilde{R}^{\approx}_{\mathcal{A}}} s_i \to^*_{\tilde{R}^{\approx}_{\mathcal{A}}} s',$$

for some $i \in \{1, \ldots, n\}$. Let

$$R = \{l_1 \to r_1, \ldots, l_k \to r_k\}$$

be the set of $\tilde{R}^{\approx}_{\mathcal{A}}$-rewrite rules, by means of which a rewrite step $s \to_{\tilde{R}^{\approx}_{\mathcal{A}}} s_i$ is possible, $1 \leq i \leq n$. Obviously, $R \subseteq \tilde{R}^{\approx}_{\mathcal{A}}$.

Let $\succ$ be the reduction ordering total on ground terms underlying $\tilde{R}^{\approx}_{\mathcal{A}}$ and $R^{\approx}_{\mathcal{A}}$. Let $l \to r$ be the rule in $R$, whose left-hand side $l$ is the smallest with respect to $\succ$ among all $l_j$'s in $R$, $1 \leq j \leq k$. Note that $l \to r$ is unique (meaning it is the only rule in $R$ whose left-hand-side is the $l$), because $r = m^{\succ}_{\mathcal{A}}(l)$ by definition of $\tilde{R}^{\approx}_{\mathcal{A}}$, and $\succ$ is total on ground terms. Moreover, $l \to r$ is the only rule in $\tilde{R}^{\approx}_{\mathcal{A}}$, that can be used to rewrite $l$. Indeed, suppose, for the sake of contradiction, that there is another rewrite rule $(l' \to r') \in \tilde{R}^{\approx}_{\mathcal{A}}$ such that $l' \neq l$ and

$$l \to_{\{l' \to r'\}} r'',$$

for some $r'' \in T_{\Omega}$. Note that $l' \to r'$ must also be contained in $R$, because it can be used to rewrite $s = s[l']$ to $s[r']$. Since $l \neq l'$, the term $l'$ occurs in $l$ at a non-top position $p \in \rho(l)$, $p \neq \varepsilon$. Therefore, $l'$ is a strict subterm of $l$, hence, by the subterm property of $\succ$, we obtain $l' \prec l$, which contradicts the minimality of $l$ within the rules of $R$.

Since $l \to r$ is the only rule in $\tilde{R}^{\approx}_{\mathcal{A}}$, by means of which $l$ can be rewritten, it follows that $(l \to r) \in R^{\approx}_{\mathcal{A}}$. Consequently, $l \to r$ can be applied to rewrite $s$ even further, contradicting the assumption that $s$ is the normal form of $t$ with respect to $R^{\approx}_{\mathcal{A}}$. ■

Lemma 3.47 and the facts that $\tilde{R}^{\approx}_{\mathcal{A}}$ is terminating, confluent, and right-reduced, and $R^{\approx}_{\mathcal{A}} \subseteq \tilde{R}^{\approx}_{\mathcal{A}}$ directly imply the following corollary.

COROLLARY 3.48 ► *Assume* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *is a base specification, with a base signature* $\Sigma = (\mathcal{S}, \Omega)$ *and a class of base algebras* $\mathscr{C}$.

*Let* $\mathcal{A} \in \mathscr{C}$ *be a base algebra.* $R^{\approx}_{\mathcal{A}}$ *is convergent and right-reduced.* $\tilde{R}^{\approx}_{\mathcal{A}}$ *and* $R^{\approx}_{\mathcal{A}}$ *induce the same equality relation on ground base terms.*

Note that any derivation $t \to^{*}_{R^{\approx}_{\mathcal{A}}} s$ is finite for any ground $\Sigma'$-term $t \in T_{\Omega'}$, where $t$ is not necessarily base. Moreover, if $t$ is a ground base term, then

$$m^{\succ}_{\mathcal{A}}(t) = t{\downarrow}_{R^{\approx}_{\mathcal{A}}}.$$

From definition of $R^{\approx}_{\mathcal{A}}$ and Lemma 3.47 it follows also that $\mathcal{A}(t) = \mathcal{A}(s)$ if and only if $t{\downarrow}_{R^{\approx}_{\mathcal{A}}} = s{\downarrow}_{R^{\approx}_{\mathcal{A}}}$, for any ground base terms $t$ and $s$.

DEFINITION 3.49 ► Clause Set $\mathsf{E}_{\mathcal{A}}$ Clause Set $\mathsf{D}_{\mathcal{A}}$

Assume $\mathsf{Sp} = (\Sigma, \mathscr{C})$ is a base specification, with a base signature $\Sigma = (\mathcal{S}, \Omega)$ and a class of base algebras $\mathscr{C}$.

Let $\mathcal{A} \in \mathscr{C}$ be a base algebra. We identify $\mathsf{E}_{\mathcal{A}}$ with the set of all positive unit clauses whose literals are the rewrite rules (equations) in the rewrite system[1] $R^{\approx}_{\mathcal{A}}$:

$$\mathsf{E}_{\mathcal{A}} \stackrel{\text{def}}{=} \{(\to t \approx t') \mid (t \to t') \in R^{\approx}_{\mathcal{A}}\}.$$

We put $\mathsf{D}_{\mathcal{A}}$ to be the set of all negative unit clauses $(t \approx t' \to)$, with $t$ and $t'$ being two distinct ground base terms reduced with respect to the rewrite system $R^{\approx}_{\mathcal{A}}$:

$$\mathsf{D}_{\mathcal{A}} \stackrel{\text{def}}{=} \{(t \approx t' \to) \mid t, t' \in T_{\Omega},\, t \neq t',\, t = t{\downarrow}_{R^{\approx}_{\mathcal{A}}},\, t' = t'{\downarrow}_{R^{\approx}_{\mathcal{A}}}\}.$$

■

[1]Please, recall Definition 3.46 of the rewrite system $R^{\approx}_{\mathcal{A}}$, page 61.

Note that since for any ground base term $t$ it holds that $m^{\succ}_{\mathcal{A}}(t) = t{\downarrow}_{R^{\approx}_{\mathcal{A}}}$, the clause set $\mathsf{D}_{\mathcal{A}}$ can be equivalently seen as follows:

$$\mathsf{D}_{\mathcal{A}} = \{(t \approx t' \rightarrow) \mid t, t' \in T_{\Omega},\ t \neq t',\ t = m^{\succ}_{\mathcal{A}}(t),\ t' = m^{\succ}_{\mathcal{A}}(t')\}.$$

Our next goal is to show that any $\mathscr{C}$-consistent ground base clause $C$ not contained in the clause set $\mathsf{E}_{\mathcal{A}}$ is a logical consequence of finitely many clauses from $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, each of which is smaller than $C$, Lemma 3.51. As a preliminary step, we prove in Proposition 3.50 that an equation between two ground base terms is $\mathscr{C}$-consistent if and only if it is entailed by finitely many clauses from $\mathsf{E}_{\mathcal{A}}$, each of which is smaller than the equation. In the proofs of Lemma 3.51 and Proposition 3.50 we *exactly* identify clauses in $\mathsf{E}_{\mathcal{A}}$ (and $\mathsf{D}_{\mathcal{A}}$) that satisfy the stated properties. For an illustration, see Example 3.53 on page 66.

◄ PROPOSITION 3.50

*Assume* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *is a base specification, with a base signature* $\Sigma = (\mathcal{S}, \Omega)$ *and a class of base algebras* $\mathscr{C}$. *Let* $\succ$ *be a reduction ordering total on ground* $\Sigma$*-terms.*

*Let* $\mathcal{A} \in \mathscr{C}$ *be a base algebra,* $t, t' \in T_{\Omega}$ *two ground base terms. If the clause* $(\rightarrow t \approx t')$ *is not contained in* $\mathsf{E}_{\mathcal{A}}$*, then* $t \approx t'$ *is true in* $\mathcal{A}$ *if and only if there exist finitely many clauses* $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}}$*, for some* $n \geq 0$*, such that* $C_1, \ldots, C_n \models t \approx t'$*, and* $C_i \prec \{t \approx t'\}$*, for every* $1 \leq i \leq n$.

◄ PROOF

**The "⇐" direction**. Assume $C_1, \ldots, C_n \models t \approx t'$ and $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}}$ for some base algebra $\mathcal{A} \in \mathscr{C}$. Then we have:

$$\begin{array}{ll} \mathcal{A} \models \mathsf{E}_{\mathcal{A}} & \text{// by def. of } \mathsf{E}_{\mathcal{A}} \\ \quad \models C_1, \ldots, C_n & \text{// as } C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}} \\ \Rightarrow \mathcal{A} \models t \approx t' & \text{// as } C_1, \ldots, C_n \models t \approx t' \text{ by assum.} \end{array}$$

**The "⇒" direction**. Assume $\mathcal{A} \models t \approx t'$ and $(\rightarrow t \approx t') \notin \mathsf{E}_{\mathcal{A}}$, for two arbitrary ground base terms $t, t' \in T_{\Omega}$ and some base algebra $\mathcal{A} \in \mathscr{C}$. As $t, t' \in T_{\Omega}$ and $\mathcal{A} \models t \approx t'$, the terms have the same normal form with respect to the rewrite system $R^{\approx}_{\mathcal{A}}$, say $s \in T_{\Omega}$, that is the minimal element in the equivalence class $[t]_{\mathcal{A}} = [s]_{\mathcal{A}} = [t']_{\mathcal{A}}$ the terms belong to:

$$t{\downarrow}_{R^{\approx}_{\mathcal{A}}} = m^{\succ}_{\mathcal{A}}(t) = s = m^{\succ}_{\mathcal{A}}(t') = t'{\downarrow}_{R^{\approx}_{\mathcal{A}}}.$$

Hence, there exist rewritings $t \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s$ and $t' \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s$ from the terms to their normal form $s$. Recall that any possible rewriting $t \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s$ from $t$ to $s$ (and $t' \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s$ from $t'$ to $s$) is finite. We prove the statement of the proposition by induction on the sum $|t \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s| + |t' \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s|$ of the lengths of the rewritings.

*Induction base.* Assume

$$|t \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s| + |t' \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s| = 0,$$

then $t = s = t'$ and $t \approx t'$ is a tautology.

*Induction hypothesis.* Assume lemma holds for any $t, t' \in T_{\Omega}$ such that

$$|t \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s| + |t' \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s| \leq k,$$

for some $k \geq 0$.

*Induction step.* Assume

$$|t \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s| + |t' \rightarrow^{*}_{R^{\approx}_{\mathcal{A}}} s| = k + 1.$$

Without loss of generality, represent the rewriting $t \to^*_{R^{\approx}_{\mathcal{A}}} s$ as follows:

$$t \to_{R^{\approx}_{\mathcal{A}}} s' \to^*_{R^{\approx}_{\mathcal{A}}} s,$$

for some ground base term $s' \in T_\Omega$ obtained from $t$ by a single application of some rule $(l \to r) \in R^{\approx}_{\mathcal{A}}$, i.e.

$$t \to_{\{l \to r\}} s'.$$

By Definition 3.49, the clause $(\to l \approx r)$ is in $\mathsf{E}_{\mathcal{A}}$. Evidently,

$$(\to l \approx r) \models t \approx s'.$$

Also, as $(l \to r) \in R^{\approx}_{\mathcal{A}}$, we know that $\mathcal{A} \models t \approx s'$, consequently $\mathcal{A} \models s' \approx t'$, because $\mathcal{A} \models t \approx t'$ by assumption. Since

$$|s' \to^*_{R^{\approx}_{\mathcal{A}}} s| + |t' \to^*_{R^{\approx}_{\mathcal{A}}} s| = k,$$

it follows by induction hypothesis that there exist clauses $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}}$ such that

$$C_1, \ldots, C_n \models s' \approx t'$$

and $C_i \prec \{s' \approx t'\}$, for every $1 \le i \le n$. Consequently, $C_i \prec \{t \approx t'\}$ as $t \succ s'$, for every $1 \le i \le n$. Since $r = m^{\succ}_{\mathcal{A}}(l)$ (by Definition 3.46 of $R^{\approx}_{\mathcal{A}}$) and $\prec$ is a reduction ordering, hence has the subterm property, we know that $t \succeq l \succ r$. Consider the following cases regarding the order of $t$ and $l$:

- if $t \succ l$, then evidently $\{t \approx t'\} \succ \{l \approx r\}$;
- if $t \simeq l$, implying $t = l$, then $s' = r$. As $r$ is irreducible in $R^{\approx}_{\mathcal{A}}$, we have $s' = s$, i.e. $|t \to^*_{R^{\approx}_{\mathcal{A}}} s| = 1$. If $|t' \to^*_{R^{\approx}_{\mathcal{A}}} s| = 0$, then $t' = s$ and $(t \approx t') = (l \approx r)$, which contradicts the assumption that $(\to t \approx t')$ is not contained in $\mathsf{E}_{\mathcal{A}}$. Therefore, $|t' \to^*_{R^{\approx}_{\mathcal{A}}} s| > 0$ and, consequently, $t' \succ s = r$, hence $\{t \approx t'\} \succ \{l \approx r\}$.

Putting $C_{n+1} = (\to l \approx r)$, we obtain

$$C_1, \ldots, C_n, C_{n+1} \models t \approx s', s' \approx t',$$

consequently

$$C_1, \ldots, C_n, C_{n+1} \models t \approx t',$$

where $C_i \in \mathsf{E}_{\mathcal{A}}$ and $\{t \approx t'\} \succ C_i$, for each $i \in \{1, \ldots, n+1\}$. ■

LEMMA 3.51 ► *Assume* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *is a base specification, with a base signature* $\Sigma = (\mathcal{S}, \Omega)$ *and a class of base algebras* $\mathscr{C}$. *Let* $\succ$ *be a reduction ordering total on ground* $\Sigma$*-terms.*

*Let* $\mathcal{A} \in \mathscr{C}$ *be a base algebra, and* $C$ *a ground base clause not contained in* $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$. *Then* $\mathcal{A} \models C$ *if and only if there exist finitely many clauses* $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, *for some* $n \ge 0$, *such that* $C_1, \ldots, C_n \models C$, *and* $C_i \prec C$, *for every* $i \in \{1, \ldots, n\}$.

PROOF ► **The "⇐" direction**. The proof for the "if"-statement is the same as that for Proposition 3.50 if the equation $t \approx t'$ replaced with the clause $C$.

**The "⇒" direction**. A ground clause $C$ is true under $\mathcal{A}$ if and only if at least one of its literals is true under $\mathcal{A}$. Assume $L = t \mathrel{\dot\approx} s$ is such a literal, where $\dot\approx \in \{\approx, \not\approx\}$. If $\{t \mathrel{\dot\approx} s\} \in \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, then $C$ is not a unit clause (as for otherwise $C$ would be contained in $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$) and the statement evidently holds in this case. For the rest of the proof assume $\{t \mathrel{\dot\approx} s\} \notin \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$. Without restriction of generality, assume $t \succ s$ (the case $t = s$ is trivial). There are two possible cases regarding the sign of $L$:

– $L$ is positive, i.e. $L = t \approx s$. By Proposition 3.50, we know that there exist finitely many clauses $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}}$ such that $C_1, \ldots, C_n \models t \approx s$ and $C_i \prec \{t \approx s\}$, for every $i \in \{1, \ldots, n\}$.

– $L$ is negative, i.e. $L = t \not\approx s$. Then $\mathcal{A} \models L$ if and only if $\mathcal{A}(t) \neq \mathcal{A}(s)$. Let us denote by $t'$ and $s'$ the minimal elements in the equivalence classes $[t]_{\mathcal{A}}$ and $[s]_{\mathcal{A}}$ of $t$ and $s$, respectively:

$$t' = m^{\succ}_{\mathcal{A}}(t),$$
$$s' = m^{\succ}_{\mathcal{A}}(s).$$

Obviously, the equations $t \approx t'$ and $s \approx s'$ are true in $\mathcal{A}$. From Proposition 3.50 we know that there exist finitely many clauses $C_1, \ldots, C_{n'}$ and $D_1, \ldots, D_{m'}$ in $\mathsf{E}_{\mathcal{A}}$, for some $n', m' \geq 0$, such that

$$C_1, \ldots, C_{n'} \models t \approx t',$$
$$D_1, \ldots, D_{m'} \models s \approx s',$$

and

$$C_i \preceq \{t \approx t'\},$$
$$D_j \preceq \{s \approx s'\},$$

for every $i \in \{1, \ldots, n'\}$ and $j \in \{1, \ldots, m'\}$.
Put $C' = \{t' \not\approx s'\}$. According to Definition 3.49, the negative unit clause $C'$ is contained in $\mathsf{D}_{\mathcal{A}}$. Thus, we have

$$\begin{aligned} C_1, \ldots, C_{n'}, D_1, \ldots, D_{m'}, C' &\models t \approx t', s \approx s', t' \not\approx s' \\ &\models t \not\approx s \\ &= L. \end{aligned}$$

Now we are to to show that all clauses $C_i$'s, $D_j$'s, and $C'$ are smaller than the disequation $t \not\approx s$, which in its turn implies that they are smaller than the clause $C$ as required by the lemma.

First, we compare disequation $t \not\approx s$ with the clause $C' = t' \not\approx s'$. Since by assumption $\{t \not\approx s\} \notin \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, and $C' \in \mathsf{D}_{\mathcal{A}}$, we conclude

$$\{t' \not\approx s'\} \neq \{t \not\approx s\}.$$

On other hand, from the fact that $t' = m^{\succ}_{\mathcal{A}}(t)$ and $s' = m^{\succ}_{\mathcal{A}}(s)$ it follows $t' \preceq t$ and $s' \preceq s$, consequently

$$\{t' \not\approx s'\} \preceq \{t \not\approx s\},$$

which in conjunction with the previous observation yields

$$\{t' \not\approx s'\} \prec \{t \not\approx s\}.$$

Second, we compare $t \not\approx s$ with the clauses $C_i$'s and $D_j$'s. From the fact $s' \preceq s$ and assumption $s \prec t$, it follows

$$\{s \approx s'\} \prec \{t \not\approx s\}.$$

Since $D_j \preceq \{s \approx s'\}$, we obtain $D_j \prec \{t \not\approx s\}$, for every $j \in \{1, \ldots, m'\}$.

Consider a clause $C_i$, for an arbitrary $i \in \{1, \ldots, n'\}$. Let us represent it as $C_i = (\rightarrow l_i \approx r_i)$. To compare $C_i$ and $\{t \not\approx s\}$ we have to compare multisets $\{l_i, r_i\}$ and $\{t, t, s, s\}$ in the multiset extension of $\succ$. The facts $C_i \preceq \{t \approx t'\}$ and $t' \preceq t$ imply

$t \succeq l_i \succ r_i$, yielding

$$\{l_i, r_i\} \prec_{\text{mul}} \{t, t, s, s\}.$$

Thus, $C_i \prec \{t \not\approx s\}$, for every $i \in \{1, \ldots, n'\}$.

Putting $n = n' + m' + 1$, $C_{n'+j} = D_j$ for every $j \in \{1, \ldots, m'\}$, and $C_n = C'$, we get

$$C_1, \ldots, C_n \models C$$

and $C_i \prec C$, for every $i \in \{1, \ldots, n\}$ and some finite $n \geq 0$, where $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$.

Each case considered satisfies the lemma's statement, and we are done. ■

From Lemma 3.51 it follows that if a ground base clause $C$ is not entailed by a base algebra $\mathcal{A} \in \mathscr{C}$, then there exist finitely many ground base clauses $D_1, \ldots, D_n \in \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, such that the conjunction of $C$ and $D_1, \ldots, D_n$ is unsatisfiable (in any algebra):

$$C, D_1, \ldots, D_n \models \bot.$$

Indeed, if $\mathcal{A} \not\models C$, then $\mathcal{A} \models \neg C = \bigwedge_{i=1}^{m} \overline{L}_i$, where $L_1, \ldots, L_m$ are all literals of the clause $C$, for some $m \geq 0$. Put $C_i = \{\overline{L}_i\}$, for every $i \in \{1, \ldots, m\}$, that is, let every $C_i$ be a unit clause, whose only literal is the $i$-th literal of $C$ negated. Thus, $\mathcal{A} \models C_1, \ldots, C_m$. From Lemma 3.51 we know there exist finitely many clauses $C_1^i, \ldots, C_{n_i}^i \in \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ such that all $C_j^i \preceq C_i$, and $C_1^i, \ldots, C_{n_i}^i \models C_i$, for every $i \in \{1, \ldots, m\}$. Letting $\{D_1, \ldots, D_n\} = \{C_1^1, \ldots, C_{n_m}^m\}$, we obtain $D_1, \ldots, D_n \models \bigwedge_{i=1}^{m} \overline{L}_i = \neg C$, which holds, in turn, iff $C, D_1, \ldots, D_n \models \bot$. Without loss of generality, assume $L_j = (s \mathrel{\dot{\approx}} t)$ be the maximal literal of $C$, for some $j \in \{1, \ldots, m\}$, some ground base terms $s, t \in T_\Omega$, and $\dot{\approx} \in \{\approx, \not\approx\}$. If $L_j$ is negative, i.e. $L_j = (s \not\approx t)$, then $\overline{L}_j = (s \approx t)$, and every clause $C_\ell^j$, where $1 \leq \ell \leq n_j$, is strictly smaller than $L_j$, with respect to the underlying ordering $\succ$. But, if $L_j$ is positive, i.e. $L_j = (s \approx t)$, and in addition $(s \approx t \rightarrow) \in \mathsf{D}_{\mathcal{A}}$, then $C_j = \{\overline{L}_j\} = (s \approx t \rightarrow) = C_1^j \succ L_j$, consequently $C_1^j \succ C$. This is the only case, when a clause among $D_1, \ldots, D_n$ is greater than $C$. We formulate the observation in the following corollary.

COROLLARY 3.52 ► *Assume* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *is a base specification, with a base signature* $\Sigma = (\mathcal{S}, \Omega)$ *and a class of base algebras* $\mathscr{C}$. *Let* $\succ$ *be a reduction ordering total on ground* $\Sigma$*-terms.*

*Let* $\mathcal{A} \in \mathscr{C}$ *be a base algebra, and* $C$ *a ground base clause. Then* $\mathcal{A} \not\models C$ *if and only if there exist finitely many clauses* $D_1, \ldots, D_n \in \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, *for some* $n \geq 0$, *such that* $C, D_1, \ldots, D_n \models \bot$. *Moreover, if the maximal literal of* $C$ *is not an equation* $s \approx t$, *such that* $(s \approx t \rightarrow) \in \mathsf{D}_{\mathcal{A}}$, *for some ground base terms* $s, t \in T_\Omega$, *then* $D_i \prec C$, *for every* $i \in \{1, \ldots, n\}$.

Let us illustrate Lemma 3.51 on the following example.

EXAMPLE 3.53 ► Let $T$ be the theory of linear integer arithmetic, $\mathcal{A}$ the standard model of LA, and

$$C = 1 + 2 + 4 < 0 \rightarrow$$

a negative unit ground base clause. Recall that $1+2+4 < 0$ stands for $(1+2+4 < 0) \approx true_<$. Let $\succ$ be a reduction ordering total on ground $\Sigma$-terms with precedence $0 \prec 1 \prec -1 \prec 2 \prec -2 \prec \ldots \prec + \prec <$. Letting

$$C_1 = \quad \rightarrow 1 + 2 \approx 3$$

$$C_2 = \quad \rightarrow 3+4 \approx 7$$
$$C_3 = 7 < 0 \rightarrow$$

we obtain $C_1, C_2, C_3 \models C$ and $C_i \prec C$, for every $i \in \{1,2,3\}$, where $C_1, C_2 \in \mathsf{E}_{\mathcal{A}}$, and $C_3 \in \mathsf{D}_{\mathcal{A}}$. ■

As we have already discussed at the beginning of the current section, a considerable number of clauses in *sgi*($N$) may become redundant in the presence of clauses in $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$. In order to identify those clauses, we next introduce notions of $R^{\approx}_{\mathcal{A}}$-reduced terms and substitutions. These notions we use to define then simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances of the clause $N$, which, as we shall show later, constitute the non-redundant subset of *sgi*($N$).

◀ DEFINITION 3.54
$R^{\approx}_{\mathcal{A}}$-reduced term
$R^{\approx}_{\mathcal{A}}$-reduced substitution

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$. Let $\mathcal{A} \in \mathscr{C}$ be a base algebra.

A $\Sigma'$-***term*** $t$ is called $R^{\approx}_{\mathcal{A}}$***-reduced***, if it is in the normal form with respect to the rewrite system[1] $R^{\approx}_{\mathcal{A}}$:

$$t \ R^{\approx}_{\mathcal{A}}\text{-reduced} \overset{\text{def}}{\iff} t = t{\downarrow}_{R^{\approx}_{\mathcal{A}}}.$$

A ***substitution*** $\sigma$ is called $R^{\approx}_{\mathcal{A}}$***-reduced***, if every term $t \in im(\sigma)$ in the image of $\sigma$ is $R^{\approx}_{\mathcal{A}}$-reduced:

$$\sigma \ R^{\approx}_{\mathcal{A}}\text{-reduced} \overset{\text{def}}{\iff} \forall\, t \in im(\sigma) : t \ R^{\approx}_{\mathcal{A}}\text{-reduced}.$$

■

◀ DEFINITION 3.55
Simple ground
$R^{\approx}_{\mathcal{A}}$-reduced instances

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

Given a $\Sigma'$-term $t$, a $\Sigma'$-clause $C$, or a set of $\Sigma'$-clauses $N$, the ***set of all simple ground*** $R^{\approx}_{\mathcal{A}}$***-reduced instances*** of $t$, $C$, $N$ is defined as:

$$sgi_{\mathcal{A}}(t) \overset{\text{def}}{=} \{t\sigma \mid \sigma \text{ simple, grounding, and } R^{\approx}_{\mathcal{A}}\text{-reduced}\},$$
$$sgi_{\mathcal{A}}(C) \overset{\text{def}}{=} \{C\sigma \mid \sigma \text{ simple, grounding, and } R^{\approx}_{\mathcal{A}}\text{-reduced}\},$$
$$sgi_{\mathcal{A}}(N) \overset{\text{def}}{=} \{C\sigma \mid C \in N, \text{ and } \sigma \text{ simple, grounding, and } R^{\approx}_{\mathcal{A}}\text{-reduced}\},$$

respectively. ■

◀ DEFINITION 3.56
Extended clause set $N_{\mathcal{A}}$

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

If $\mathcal{A} \in \mathscr{C}$ is a base algebra, and $N$ a set of $\Sigma'$-clauses, then[2]

$$N_{\mathcal{A}} \overset{\text{def}}{=} sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}.$$

■

Our next point is to prove that $N_{\mathcal{A}} \models sgi(N)$ and that every clause in the set $sgi(N) \setminus N_{\mathcal{A}}$ is a logical consequence of a finite number of smaller clauses from $N_{\mathcal{A}}$, for any clause set $N$ and base algebra $\mathcal{A}$, Lemma 3.61. For proving that, we need some preliminary properties of $R^{\approx}_{\mathcal{A}}$-reduced terms and substitutions. First,

[1] Please, recall Definition 3.46 of the rewrite system $R^{\approx}_{\mathcal{A}}$, page 61.

[2] Please, recall Definition 3.49 of the clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, page 62.

we show that an equation $t \approx t'$ between any two ground $\Sigma'$-terms, one of which is an $R^{\approx}_{\mathcal{A}}$-reduced variant of another, follows from a finitely many smaller clauses from $\mathsf{E}_{\mathcal{A}}$, Proposition 3.57. Then, we prove an analogous property for an equation $t\sigma \approx t\sigma'$ between two simple ground instances of a $\Sigma'$-term $t \in T_{\Omega'}(\mathcal{X}')$, one of which is $R^{\approx}_{\mathcal{A}}$-reduced, Proposition 3.58. And the last preliminary step is Proposition 3.59, where we show that any simple ground instance $C\sigma$ of a $\Sigma'$-clause follows from the respective $R^{\approx}_{\mathcal{A}}$-reduced simple ground instance $C\sigma'$ of the clause and a finite subset of $\mathsf{E}_{\mathcal{A}}$, with the clauses implying $C\sigma$ being all smaller then $C\sigma$. For an illustration for the last statement see Example 3.60, page 72.

PROPOSITION 3.57 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ *and* $\Omega'' = \Omega' \setminus \Omega$ *be the enrichment sorts and operators, respectively. Let* $\succ$ *be a reduction ordering total on ground* $\Sigma'$*-terms.*

*Let* $\mathcal{A} \in \mathscr{C}$ *be a base algebra,* $t \in T_{\Omega'}$ *a ground* $\Sigma'$*-term, and* $t' = t{\downarrow}_{R^{\approx}_{\mathcal{A}}}$ *the normal form of* $t$ *with respect to the rewrite system* $R^{\approx}_{\mathcal{A}}$. *If the clause* $(\to t \approx t')$ *is not contained in* $\mathsf{E}_{\mathcal{A}}$, *then there exist finitely many clauses* $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}}$, *for some* $n \geq 0$, *such that* $C_1, \ldots, C_n \models t \approx t'$, *and* $C_i \prec \{t \approx t'\}$, *for every* $i \in \{1, \ldots, n\}$.

PROOF ► Assume $\mathcal{A}$ is an arbitrary algebra from $\mathscr{C}$. We prove the statement by induction on the depth $depth(t)$ of the term $t$.

**Induction base**. Assume $depth(t) = 0$, then $t$ is actually a constant symbol, say $a$.

If $a \in \Omega''$, that is $a$ is a free constant, then $t' = t = a$, because no rule in $R^{\approx}_{\mathcal{A}}$ can be used to rewrite a free term $t$, as for every $(l \approx r) \in R^{\approx}_{\mathcal{A}}$ the terms $l$ and $r$ are base, Definition 3.46 of $R^{\approx}_{\mathcal{A}}$. Thus, $t \approx t'$ is a tautology, and $n = 0$.

If, otherwise, $a \in \Omega$, that is $a$ is a base constant, then $t'$ is a base term, and $\mathcal{A} \models a \approx t'$. Therefore, by Proposition 3.50, there exist finitely many clauses $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}}$, such that $C_1, \ldots, C_n \models a \approx t'$ and $C_i \prec \{a \approx t'\}$, for every $i \in \{0, \ldots, n\}$ and some $n \geq 0$.

**Induction hypothesis**. Assume, the lemma holds for any ground $\Sigma'$-term $t \in T_{\Omega'}$ such that $depth(t) \leq k$, for some $k \geq 0$

**Induction step**. Assume $depth(t) = k + 1$. Without loss of generality, assume $t = f(t_1, \ldots, t_\ell)$, for some $\ell \geq 1$. Since $t' = t{\downarrow}_{R^{\approx}_{\mathcal{A}}}$, there is a finite rewriting $t \to^*_{R^{\approx}_{\mathcal{A}}} t'$ of a length $m = |t \to^*_{R^{\approx}_{\mathcal{A}}} t'|$, for some $m \geq 0$ (recall that any $R^{\approx}_{\mathcal{A}}$-rewriting from a ground $\Sigma'$-term is finite). If $m = 0$, then $t = t'$ and $(\to t \approx t')$ is a tautology. For otherwise, put $s_0 = t$ and $s_m = t'$, then the rewriting $t \to^*_{R^{\approx}_{\mathcal{A}}} t'$ can be represented as follows:

$$\begin{tikzcd}
t \arrow[rrr, "*"', "R^{\approx}_{\mathcal{A}}"] \arrow[d, phantom, "\shortparallel"] & & & t' \arrow[d, phantom, "\shortparallel"] \\
s_0 \arrow[r, "R^{\approx}_{\mathcal{A}}"'] & s_1 \arrow[r, "R^{\approx}_{\mathcal{A}}"'] & \cdots \arrow[r, "R^{\approx}_{\mathcal{A}}"'] & s_m
\end{tikzcd}$$

where for every $i \in \{0, \cdots, m-1\}$ the $(i+1)$-st rewrite step is performed via a rule

$(l_i \to r_i) \in R^{\approx}_{\mathcal{A}}$ at a position $p_i \in \rho(s_i)$ in $s_i$:

$$\begin{array}{ccc} s_i & \to_{\{l_i \to r_i\}} & s_{i+1} \\ \shortparallel & & \shortparallel \\ \overbrace{s_i[l_i]_{p_i}} & \to_{\{l_i \to r_i\}} & \overbrace{s_i[r_i]_{p_i}} \end{array}$$

Let $p_j$ be a highest position among $p_0, \ldots, p_m$, i.e. for every $i \neq j$ either $p_i$ is below $p_j$, or $p_i$ and $p_j$ are parallel. Consider two possible cases with respect to the length $|p_j|$:

- $|p_j| > 0$, i.e. no rewrite step has been done at $\varepsilon$. Hence, the top symbols of $t$ and $t'$ are the same symbol $f = top(t'/\varepsilon) = top(t/\varepsilon)$, for some $f \in \Omega'$, and $t' = f(t'_1, \ldots, t'_\ell)$ for some ground $\Sigma'$-terms $t'_1, \ldots, t'_\ell \in T_{\Omega'}$. Every immediate subterm $t'_i$ of $t'$ is the normal form of the respective immediate subterm $t_i$ of $t$ (as otherwise $t'$ would not be the normal form of $t$), and every immediate subterm $t_i$ of $t$ has depth $depth(t_i) \le k$. By induction hypothesis, for every pair $t_i$ and $t'_i$, $i \in \{1, \ldots, \ell\}$, there are finitely many clauses $C^i_1, \ldots, C^i_{n_i} \in \mathsf{E}_{\mathcal{A}}$, for some $n_i \ge 0$, such that
$$C^i_1, \ldots, C^i_{n_i} \models t_i \approx t'_i$$
and $C^i_j \preceq \{t_i \approx t'_i\}$, for every $j \in \{1, \ldots, n_i\}$. Putting together all clauses $C^i_j$, with $i$ ranging from 1 to $\ell$, and $j$ from 1 to $n_i$, each equation $t_i \approx t'_i$ is a consequence thereof:
$$C^1_1, \ldots, C^\ell_{n_\ell} \models t_1 \approx t'_1, \ldots, t_\ell \approx t'_\ell.$$
Then for any $\Sigma'$-algebra $\mathcal{A}'$ satisfying all the clauses $C^i_j$, we have:
$$\begin{array}{lll} & \mathcal{A}' \models C^1_1, \ldots, C^\ell_{n_\ell} & \\ \Rightarrow & \mathcal{A}' \models t_1 \approx t'_1, \ldots, t_\ell \approx t'_\ell & \\ \Leftrightarrow & \mathcal{A}'(t_i) = \mathcal{A}'(t'_i) & \text{// for each } i \in \{1, \ldots, \ell\} \end{array}$$
Hence,
$$\begin{array}{lll} & \mathcal{A}'(t) = \mathcal{A}'(f(t_1, \ldots, t_\ell)) & \\ & = f_{\mathcal{A}'}(\mathcal{A}'(t_1), \ldots, \mathcal{A}'(t_\ell)) & \\ & = f_{\mathcal{A}'}(\mathcal{A}'(t'_1), \ldots, \mathcal{A}'(t'_\ell)) & \text{// as } \mathcal{A}'(t_i) = \mathcal{A}'(t'_i) \\ & & \text{for each } i \in \{1, \ldots, \ell\} \\ & = \mathcal{A}'(f(t'_1, \ldots, t'_\ell)) & \\ & = \mathcal{A}'(t') & \\ \Leftrightarrow & \mathcal{A}' \models t \approx t' & \end{array}$$
As $\mathcal{A}'$ has been picked arbitrarily, we conclude
$$C^1_1, \ldots, C^\ell_{n_\ell} \models t \approx t'.$$
The subterm property of the reduction ordering $\prec$ yields
$$\{t_i \approx t'_i\} \prec \{f(t_1, \ldots, t_\ell) \approx f(t'_1, \ldots, t'_\ell)\},$$
for every $i \in \{1, \ldots, \ell\}$. Consequently, $C^i_j \prec \{t \approx t'\}$, for all $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, n_i\}$.

- $|p_j| = 0$, meaning $p_j = \varepsilon$. Therefore, $s_j = l_j$ and $s_{j+1} = r_j$ for some rule $(l_j \to r_j) \in R^{\approx}_{\mathcal{A}}$. Obviously, $s_j$ is a base term as well as every predecessor and successor of $s_j$ in the derivation $s_0 \to^*_{R^{\approx}_{\mathcal{A}}} s_m$, including $t = s_0$ and $t' = s_m$. This implies that[1] $t' = t{\downarrow}_{R^{\approx}_{\mathcal{A}}} = m^{\succ}_{\mathcal{A}}(t)$, hence $\mathcal{A} \models t \approx t'$. According to Proposition 3.50, the statement of the lemma is satisfied for this case too.

For the both cases we have shown that the lemma's statement holds; and we are done. ■

PROPOSITION 3.58 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively. Let* $\succ$ *be a reduction ordering total on ground* $\Sigma'$*-terms.*

*If* $\mathcal{A} \in \mathscr{C}$ *is a base algebra,* $t \in T_{\Omega'}(\mathcal{X}')$ *a* $\Sigma'$*-term,* $\sigma$ *a simple grounding substitution for* $t$*, and* $\sigma'$ *the* $R^{\approx}_{\mathcal{A}}$*-reduced*[2] $\sigma$*, then there exist finitely many clauses* $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}}$*, for some* $n \geq 0$*, such that* $C_1, \ldots, C_n \models t\sigma \approx t\sigma'$*, and* $C_i \preceq \{t\sigma \approx t\sigma'\}$*.*

PROOF ► We prove the proposition by induction on depth of $t$.

**Induction base**. If $depth(t) = 0$, then $t$ is either a constant, or a variable. If $t$ is a constant, then obviously $t\sigma = t = t\sigma'$ and $t\sigma \approx t\sigma'$ is a tautology. Assume $t$ is a variable, say $x \in \mathcal{X}'$. Let $s = x\sigma$ and $s' = x\sigma'$. From Definition 3.54 of a simple grounding $R^{\approx}_{\mathcal{A}}$-reduced substitution, we know $s' = s{\downarrow}_{R^{\approx}_{\mathcal{A}}}$. Consider the following two possible case:

- if $(\rightarrow s \approx s') \in \mathsf{E}_{\mathcal{A}}$, then the assertion trivially holds;
- otherwise, if $(\rightarrow s \approx s') \notin \mathsf{E}_{\mathcal{A}}$, we know from Proposition 3.57 that there exist finitely many clauses $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}}$, such that $C_1, \ldots, C_n \models s \approx s'$ and $C_i \prec \{s \approx s'\}$, for every $i \in \{1, \ldots, n\}$.

**Induction hypothesis**. Assume the proposition's statement holds for any term $t$ of depth $depth(t) \leq k$, for some $k \geq 0$.

**Induction step**. Consider a term $t$ of depth $depth(t) = k + 1$. Without loss of generality, we represent the term in form $t = f(t_1, \ldots, t_m)$, for some $f \in \Omega'$ and $t_i \in T_{\Omega'}(\mathcal{X}')$, for every $i \in \{1, \ldots, m\}$, where $m \geq 1$. From definition of a substitution application, we know $t\sigma = \big(f(t_1, \ldots, t_m)\big)\sigma = f(t_1\sigma, \ldots, t_m\sigma)$, and analogously for $t\sigma'$. By induction hypothesis, for every pair $t_i\sigma$ and $t_i\sigma'$, $i \in \{1, \ldots, m\}$, there exist finitely many clauses $C^i_1$, ..., $C^i_{n_i} \in \mathsf{E}_{\mathcal{A}}$, for some $n_i \geq 0$, such that

$$C^i_1, \ldots, C^i_{n_i} \models t_i\sigma \approx t_i\sigma'$$

and $C^i_j \preceq \{t_i\sigma \approx t_i\sigma'\}$, for every $j \in \{1, \ldots, n_i\}$. Putting together all clauses $C^i_j$, with $i$ ranging from 1 to $m$, and $j$ from 1 to $n_i$, each equation $t_i \approx t'_i$ is a consequence thereof:

$$C^1_1, \ldots, C^m_{n_m} \models t_1\sigma \approx t_1\sigma', \ldots, t_m\sigma \approx t_m\sigma'$$

Then for any $\Sigma'$-algebra $\mathcal{A}'$ satisfying all the clauses $C^i_j$, we have:

$$\begin{array}{lll} & \mathcal{A}' \models C^1_1, \ldots, C^m_{n_m} & \\ \Rightarrow & \mathcal{A}' \models t_1\sigma \approx t_1\sigma', \ldots, t_m\sigma \approx t_m\sigma' & \\ \Leftrightarrow & \mathcal{A}'(t_i\sigma) = \mathcal{A}'(t_i\sigma') & \text{// for each } i \in \{1, \ldots, m\} \end{array}$$

[1]Please, recall Definition 3.45 of $m^{\succ}_{\mathcal{A}}(t)$, page 60.

[2]Please, recall Definition 3.54 of a $R^{\approx}_{\mathcal{A}}$-reduced substitution, page 67.

Hence,

$$\begin{aligned}\mathcal{A}'\big(t\sigma\big) &= \mathcal{A}'\big((f(t_1,\dots,t_m))\sigma\big)\\ &= f_{\mathcal{A}'}\big(\mathcal{A}'(t_1\sigma),\dots,\mathcal{A}'(t_m\sigma)\big)\\ &= f_{\mathcal{A}'}\big(\mathcal{A}'(t_1\sigma'),\dots,\mathcal{A}'(t_m\sigma')\big) && \text{// as } \mathcal{A}'(t_i\sigma)=\mathcal{A}'(t_i\sigma') \\ & && \text{for each } i\in\{1,\dots,m\}\\ &= \mathcal{A}'\big((f(t_1,\dots,t_m))\sigma'\big)\\ &= \mathcal{A}'\big(t\sigma'\big)\\ \Leftrightarrow\quad \mathcal{A}' &\models t\sigma \approx t\sigma'\end{aligned}$$

As $\mathcal{A}'$ has been picked arbitrarily, we conclude

$$C_1^1,\dots,C_{n_m}^m \models t\sigma \approx t\sigma'.$$

The subterm property of the reduction ordering $\prec$ yields

$$\{t_i\sigma \approx t_i\sigma'\} \prec \{t\sigma \approx t\sigma'\},$$

for every $i \in \{1,\dots,m\}$. Consequently, $C_j^i \prec \{t\sigma \approx t\sigma'\}$, for every $i \in \{1,\dots,m\}$ and $j \in \{1,\dots,n_i\}$. ■

◄ PROPOSITION 3.59

*Assume* $\mathsf{HSp} = (\mathsf{Sp},\mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma,\mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma',Ax')$. *Assume* $\succ$ *is a reduction ordering total on ground* $\Sigma'$*-terms which orients any non-base term greater than any base one.*

*Let* $\mathcal{A} \in \mathscr{C}$ *be a base algebra,* $C$ *an arbitrary non-base* $\Sigma'$*-clause,* $\sigma$ *a simple grounding substitution for* $C$*, and* $\sigma'$ *the* $R_{\mathcal{A}}^{\approx}$*-reduced* $\sigma$*. Then there exist finitely many clauses* $C_1,\dots,C_n \in \mathsf{E}_{\mathcal{A}}$*, for some* $n \geq 0$*, such that* $C_1,\dots,C_n,C\sigma' \models C\sigma$*, and* $C_i \prec C\sigma$*, for every* $1 \leq i \leq n$*.*

◄ PROOF

Let $C$ consist of literals $L_1,\dots,L_m$, for some $m \geq 0$, and $L_i = (s_i \mathrel{\dot{\approx}} t_i)$, where $\dot{\approx} \in \{\approx,\not\approx\}$, for every $i \in \{1,\dots,m\}$. From Proposition 3.58 it follows that for every $s_i \mathrel{\dot{\approx}} t_i$ there exist finitely many clauses $C_1^i,\dots,C_{n_i}^i$ such that

$$C_1^i,\dots,C_{n_i}^i \models s_i\sigma' \approx s_i\sigma, t_i\sigma' \approx t_i\sigma,$$

Putting together all clauses $C_j^i$, we have

$$C_1^1,\dots,C_{n_m}^m \models s_k\sigma' \approx s_k\sigma, t_k\sigma' \approx t_k\sigma,$$

for any literal $L_k = (s_k \mathrel{\dot{\approx}} t_k)$ in the clause $C$, where $k \in \{1,\dots,m\}$.

Let $\mathcal{A}'$ be an arbitrary $\Sigma'$-algebra satisfying all clauses $C_1^1,\dots,C_{n_m}^m$ and the clause $C\sigma'$. Since $C\sigma'$ is ground, we know that $\mathcal{A}' \models C\sigma'$ if and only if $\mathcal{A}'$ satisfies at least one of its literals, say a literal $L_j\sigma' = (s_j\sigma' \mathrel{\dot{\approx}} t_j\sigma')$, where $\dot{\approx} \in \{\approx,\not\approx\}$, for some $j \in \{1,\dots,m\}$. On other hand, from the observation above and the assumption that $\mathcal{A}' \models C_1^1,\dots,C_{n_m}^m$ we learn

$$\mathcal{A}' \models s_k\sigma' \approx s_k\sigma, t_k\sigma' \approx t_k\sigma,$$

for any literal $L_k = (s_k \mathrel{\dot{\approx}} t_k)$ in the clause $C$. Putting these together and letting $k = j$, we obtain

$$\begin{aligned} & \mathcal{A}' \models s_j\sigma' \approx s_j\sigma, t_j\sigma' \approx t_j\sigma, s_j\sigma' \mathrel{\dot{\approx}} t_j\sigma'\\ \Rightarrow\quad & \mathcal{A}' \models s_j\sigma \mathrel{\dot{\approx}} t_j\sigma\\ \Leftrightarrow\quad & \mathcal{A}' \models L_j\sigma\\ \Rightarrow\quad & \mathcal{A}' \models C\sigma.\end{aligned}$$

As $\mathcal{A}'$ is an arbitrary algebra that satisfies the clauses $C_1^1, \ldots, C_{n_m}^m$, and $C\sigma'$, we conclude

$$C_1^1, \ldots, C_{n_m}^m, C\sigma' \models C\sigma.$$

Since $C$ is non-base, the clause $C\sigma$ is non-base as well, and therefore $C_j^i \prec C\sigma$. ■

Let us illustrate Proposition 3.59 by the following example.

EXAMPLE 3.60 ► Let $T$ be the theory of linear integer arithmetic. Assume we are given an abstracted clause

$$C = y \approx x + 3 \parallel f(x,z) \approx g(y,z) \to P(x,y)$$

and a substitution

$$\sigma = \{x \mapsto 1+2, y \mapsto 6, z \mapsto h(2+2+1)\},$$

where variables $x, y$ are base, and $z$ a free variable; $f, g, h, P$ free functions and a free predicate, respectively. Let $\succ$ be a reduction ordering total on ground $\Sigma'$-terms with precedence $0 \prec 1 \prec -1 \prec 2 \prec -2 \prec \ldots \prec + \prec \ < \ \prec f \prec g \prec h \prec P$.

The $R_{\mathcal{A}}^{\approx}$-reduced $\sigma$ is the substitution $\sigma' = \{x \mapsto 3, y \mapsto 6, z \mapsto h(5)\}$. The clauses $C\sigma$ and $C\sigma'$ are:

$$\begin{array}{llll} C\sigma = 6 \approx 1+2+3 & \parallel f(1+2, h(2+2+1)) & \approx g(6, h(2 \cdot 2+1)) & \to P(1+2, 6) \\ C\sigma' = 6 \approx 3+3 & \parallel f(3, \quad h(5)) & \approx g(6, h(5)) & \to P(3, \quad 6) \end{array}$$

We want to find clauses $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}}$ such that $C_1, \ldots, C_n, C\sigma' \models C\sigma$, and $C_i \prec C\sigma$, for every $0 \le i \le n$ and some $n \ge 0$. From Proposition 3.59 we know that there always exist finitely many such clauses. From the proofs of Propositions 3.59, 3.58, 3.57, and 3.50, we know that the required clauses $C_1, \ldots, C_n$ are the clauses in $\mathsf{E}_{\mathcal{A}}$ that correspond to the rewrite rules which have been used for reducing $\sigma$ to $\sigma'$. This way, letting

$$\begin{aligned} C_1 &= \ \to 1+2 \approx 3, \\ C_2 &= \ \to 2+2 \approx 4, \\ C_3 &= \ \to 4+1 \approx 5 \end{aligned}$$

we get

$$C_1, C_2, C_3, C\sigma' \models C\sigma,$$

where $C_i \prec C\sigma$, for every $i \in \{1,2,3\}$. ■

The next lemma makes the crucial assertion in this section that every clause $C$ in $sgi(N) \setminus N_{\mathcal{A}}$ follows from smaller clauses in $N_{\mathcal{A}}$. According to Definition 3.40 of the standard redundancy criterion $\mathcal{R}^{\mathcal{F}}$ for ground clauses, every such clause $C$ is redundant for $N_{\mathcal{A}}$.

LEMMA 3.61 ► *Assume* HSp = (Sp, Sp$'$) *is a hierarchic specification with the base specification* Sp = $(\Sigma, \mathscr{C})$ *and the body* Sp$'$ = $(\Sigma', Ax')$. *Assume* $\succ$ *is a reduction ordering total on ground* $\Sigma'$*-terms which orients any non-base term greater than any base one.*

*If $N$ is a set of $\Sigma'$-clauses, and $\mathcal{A} \in \mathscr{C}$ a base algebra, then:*

*(i) for every clause $C \in sgi(N) \setminus N_{\mathcal{A}}$ there exist finitely many clauses $C_1, \ldots, C_n \in N_{\mathcal{A}}$, such that $C_1, \ldots, C_n \models C$ and $C_i \prec C$, for every $0 \le i \le n$ and some $n \ge 0$;*

*(ii) $N_{\mathcal{A}} \models sgi(N)$.*

Statement (i) follows from Lemma 3.51 and Proposition 3.59; statement (ii) is a direct consequence of (i). ◄ PROOF ■

In the next section, we introduce the hierarchic redundancy criterion $\mathcal{R}^{\mathcal{H}}$ for general $\Sigma'$-clauses. The criterion is based on the standard redundancy criterion $\mathcal{R}^{\mathcal{F}}$ for ground clauses. Given a set $N$ of general $\Sigma'$-clauses, redundant clauses and inferences for $N$ are defined with respect to the clause set $N_{\mathcal{A}} = sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, which makes the results of the current section very important for the subsequent sections devoted to the hierarchic redundancy criterion and lifting and saturation of sets of general $\Sigma'$-clauses.

### 3.4.4 Hierarchic Redundancy Criterion

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

A key ingredient for satisfying the mutual saturation condition[1] is a correctly chosen redundancy criterion. In the "flat" case, the general redundancy criterion is formulated mainly according to the point of view that a non-ground clause/inference is a generalisation of all its ground instances. Thus, a clause $C$ is redundant for a given clause set $N$, if all its ground instances $gi(C)$ are redundant for $gi(N)$ with respect to the ground redundancy criterion $\mathcal{R}^{\mathcal{F}}_{F}$. This definition complies with the requirement of entailment of redundant clauses by non-redundant ones, as well as with the other requirements given in Definition 3.1 of an abstract redundancy criterion (see Section 3.4.2, Theorem 3.42).

As we have already discussed in the overview to Section 3.4, such a formulation is hardly applicable in the hierarchic case: first, we are straitened by *simple* substitutions, second, we are interested in $\mathscr{C}$-entailment[2] $\models_{\mathscr{C}}$, rather than in the general one $\models$. In this connection, we need to develop a hierarchic redundancy criterion, which reduces the redundancy notion of a redundant clause/inference to the level of *simple* ground instances (not *all* ground instances like in the flat case), and enjoys entailment relative to $\mathscr{C}$.

Still, we have to take into account the two following points essential for any redundancy criterion:

- on one hand, it must allow performing sufficiently many non-redundant inferences on a given clause set $N$ to guarantee saturation on the ground level;
- on the other hand, it should define a reasonable bunch of redundant clauses to keep $N$ as small as possible.

These two requirements are quite opponent to each other. Thus, the hierarchic redundancy criterion originally suggested by Bachmair, Ganzinger, and Waldmann in [BGW94], where they call a clause $C$ redundant for a given clause set $N$ whenever all its simple ground instances $sgi(C)$ are redundant for $sgi(N)$, is sufficient

[1]The mutual saturation condition is discussed in the overview to Section 3.4 on pages 50–51

[2]Please, recall Definition 3.25 of $\mathscr{C}$-entailment, page 45.

to ensure the saturation condition, but is too weak and does not allow to delete numerous redundant clauses above very trivial tautologies and trivially subsumed clauses. We have developed a more sophisticated criterion better fitting to the indicated restrictions and the intended purposes.

First, we introduce a notion of ($R^{\approx}_{\mathcal{A}}$-reduced) simple ground instances of an inference, Definitions 3.62-3.63, which we then use to define the hierarchic redundancy criterion $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_F, \mathcal{R}^{\mathcal{H}}_I)$ in Definition 3.64. In the next step, we show that our notion of hierarchic redundancy is more general than the one authored by Bachmair, Ganzinger, and Waldmann, Lemma 3.66. The rest of the section is devoted to proving that $\mathcal{R}^{\mathcal{H}}$ conforms to the properties of an abstract redundancy criterion given in Definition 3.1: in Propositions 3.67-3.69 we give some preliminary statements that then are exploited in Lemma 3.70 asserting $\mathcal{R}^{\mathcal{H}}$ to be really a redundancy criterion.

Before we proceed, let us recall the following notations:

- $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$ denotes the standard superposition calculus SUP for ground clauses, where $\mathcal{F}$ is the system of the standard inference rules, and $\mathcal{R}^{\mathcal{F}}$ is the standard redundancy criterion for ground clauses (see Section 3.4.2);
- $\mathcal{H}$ the system of the hierarchic inference rules (see Section 3.3).

DEFINITION 3.62 ► Simple Ground Instance of Inference

Assume HSp = (Sp, Sp′) is a hierarchic specification with the base specification Sp = $(\Sigma, \mathscr{C})$ and the body Sp′ = $(\Sigma', Ax')$.

Let $I$ be a hierarchic superposition inference in $\mathcal{H}$ with premises $C_1, \dots, C_n$ and conclusion $C$, where the clauses $C_1, \dots, C_n$ have no variables in common. Let $I'$ be a standard superposition inference in $\mathcal{F}$ with premises $C'_1, \dots, C'_n$ and conclusion $C'$. If $\sigma$ is a simple grounding substitution s.t. $C\sigma = C'$ and $C_i\sigma = C'_i$ for all $1 \le i \le n$, then $I'$ is called a ***simple ground instance of inference*** $I$. The set of all simple ground instances of an inference $I$ is denoted by $sgi(I)$. ■

DEFINITION 3.63 ► Simple Ground $R^{\approx}_{\mathcal{A}}$-reduced Instance of Inference

Assume HSp = (Sp, Sp′) is a hierarchic specification with the base specification Sp = $(\Sigma, \mathscr{C})$ and the body Sp′ = $(\Sigma', Ax')$.

Let $\mathcal{A} \in \mathscr{C}$ be a base algebra, and $I$ a hierarchic superposition inference in $\mathcal{H}$ with premises $C_1, \dots, C_n$ and conclusion $C$, where the clauses $C_1, \dots, C_n$ have no variables in common. Let $I'$ be a standard superposition inference in $\mathcal{F}$ with premises $C'_1, \dots, C'_n$ and conclusion $C'$. If $\sigma$ is a simple grounding $R^{\approx}_{\mathcal{A}}$-reduced substitution[1] s.t. $C\sigma = C'$ and $C_i\sigma = C'_i$ for all $1 \le i \le n$, then $I'$ is called a ***simple ground $R^{\approx}_{\mathcal{A}}$-reduced instance of inference*** $I$. The set of all simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances of an inference $I$ is denoted by $sgi_{\mathcal{A}}(I)$. ■

DEFINITION 3.64 ► $\mathcal{H}$-Redundant Clauses $\mathcal{R}^{\mathcal{H}}_F$ $\mathcal{H}$-Redundant Inferences $\mathcal{R}^{\mathcal{H}}_I$ $\mathcal{R}^{\mathcal{H}}$

Assume HSp = (Sp, Sp′) is a hierarchic specification with the base specification Sp = $(\Sigma, \mathscr{C})$ and the body Sp′ = $(\Sigma', Ax')$.

Let $\mathcal{A} \in \mathscr{C}$ be a base algebra, and $N$ a set of $\Sigma'$-clauses. We define $\mathcal{R}^{\mathcal{H}}_F(N)$ to be the set of all $\Sigma'$-clauses $C$ such that for every base algebra $\mathcal{A} \in \mathscr{C}$ all simple ground

[1]The notions of the rewrite system $R^{\approx}_{\mathcal{A}}$ and $R^{\approx}_{\mathcal{A}}$-reduced terms/substitutions are introduced in Section 3.4.3, Definitions 3.46 and 3.54, respectively.

$R^{\approx}_{\mathcal{A}}$-reduced instances of $C$ are redundant[1] for $N_{\mathcal{A}}$ or contained[2] in $\mathsf{E}_{\mathcal{A}}$ or $\mathsf{D}_{\mathcal{A}}$:

$$\mathcal{R}^{\mathcal{H}}_F(N) \overset{\text{def}}{=} \{C \in Cl_{\Sigma'} \mid \forall \mathcal{A} \in \mathscr{C} : sgi_{\mathcal{A}}(C) \subseteq \mathcal{R}^{\mathcal{F}}_F(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}\}$$

We define $\mathcal{R}^{\mathcal{H}}_I(N)$ to be the set of all hierarchic superposition inferences $I$ such that either $I$ is not a constraint refutation inference and for every base algebra $\mathcal{A} \in \mathscr{C}$ all simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances of $I$ are redundant for $N_{\mathcal{A}}$, or else $I$ is a constraint refutation inference and $\square \in N$:

$$\mathcal{R}^{\mathcal{H}}_I(N) \overset{\text{def}}{=} \{I \in \mathcal{H}(N) \mid \begin{array}{l} \text{(i) } I \text{ not a Constraint Refutation inference,} \\ \quad \text{and } \forall \mathcal{A} \in \mathscr{C} : sgi_{\mathcal{A}}(I) \subseteq \mathcal{R}^{\mathcal{F}}_I(N_{\mathcal{A}}); \text{ or} \\ \text{(ii) } I \text{ a Constraint Refutation inference,} \\ \quad \text{and } \square \in N\}. \end{array}$$

We write $\mathcal{R}^{\mathcal{H}}$ to denote the pair $(\mathcal{R}^{\mathcal{H}}_I, \mathcal{R}^{\mathcal{H}}_F)$. ■

Our goal now is to show that $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_I, \mathcal{R}^{\mathcal{H}}_F)$ is a redundancy criterion with respect to the inference system $\mathcal{H}$ and entailment relation $\models_{\mathscr{C}}$. But first we show that our definition of $\mathcal{H}$-redundant clauses and inferences is stronger than the one given in the paper of Bachmair, Ganzinger, and Waldmann [BGW94], meaning that a clause or an inference that is redundant with respect to the hierarchic redundancy criterion given in [BGW94] is also redundant relative to our notion of the hierarchic redundancy criterion.

In the following proposition we prove that ground clauses/inferences that are redundant with respect to the set $sgi(N)$ of all simple ground instances of a clause set $N$ are also redundant with respect to the set $N_{\mathcal{A}}$, for any base algebra $\mathcal{A} \in \mathscr{C}$. The property is needed in the proof of Lemma 3.66 where we show that a clause or an inference, that is redundant in the sense of Bachmair, Ganzinger, and Waldmann, is also redundant relative to Definition 3.64.

◄ PROPOSITION 3.65

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let* $\mathcal{A} \in \mathscr{C}$ *be a base algebra,* $N$ *a set of* $\Sigma'$*-clauses. If a ground* $\Sigma'$*-clause* $C$ *(an* $\mathcal{F}$*-inference* $I$*) is redundant for* $sgi(N)$*, then it is also redundant for* $N_{\mathcal{A}}$*:*

$$\forall \mathcal{A} \in \mathscr{C} : \mathcal{R}^{\mathcal{F}}(sgi(N)) \subseteq \mathcal{R}^{\mathcal{F}}(N_{\mathcal{A}}).$$

◄ PROOF

Follows from Lemma 3.61(i) and Lemma 3.43. ■

Now we are in a position to show our redundancy notion being stronger the one given by Bachmair, Ganzinger, and Waldmann in [BGW94], according to which[3] a $\Sigma'$-clause $C$ is redundant for a set of $\Sigma'$-clauses $N$ if all simple ground instances of $C$ are redundant for the set of all simple ground instances of $N$; and a hierarchic $\mathcal{H}$-inference $I$ is redundant for $N$ if either (i) $I$ is not a constraint refutation

[1]Please, recall Definition 3.40 of the standard redundancy criterion $\mathcal{R}^{\mathcal{F}} = (\mathcal{R}^{\mathcal{F}}_F, \mathcal{R}^{\mathcal{F}}_I)$ for ground clauses, page 56.

[2]The notions $\mathsf{E}_{\mathcal{A}}$, $\mathsf{D}_{\mathcal{A}}$, and $N_{\mathcal{A}}$ are introduced in Section 3.4.3, Definitions 3.49 and 3.56, respectively.

[3]The hierarchic redundancy criterion, invented by Bachmair, Ganzinger, and Waldmann, can be found in [BGW94] on page 203, Definition 14.

inference and all simple ground instances of $I$ are redundant for the set of all simple ground instances of $N$, or else (ii) $I$ is a constraint refutation inference and an empty clause $\Box$ is in $N$:

$$\begin{array}{rl} C \text{ redundant}^1\text{, if} & sgi(C) \subseteq \mathcal{R}^{\mathcal{F}}_F(sgi(N)), \\ I \text{ redundant}^1\text{, if} & \text{(i) } I \text{ not a Constraint Refutation inference,} \\ & \quad \text{and } sgi(I) \subseteq \mathcal{R}^{\mathcal{F}}_I(sgi(N)); \text{ or} \\ & \text{(ii) } I \text{ a Constraint Refutation inference,} \\ & \quad \text{and } \Box \in N, \end{array}$$

where $\mathcal{R}^{\mathcal{F}} = (\mathcal{R}^{\mathcal{F}}_F, \mathcal{R}^{\mathcal{F}}_I)$ is the standard redundancy criterion for ground clauses, Definition 3.40. In the following lemma, we do not consider the case of $I$ being a constraint refutation inference, because this case is the same as in our definition of an $\mathcal{H}$-redundant inference.

LEMMA 3.66 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $N$ be a set of $\Sigma'$-clauses, $C$ a $\Sigma'$-clause, and $I$ an $\mathcal{H}$-inference. Then:*

*(i)* $sgi(C) \subseteq \mathcal{R}^{\mathcal{F}}_F(sgi(N))$ *implies* $sgi_{\mathcal{A}}(C) \subseteq \mathcal{R}^{\mathcal{F}}_F(N_{\mathcal{A}})$, *for every* $\mathcal{A} \in \mathscr{C}$;

*(ii)* $sgi(I) \subseteq \mathcal{R}^{\mathcal{F}}_I(sgi(N))$ *implies* $sgi_{\mathcal{A}}(I) \subseteq \mathcal{R}^{\mathcal{F}}_I(N_{\mathcal{A}})$, *for every* $\mathcal{A} \in \mathscr{C}$, *if $I$ is not a constraint refutation inference.*

PROOF ► For (i) assume $sgi(C) \subseteq \mathcal{R}^{\mathcal{F}}_F(sgi(N))$, then:

$$\begin{array}{rll} & sgi_{\mathcal{A}}(C) \subseteq \mathcal{R}^{\mathcal{F}}_F(sgi(N)) & \text{// as } sgi_{\mathcal{A}}(C) \subseteq sgi(C) \\ \Rightarrow & sgi_{\mathcal{A}}(C) \subseteq \mathcal{R}^{\mathcal{F}}_F(N_{\mathcal{A}}) & \text{// by Proposition 3.65} \end{array}$$

For (ii) a proof is the same as for (i) if $C$ replaced with $I$, and $\mathcal{R}^{\mathcal{F}}_F$ with $\mathcal{R}^{\mathcal{F}}_I$. ■

Until recently, [BGW94] had been the *only* work so far dedicated to hierarchic superposition, where a definition of hierarchic redundancy is given and shown to be really a redundancy criterion. All subsequent works [AKW09a, EKS+11, KW11, FW11, FKW12b] are based on this paper, and replicate the definition of hierarchic redundancy from there. As it turns out, most results presented in these works are true *only* with respect to our definition of hierarchic redundancy, but false with respect to the one of [BGW94]. The reason is that the mentioned works heavily rely on using reduction rules, which, as claimed, obey the redundancy criterion of [BGW94], but in fact they do not. However, reduction rules exploited there agree with our definition, so the papers' contributions are revived if coped with the redundancy criterion given here in Definition 3.64.

### Comparison to hierarchic redundancy criterion of [BW13]

The most recent work by Baumgartner and Waldmann [BW13] offers another hierarchic reduction criterion very much similar to the one proposed by us, namely:

[1]. . . with respect to Bachmair, Ganzinger, and Waldmann's hierarchic redundancy notion.

$C$ redundant, if $sgi(C) \subseteq \mathcal{R}_F^{\mathcal{F}}(sgi(N) \cup \mathsf{Gnd}(\mathscr{C})) \cup \mathsf{Gnd}(\mathscr{C})$,

$I$ redundant, if (i) $I$ not a Constraint Refutation inference, and $sgi(I) \subseteq \mathcal{R}_I^{\mathcal{F}}(sgi(N) \cup \mathsf{Gnd}(\mathscr{C}))$; or
(ii) $I$ a Constraint Refutation inference, and $\square \in N$,

where $\mathsf{Gnd}(\mathscr{C})$ is the set of all ground base formulae that are satisfied by every base algebra $\mathcal{A} \in \mathscr{C}$. (Baumgartner, Waldmann [BW13])

From Lemma 3.51 it follows that the set $\mathsf{Gnd}(\mathscr{C})$ is entailed by $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, for every base algebra $\mathcal{A} \in \mathscr{C}$, but $\mathsf{Gnd}(\mathscr{C})$ does not entail $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, in general. Consequently, the Hierarchic Redundancy Criterion $\mathcal{R}^{\mathcal{H}}$ proposed in Definition 3.64 is stronger than (or, in general, is at least as strong as) the one from [BW13].

We proceed to show that $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}_I^{\mathcal{H}}, \mathcal{R}_F^{\mathcal{H}})$ is a redundancy criterion with respect to the inference system $\mathcal{H}$ and entailment relation $\models_{\mathscr{C}}$. To do this we need to make sure that $\mathcal{R}^{\mathcal{H}}$ agrees with each condition in Definition 3.1 of an abstract redundancy criterion. We prove this in Lemma 3.70, and in Propositions 3.67-3.69 we show some preliminary properties of simple ground instances that are exploited in the proof of Lemma 3.70. Particularly, in Proposition 3.67 we provide three sufficient conditions of entailment relative to $\mathscr{C}$ of one clause set by another expressed in terms of simple ground instances of the clause sets. Proposition 3.68 asserts a simple property that for any two clause sets $N$ and $M$, if a clause $C$ is contained in $sgi_{\mathcal{A}}(N)$ and not in $sgi_{\mathcal{A}}(M)$, then $C$ is a simple ground instance of some clause in $N \setminus M$. Proposition 3.69 states that given a clause set $N$, the set $sgi_{\mathcal{A}}(N) \setminus (\mathcal{R}_F^{\mathcal{F}}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})$ of all non-redundant simple ground $R_{\mathcal{A}}^{\approx}$-reduced instances of $N$ is contained in the set $sgi_{\mathcal{A}}(N \setminus \mathcal{R}_F^{\mathcal{H}}(N))$ of all simple ground $R_{\mathcal{A}}^{\approx}$-reduced instances of the non-redundant subset of $N$, or roughly speaking, non-redundant simple ground $R_{\mathcal{A}}^{\approx}$-reduced instances of $N$ are instances of non-redundant clauses in $N$.

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.* ◄ PROPOSITION 3.67

*Let* $N$ *and* $M$ *be sets of* $\Sigma'$*-clauses. If one of the following conditions is fulfilled*[1]*:*

*(i)* $sgi(N) \models sgi(M)$*, or*

*(ii)* $sgi_{\mathcal{A}}(N) \models sgi_{\mathcal{A}}(M)$ *for every base algebra* $\mathcal{A} \in \mathscr{C}$*, or*

*(iii)* $N_{\mathcal{A}} \models sgi_{\mathcal{A}}(M)$ *for every base algebra* $\mathcal{A} \in \mathscr{C}$*,*

*then* $N \models_{\mathscr{C}} M$*.*

First we show item (iii) implies $N \models_{\mathscr{C}} M$. So, assume there exists a $\Sigma'$-model $\mathcal{A}'$ of $N$ such that $\mathcal{A}'|_{\Sigma} = \mathcal{A}$ for some base algebra $\mathcal{A} \in \mathscr{C}$; if there is no such a model ◄ PROOF

[1] Please, recall Definition 3.49 of the clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, page 62; Definition 3.54 of a set of all simple ground $R_{\mathcal{A}}^{\approx}$-reduced instances $sgi_{\mathcal{A}}$, page 67; and Definition 3.56 of a clause set $N_{\mathcal{A}}$, page 67.

$\mathcal{A}'$ of $N$, then $N \models_{\mathscr{C}} M$ trivially holds. Assume also $N_{\mathcal{A}} \models sgi_{\mathcal{A}}(M)$ holds for every base algebra $\mathcal{A} \in \mathscr{C}$. Then:

$$\begin{array}{lll}
 & \mathcal{A}' \models gi(N) & \text{// as } \mathcal{A}' \models N \text{ by assum.} \\
 & \quad \models sgi_{\mathcal{A}}(N) & \text{// as } sgi_{\mathcal{A}}(N) \subseteq gi(N) \\
\Rightarrow & \mathcal{A}' \models sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \text{// as } \mathcal{A}'|_{\Sigma} = \mathcal{A} \in \mathscr{C} \text{ by assum.} \\
 & \quad = N_{\mathcal{A}} & \text{// by Def. 3.56} \\
\Rightarrow & \mathcal{A}' \models sgi_{\mathcal{A}}(M) & \text{// as } N_{\mathcal{A}} \models sgi_{\mathcal{A}}(M) \text{ by assum.} \\
\Rightarrow & \mathcal{A}' \models sgi_{\mathcal{A}}(M) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \text{// as } \mathcal{A}'|_{\Sigma} = \mathcal{A} \text{ by assum.} \\
 & \quad = M_{\mathcal{A}} & \text{// by Def. 3.56} \\
 & \quad \models sgi(M) & \text{// by Lemma 3.61}
\end{array}$$

Thus, $\mathcal{A}' \models sgi(M)$.

From the assumption $\mathcal{A}'|_{\Sigma} = \mathcal{A} \in \mathscr{C}$, we know that $\mathsf{S}_{\mathcal{A}'} = \mathsf{S}_{\mathcal{A}}$ holds for every base sort $\mathsf{S} \in \mathcal{S}$. Consider an arbitrary ground $\Sigma'$-term $t' \in T_{\Omega'}$ of a base sort $\mathsf{S} \in \mathcal{S}$. Let $\mathcal{A}'(t') = e$ for some $e \in \mathsf{S}_{\mathcal{A}'}$. Since $\mathsf{S}_{\mathcal{A}'} = \mathsf{S}_{\mathcal{A}}$ and $\mathscr{C}$ consists of term-generated algebras, there exists a ground base term $t \in T_{\Omega}$ such that $\mathcal{A}(t) = e = \mathcal{A}'(t) = \mathcal{A}'(t')$. From this we conclude that for any grounding substitution $\sigma'$ there exists an $\mathcal{A}'$-equivalent simple grounding substitution $\sigma$, in the sense that $dom(\sigma') = dom(\sigma)$, and $\forall\, x \in dom(\sigma') : \mathcal{A}'(x\sigma') = \mathcal{A}'(x\sigma)$. This implies that each ground instance $C\sigma'$ of any clause $C$ in $M$ is equivalent to some simple ground instance $C\sigma$ under $\mathcal{A}'$, i.e. $\mathcal{A}' \models C\sigma'$ iff $\mathcal{A}' \models C\sigma$. Therefore,

$$\begin{array}{lll}
 & \mathcal{A}' \models sgi(M) \text{ iff } \mathcal{A}' \models gi(M) & \\
\Rightarrow & \mathcal{A}' \models gi(M) & \text{// as } \mathcal{A}' \models sgi(M) \\
\Leftrightarrow & \mathcal{A}' \models M &
\end{array}$$

As $\mathcal{A}'$ has been picked arbitrarily, we conclude $N \models_{\mathscr{C}} M$.

Proofs for items (i) and (ii) are similar to (and simpler than) that for (iii). ■

PROPOSITION 3.68 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $N, M$ be sets of $\Sigma'$-clauses. For every base algebra $\mathcal{A} \in \mathscr{C}$ it holds that*

$$sgi_{\mathcal{A}}(N) \setminus sgi_{\mathcal{A}}(M) \subseteq sgi_{\mathcal{A}}(N \setminus M).$$

PROOF ► Let $\mathcal{A}$ be a base algebra in $\mathscr{C}$, and $C'$ a clause in $sgi_{\mathcal{A}}(N) \setminus sgi_{\mathcal{A}}(M)$, if any. Obviously, $C' \in sgi_{\mathcal{A}}(N)$ and $C' \notin sgi_{\mathcal{A}}(M)$. Since $C' \in sgi_{\mathcal{A}}(N)$, there exists a clause $C \in N$ such that $C\sigma = C'$ for some simple grounding $R^{\approx}_{\mathcal{A}}$-reduced substitution $\sigma$, hence $C' \in sgi_{\mathcal{A}}(C)$. On the other hand, since $C' \notin sgi_{\mathcal{A}}(M)$, there is no $D \in M$ such that $D\rho = C'$, for a simple grounding $R^{\approx}_{\mathcal{A}}$-reduced substitution $\rho$. Therefore, $C \notin M$, hence $C \in N \setminus M$, and consequently $C' \in sgi_{\mathcal{A}}(N \setminus M)$. As $\mathcal{A}$ and $C'$ have been picked arbitrarily, the assertion follows. ■

PROPOSITION 3.69 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $N$ be a set of $\Sigma'$-clauses. For every base algebra $\mathcal{A} \in \mathscr{C}$ it holds that*

$$sgi_{\mathcal{A}}(N) \setminus (\mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \subseteq sgi_{\mathcal{A}}(N \setminus \mathcal{R}^{\mathcal{H}}_{F}(N)).$$

PROOF ► Let $\mathcal{A}$ be a base algebra in $\mathscr{C}$, and $C'$ a clause in $sgi_{\mathcal{A}}(N) \setminus (\mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})$, if any. Obviously, $C' \in sgi_{\mathcal{A}}(N)$, and $C' \notin \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$. Since $C' \in sgi_{\mathcal{A}}(N)$, there exists a clause $C \in N$, s.t. $C\sigma = C'$, for some simple grounding $R^{\approx}_{\mathcal{A}}$-reduced substitution $\sigma$, hence $C' \in sgi_{\mathcal{A}}(C)$. Thus, we have:

$$\begin{array}{rll} & C' \notin \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \\ \Rightarrow & sgi_{\mathcal{A}}(C) \not\subseteq \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \text{// as } C' \in sgi_{\mathcal{A}}(C) \\ \Rightarrow & C \notin \mathcal{R}^{\mathcal{H}}_{F}(N) & \text{// by Def. 3.64 of } \mathcal{R}^{\mathcal{H}}_{F} \\ \Rightarrow & C \in N \setminus \mathcal{R}^{\mathcal{H}}_{F}(N) & \text{// as } C \in N \\ \Rightarrow & C' \in sgi_{\mathcal{A}}(N \setminus \mathcal{R}^{\mathcal{H}}_{F}(N)) & \text{// as } C \in sgi_{\mathcal{A}}(C) \end{array}$$

Since the algebra $\mathcal{A}$ and clause $C'$ have been picked arbitrarily, the assertion follows. ■

Having disposed of the preliminary step that provides all properties needed, next we prove $\mathcal{R}^{\mathcal{H}}$ to be a redundancy criterion.

◄ LEMMA 3.70 Hierarchic Redundancy Criterion

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*The pair* $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_{I}, \mathcal{R}^{\mathcal{H}}_{F})$ *is a redundancy criterion with respect to*[1] *the inference system* $\mathcal{H}$ *and entailment relation* $\models_{\mathscr{C}}$.

◄ PROOF

For $\mathcal{R}^{\mathcal{H}}$ to be a redundancy criterion, it must satisfy all conditions of Definition 3.1, which in the context of $\mathcal{H}$ and $\models_{\mathscr{C}}$ are:

(i) $N \setminus \mathcal{R}^{\mathcal{H}}_{F}(N) \models_{\mathscr{C}} \mathcal{R}^{\mathcal{H}}_{F}(N)$;

(ii) if $N \subseteq M$, then $\mathcal{R}^{\mathcal{H}}_{F}(N) \subseteq \mathcal{R}^{\mathcal{H}}_{F}(M)$ and $\mathcal{R}^{\mathcal{H}}_{I}(N) \subseteq \mathcal{R}^{\mathcal{H}}_{I}(M)$;

(iii) if $I \in \mathcal{H}(M)$ and $concl(I) \in N \cup \mathcal{R}^{\mathcal{H}}_{F}(N)$, then $I \in \mathcal{R}^{\mathcal{H}}_{I}(N)$;

(iv) if $M \subseteq \mathcal{R}^{\mathcal{H}}_{F}(N)$, then $\mathcal{R}^{\mathcal{H}}_{F}(N) \subseteq \mathcal{R}^{\mathcal{H}}_{F}(N \setminus M)$ and $\mathcal{R}^{\mathcal{H}}_{I}(N) \subseteq \mathcal{R}^{\mathcal{H}}_{I}(N \setminus M)$.

Next, we show every item in turn.

**Condition (i)**. Let $N$ be a set of $\Sigma'$-clauses, and $C$ a clause in $\mathcal{R}^{\mathcal{H}}_{F}(N)$. From Definition 3.64 of $\mathcal{R}^{\mathcal{H}}_{F}$ we know that $sgi_{\mathcal{A}}(C) \subseteq \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ holds for every base algebra $\mathcal{A} \in \mathscr{C}$. Pick an arbitrary base algebra $\mathcal{A} \in \mathscr{C}$. We split the set $sgi_{\mathcal{A}}(C)$ into two parts $M_1$ and $M_2$ defined as follows:

$$\begin{array}{rcl} M_1 & = & sgi_{\mathcal{A}}(C) \cap \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \\ M_2 & = & sgi_{\mathcal{A}}(C) \cap (\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}). \end{array}$$

Evidently, $sgi_{\mathcal{A}}(C) = M_1 \cup M_2$. Since $\mathcal{R}^{\mathcal{F}}$ is a redundancy criterion, Theorem 3.42, we have

$$\begin{array}{rll} N_{\mathcal{A}} \setminus \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) & \models \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) & \text{// by Cond. (i) of Def. 3.1} \\ & \models M_1 & \text{// as } M_1 \subseteq \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \\ \Rightarrow N_{\mathcal{A}} \setminus \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \models M_1 \cup M_2 & \text{// as } M_2 \subseteq \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \end{array}$$

[1] Please, recall Definition 3.32 of the hierarchic superposition inference system $\mathcal{H}$, page 49; Definition 3.39 of the standard superposition inference system $\mathcal{F}$, page 56; Definition 3.25 of the entailment relation $\models_{\mathscr{C}}$, page 45; Definition 3.46 of the rewrite system $R^{\approx}_{\mathcal{A}}$, page 61; Definition 3.49 of clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, page 62; Definition 3.55 of sets of all simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances $sgi_{\mathcal{A}}$, page 67; Definition 3.56 of a clause set $N_{\mathcal{A}}$, page 67; Definitions 3.64 and 3.40 of the hierarchic $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_{F}, \mathcal{R}^{\mathcal{H}}_{I})$ and the standard ground $\mathcal{R}^{\mathcal{F}} = (\mathcal{R}^{\mathcal{F}}_{F}, \mathcal{R}^{\mathcal{F}}_{I})$ redundancy criterions, pages 74 and 56, respectively.

$$= sgi_{\mathcal{A}}(C)$$

Consider the set $N_{\mathcal{A}} \setminus \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ more elaborately:

$$\begin{aligned}
&N_{\mathcal{A}} \setminus \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \\
&= (\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N)) \setminus \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} && \text{// by def. of } N_{\mathcal{A}} \\
&= (\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \setminus \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup sgi_{\mathcal{A}}(N) \setminus \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \\
&= \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N) \setminus \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \\
&= \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N) \setminus (\mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \\
&\subseteq \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N \setminus \mathcal{R}^{\mathcal{H}}_{F}(N)) && \text{// by Prop. 3.69} \\
&= \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N') && \text{// where } N' = N \setminus \mathcal{R}^{\mathcal{H}}_{F}(N) \\
&= N'_{\mathcal{A}}
\end{aligned}$$

As the algebra $\mathcal{A}$ has been picked arbitrarily, it follows that $N'_{\mathcal{A}} \models sgi_{\mathcal{A}}(C)$ holds for every base algebra $\mathcal{A} \in \mathscr{C}$. From Proposition 3.67 (iii), it follows $N' \models_{\mathscr{C}} C$, or equivalently, $N \setminus \mathcal{R}^{\mathcal{H}}_{F}(N) \models_{\mathscr{C}} C$. As $C$ is an arbitrary clause from $\mathcal{R}^{\mathcal{H}}_{F}(N)$, we conclude $N \setminus \mathcal{R}^{\mathcal{H}}_{F}(N) \models_{\mathscr{C}} \mathcal{R}^{\mathcal{H}}_{F}(N)$.

**Condition (ii).** Let $N$ and $M$ be sets of $\Sigma'$-clauses, such that $N \subseteq M$. Evidently, $sgi_{\mathcal{A}}(N) \subseteq sgi_{\mathcal{A}}(M)$ is true for every base algebra $\mathcal{A} \in \mathscr{C}$. Consequently, $N_{\mathcal{A}} \subseteq M_{\mathcal{A}}$ for every $\mathcal{A} \in \mathscr{C}$. Since $\mathcal{R}^{\mathcal{F}}$ has the monotonicity property (Condition (ii) of Definition 3.1), we conclude

$$\forall \mathcal{A} \in \mathscr{C} : \mathcal{R}^{\mathcal{F}}(N_{\mathcal{A}}) \subseteq \mathcal{R}^{\mathcal{F}}(M_{\mathcal{A}}).$$

Let $C$ be an arbitrary clause from $\mathcal{R}^{\mathcal{H}}_{F}(N)$. Thus, we have

$$\begin{aligned}
& C \in \mathcal{R}^{\mathcal{H}}_{F}(N) \\
\Leftrightarrow\quad & \forall \mathcal{A} \in \mathscr{C} : sgi_{\mathcal{A}}(C) \subseteq \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} && \text{// by Def. 3.64 of } \mathcal{R}^{\mathcal{H}}_{F} \\
\Rightarrow\quad & \forall \mathcal{A} \in \mathscr{C} : sgi_{\mathcal{A}}(C) \subseteq \mathcal{R}^{\mathcal{F}}_{F}(M_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} && \text{// as } \mathcal{R}^{\mathcal{F}}(N_{\mathcal{A}}) \subseteq \mathcal{R}^{\mathcal{F}}(M_{\mathcal{A}}), \\
& && \text{by the observ. above} \\
\Leftrightarrow\quad & C \in \mathcal{R}^{\mathcal{H}}_{F}(M) && \text{// by def. of } \mathcal{R}^{\mathcal{H}}_{F}
\end{aligned}$$

Let $I$ be an arbitrary inference from $\mathcal{R}^{\mathcal{H}}_{I}(N)$ different from constraint refutation, then:

$$\begin{aligned}
& I \in \mathcal{R}^{\mathcal{H}}_{I}(N) \\
\Leftrightarrow\quad & \forall \mathcal{A} \in \mathscr{C} : sgi_{\mathcal{A}}(I) \subseteq \mathcal{R}^{\mathcal{F}}_{I}(N_{\mathcal{A}}) && \text{// by Def. 3.64 of } \mathcal{R}^{\mathcal{H}}_{I} \\
\Rightarrow\quad & \forall \mathcal{A} \in \mathscr{C} : sgi_{\mathcal{A}}(I) \subseteq \mathcal{R}^{\mathcal{F}}_{I}(M_{\mathcal{A}}) && \text{// as } \mathcal{R}^{\mathcal{F}}(N_{\mathcal{A}}) \subseteq \mathcal{R}^{\mathcal{F}}(M_{\mathcal{A}}), \\
& && \text{by the observ. above} \\
\Rightarrow\quad & I \in \mathcal{R}^{\mathcal{H}}_{I}(M) && \text{// by def. of } \mathcal{R}^{\mathcal{H}}_{I}
\end{aligned}$$

If $I$ is a constraint refutation inference, then $\square \in N$ – by definition of $\mathcal{R}^{\mathcal{H}}_{I}$. Consequently, $\square \in M$, and $I \in \mathcal{R}^{\mathcal{H}}_{I}(M)$ follows. As the clause $C$ and inference $I$ have been picked arbitrarily, we conclude $\mathcal{R}^{\mathcal{H}}_{F}(N) \subseteq \mathcal{R}^{\mathcal{H}}_{F}(M)$ and $\mathcal{R}^{\mathcal{H}}_{I}(N) \subseteq \mathcal{R}^{\mathcal{H}}_{I}(M)$.

**Condition (iii).** Let $N$ and $M$ be sets of $\Sigma'$-clauses. Consider an arbitrary hierarchic $\mathcal{H}$-inference $I \in \mathcal{H}(M)$, whose premises *prem*$(I)$ are $C_1, \dots, C_n \in M$, and conclusion *concl*$(I)$ is $C \in N \cup \mathcal{R}^{\mathcal{H}}_{F}(N)$.

If $I$ is a constraint refutation inference, then, by Definition 3.31, the conclusion $C$ is actually an empty clause $\square$. Since there is no clause $D \prec \square$, it holds, according to Definition 3.40, that $\square \notin \mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}})$ for any $\mathcal{A} \in \mathscr{C}$, consequently $\square \notin \mathcal{R}^{\mathcal{H}}_{F}(N)$. Therefore, $\square \in N$, and $I \in \mathcal{R}^{\mathcal{H}}_{I}(N)$ follows by definition of $\mathcal{R}^{\mathcal{H}}_{I}(N)$.

Assume $I$ is not a constraint refutation inference. Pick an arbitrary base algebra $\mathcal{A} \in \mathscr{C}$ and consider an arbitrary simple ground $R^{\approx}_{\mathcal{A}}$-reduced instance $I' \in sgi_{\mathcal{A}}(I)$ of the inference $I$, if any. By Definition 3.63 of a simple ground $R^{\approx}_{\mathcal{A}}$-reduced instance of an inference, premises of $I'$ are clauses $C_1\sigma, \ldots, C_n\sigma$, and the conclusion is a clause $C\sigma$, for some simple grounding $R^{\approx}_{\mathcal{A}}$-reduced substitution $\sigma$. As $C \in N \cup \mathcal{R}^{\mathcal{H}}_F(N)$, we have

$$\begin{aligned}
C\sigma &\in sgi_{\mathcal{A}}(N \cup \mathcal{R}^{\mathcal{H}}_F(N)) \\
&= sgi_{\mathcal{A}}(N) \cup sgi_{\mathcal{A}}(\mathcal{R}^{\mathcal{H}}_F(N)) \\
&\subseteq sgi_{\mathcal{A}}(N) \cup \mathcal{R}^{\mathcal{F}}_F(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} && \text{// by Def. 3.64 of } \mathcal{R}^{\mathcal{H}}_F \\
&= N_{\mathcal{A}} \cup \mathcal{R}^{\mathcal{F}}_F(N_{\mathcal{A}}) && \text{// as } N_{\mathcal{A}} = sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}, \\
& && \quad \text{by Def. 3.56 of } N_{\mathcal{A}}
\end{aligned}$$

yielding, by condition (iii) of Definition 3.1 of an abstract redundancy criterion, that $I' \in \mathcal{R}^{\mathcal{F}}_I(N_{\mathcal{A}})$. Since $I'$ is picked from $sgi_{\mathcal{A}}(I)$ arbitrarily, we conclude $sgi_{\mathcal{A}}(I) \subseteq \mathcal{R}^{\mathcal{F}}_I(N_{\mathcal{A}})$). As $\mathcal{A}$ has been taken arbitrarily from $\mathscr{C}$, we conclude, by definition of $\mathcal{R}^{\mathcal{H}}_I$, that $I \in \mathcal{R}^{\mathcal{H}}_I(N)$.

**Condition (iv)**. Assume $N$ and $M$ are arbitrary sets of $\Sigma'$-clauses, such that $M \subseteq \mathcal{R}^{\mathcal{H}}_F(N)$. Let $C$ be an arbitrary clause in $\mathcal{R}^{\mathcal{H}}_F(N)$, and $I$ an arbitrary inference in $\mathcal{R}^{\mathcal{H}}_I(N)$. Also, suppose $I$ is not a constraint refutation inference. Then by Definition 3.64 of $\mathcal{R}^{\mathcal{H}}$, we know that for any base algebra $\mathcal{A} \in \mathscr{C}$ the following hold:

$$\begin{aligned}
sgi_{\mathcal{A}}(I) &\subseteq \mathcal{R}^{\mathcal{F}}_I(N_{\mathcal{A}}), \\
sgi_{\mathcal{A}}(C) &\subseteq \mathcal{R}^{\mathcal{F}}_F(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}, \text{ and} \\
sgi_{\mathcal{A}}(M) &\subseteq \mathcal{R}^{\mathcal{F}}_F(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}.
\end{aligned}$$

We split the set $sgi_{\mathcal{A}}(M)$ into two parts $M_1$ and $M_2$ defined as follows:

$$\begin{aligned}
M_1 &= sgi_{\mathcal{A}}(M) \cap \mathcal{R}^{\mathcal{F}}_F(N_{\mathcal{A}}), \\
M_2 &= sgi_{\mathcal{A}}(M) \cap (\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}).
\end{aligned}$$

Evidently, $sgi_{\mathcal{A}}(M) = M_1 \cup M_2$. Consider the set $\mathcal{R}^{\mathcal{F}}(N_{\mathcal{A}})$ more elaborately:

$$\begin{aligned}
\mathcal{R}^{\mathcal{F}}(N_{\mathcal{A}}) &\subseteq \mathcal{R}^{\mathcal{F}}(N_{\mathcal{A}} \setminus M_1) && \text{// as } M_1 \subseteq \mathcal{R}^{\mathcal{F}}_F(N_{\mathcal{A}}), \text{ and} \\
& && \quad \text{by cond. (iv) of Def. 3.1} \\
&= \mathcal{R}^{\mathcal{F}}\big((\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N)) \setminus M_1\big) && \text{// by def. of } N_{\mathcal{A}} \\
&= \mathcal{R}^{\mathcal{F}}\big((\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \setminus M_1 \cup sgi_{\mathcal{A}}(N) \setminus M_1\big) \\
&\subseteq \mathcal{R}^{\mathcal{F}}\big(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N) \setminus M_1\big) && \text{// by monoton. of } \mathcal{R}^{\mathcal{F}} \\
&= \mathcal{R}^{\mathcal{F}}\big(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N) \setminus (M_1 \cup M_2)\big) && \text{// as } M_2 \subseteq \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \\
&= \mathcal{R}^{\mathcal{F}}\big(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N) \setminus sgi_{\mathcal{A}}(M)\big) \\
&\subseteq \mathcal{R}^{\mathcal{F}}\big(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N \setminus M)\big) && \text{// by Prop. 3.68 and} \\
& && \quad \text{monoton. of } \mathcal{R}^{\mathcal{F}} \\
&= \mathcal{R}^{\mathcal{F}}\big(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N')\big) && \text{// where } N' = N \setminus M \\
&= \mathcal{R}^{\mathcal{F}}\big(N'_{\mathcal{A}}\big) && \text{// by Def. 3.56}
\end{aligned}$$

Thus, $sgi_{\mathcal{A}}(C) \subseteq \mathcal{R}^{\mathcal{F}}_F(N'_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ and $sgi_{\mathcal{A}}(I) \subseteq \mathcal{R}^{\mathcal{F}}_I(N'_{\mathcal{A}})$ for any $\mathcal{A} \in \mathscr{C}$, where $N' = N \setminus M$. Therefore, by definition of $\mathcal{R}^{\mathcal{H}}$, we obtain $C \in \mathcal{R}^{\mathcal{H}}_F(N \setminus M)$ and $I \in \mathcal{R}^{\mathcal{H}}_I(N \setminus M)$.

If $I$ is a constraint refutation inference, then, by Definition 3.64, $\square \in N$. Since there is no clause $C' \prec \square$, it holds, according to Definition 3.40, that $\square \notin \mathcal{R}^{\mathcal{F}}_F(N_{\mathcal{A}})$

for any $\mathcal{A} \in \mathscr{C}$, consequently $\Box \notin \mathcal{R}_F^{\mathcal{H}}(N)$. Hence, $concl(I) = \Box \in N \setminus \mathcal{R}_F^{\mathcal{H}}(N) \subseteq N \setminus M$. By Condition (iii), proven above, $I \in \mathcal{R}_I^{\mathcal{H}}(N \setminus M)$.

As $C$ and $I$ have been picked arbitrarily, we conclude $\mathcal{R}_F^{\mathcal{H}}(N) \subseteq \mathcal{R}_F^{\mathcal{H}}(N \setminus M)$ and $\mathcal{R}_I^{\mathcal{H}}(N) \subseteq \mathcal{R}_I^{\mathcal{H}}(N \setminus M)$.

The pair $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}_F^{\mathcal{H}}, \mathcal{R}_I^{\mathcal{H}})$ has been shown to comply with every condition of Definition 3.1, thus $\mathcal{R}^{\mathcal{H}}$ is a redundancy criterion. And the proof is complete. ■

### 3.4.5 Lifting and Saturation

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

One of the two main steps towards proving refutational completeness of SUP(T) is to guarantee mutual saturation[1] of a given set $N$ of abstracted clauses with respect to the hierarchic calculus $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$, on one hand, and the respective ground clause set[2] $N_{\mathcal{A}}$ – with respect to the flat (standard) ground calculus $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$, on the other. In the case of the standard superposition calculus SUP, the step is done mainly via so-called *Lifting Lemma*. Roughly speaking, the lemma asserts that for a given set $M$ of general FOL clauses, if there exists a non-redundant ground inference with premises from $gi(M)$, then there exists a non-redundant general inference from $M$, such that the former inference is a ground instance of the latter one. From this it follows that if $M$ is saturated with respect to the SUP calculus for general clauses then $gi(M)$ is saturated with respect to the SUP calculus for ground clauses, because, for otherwise, if there existed a non-redundant inference from $gi(M)$, then, according to the Lifting Lemma, there must exist a non-redundant inference from $M$, contradicting the assumption that $M$ is saturated.

If we want to follow a similar schema in the hierarchic case, then we need a variant of the lifting lemma, that would claim for any non-redundant ground standard inference on $N_{\mathcal{A}}$ existence of the corresponding non-redundant hierarchic inference on $N$. But then we immediately face a problem of ground inferences involving base clauses. Recall that the hierarchic superposition inference rules admit as premises only non-base clauses (except the constraint refutation rule, which anyway does not have a corresponding rule in flat ground calculus). Thus, since a ground base clause can be an instance only of a base clause, we get that for any $\mathcal{F}$-inference with a base premise $C\sigma \in sgi_{\mathcal{A}}(N)$ there is no $\mathcal{H}$-inference with the corresponding base premise $C \in N$, which violates the lifting lemma. Moreover, the set $N_{\mathcal{A}}$ contains clauses, that are not instances of any clause from $N$: these are the clauses in $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ (except custom cases when $N$ and $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ share some clauses).

A possibility to resolve the problem of non-applicability of the standard lifting lemma to the case of ground inferences with base premises is to restrict the statement of the lemma to such inferences that involve only *non-base* premises.

[1]The mutual saturation condition is discussed in the overview to Section 3.4, pages 50-51

[2]Please, recall Definition 3.20 of an abstracted clause, page 41; Definition 3.46 of a rewrite system $R^{\approx}_{\mathcal{A}}$, page 61; Definition 3.54 of a set of all simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances $sgi_{\mathcal{A}}$, page 67; Definitions 3.49 and 3.56 of clause sets $\mathsf{E}_{\mathcal{A}}$, $\mathsf{D}_{\mathcal{A}}$, and $sgi_{\mathcal{A}}(N)$, pages 62 and 67, respectively.

This is what we do in the *Hierarchic Lifting Lemma*, which asserts that for any non-redundant $\mathcal{F}$-inference with *non-base* premises from $N_{\mathcal{A}}$, there exists a non-redundant $\mathcal{H}$-inference with premises in $N$, such that the $\mathcal{F}$-inference is a simple ground $R^{\approx}_{\mathcal{A}}$-reduced instance of the $\mathcal{H}$-inference, Lemma 3.71, where $\mathcal{A} \in \mathscr{C}$ is an arbitrary base algebra. Of course, this is not sufficient to gain mutual saturation of $N$ and $N_{\mathcal{A}}$. In Section 3.4.3, in Lemma 3.51 we have shown that any ground base clause $C$, that is entailed by the base algebra $\mathcal{A}$ and not contained in $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, follows from smaller clauses in $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, thus $C$ is redundant for $N_{\mathcal{A}} = sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, consequently any ground inference with such a base clause $C$ as a premise is also redundant. As it turns out, any ground inference with a premise coming from the set $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ is also redundant. We give a formal proof of these statements in the *Hierarchic Saturation Theorem*, that asserts the mutual saturation of $N$ and $N_{\mathcal{A}}$, where $\mathcal{A} \in \mathscr{C}$ is a base algebra satisfying every base clause in $sgi(N)$, Theorem 3.72. If no base algebra satisfies all base clauses within $sgi(N)$, then due to compactness of the base specification $\mathsf{Sp}$ there are finitely many such base clauses within $sgi(N)$, so that the Constraint Refutation rule produces an empty clause out of the corresponding base clauses in $N$; we shall discuss this issue in Section 3.4.8.

Before we proceed with the proofs, let us recall the following notations:

- $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$ denotes the standard superposition calculus SUP for ground clauses, Definition 3.40 on page 56, where $\mathcal{F}$ the system of the standard inference rules, and $\mathcal{R}^{\mathcal{F}}$ the standard redundancy criterion for ground clauses; see Section 3.4.2 for a detailed exposition of $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$.
- $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$ stands for the hierarchic superposition calculus SUP(T) for general abstracted clauses, Definition 3.64 on page 74, where $\mathcal{H}$ the system of the hierarchic inference rules, and $\mathcal{R}^{\mathcal{H}}$ the hierarchic redundancy criterion; see Sections 3.3 and 3.4.4 for a detailed exposition of $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$.

◄ LEMMA 3.71
Hierarchic Lifting Lemma

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ *and* $\Omega'' = \Omega' \setminus \Omega$ *be the enrichment sorts and operators, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

*Let* $\mathcal{A} \in \mathscr{C}$ *be a base algebra, and* $N$ *a set of abstracted clauses*[1]. *If* $I'$ *is a non-redundant standard* $\mathcal{F}$*-inference with non-base premises in* $N_{\mathcal{A}}$, *then there exists a non-redundant hierarchic* $\mathcal{H}$*-inference* $I$ *with premises in* $N$ *such that* $I'$ *is a simple ground* $R^{\approx}_{\mathcal{A}}$*-reduced instance of* $I$:

$$\begin{aligned} &\textit{for every}^{2}\ I' \in \mathcal{F}(N_{\mathcal{A}} \cap (Cl_{\Sigma'} \setminus Cl_{\Sigma})) \setminus \mathcal{R}^{\mathcal{F}}_{I}(N_{\mathcal{A}}),\\ &\textit{exists an}\ \ I \in \mathcal{H}(N) \setminus \mathcal{R}^{\mathcal{H}}_{I}(N),\\ &\textit{such that}\ \ I' \in sgi_{\mathcal{A}}(I). \end{aligned}$$

[1]Please, recall Definition 3.20 of an abstracted clause, page 41; Definition 3.25 of the entailment relation $\models_{\mathscr{C}}$, page 45; Definition 3.32 of the hierarchic superposition inference system $\mathcal{H}$, page 49; Definition 3.39 of the standard superposition inference system $\mathcal{F}$, page 56; Definition 3.46 of the rewrite system $R^{\approx}_{\mathcal{A}}$, page 61; Definition 3.49 of clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, page 62; Definition 3.55 of sets of all simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances $sgi_{\mathcal{A}}$, page 67; Definition 3.56 of a clause set $N_{\mathcal{A}}$, page 67; Definitions 3.64 and 3.40 of the hierarchic $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_{F}, \mathcal{R}^{\mathcal{H}}_{I})$ and the standard ground $\mathcal{R}^{\mathcal{F}} = (\mathcal{R}^{\mathcal{F}}_{F}, \mathcal{R}^{\mathcal{F}}_{I})$ redundancy criterions, pages 74 and 56, respectively.

PROOF ► We demonstrate the lemma's statement in detail for the equality resolution (Definition 3.37 and 3.29) and the superposition left (Definition 3.35 and 3.27) inference rules; analysis of the other rules is similar.

First, let us make the following very important observations. Let $C$ be an abstracted clause. If a simple ground instance $C'$ of $C$ is non-base, then $C$ is non-base as well, i.e. it contains a literal with an occurrence of an operator from $\Omega''$ or a variable from $\mathcal{X}''$. Moreover, we claim that if some literal $L'$ is (strictly) maximal in $C'$ or if it is selected, then the corresponding literal $L$ in $C$ is non-base as well (recall that $\mathcal{F}$ admits selection only of *non-base* negative literals). Indeed, any ground non-base literal is strictly greater than any base one, hence $L'$ is non-base, and besides, ground non-base literals can be simple instances only of non-base literals, because simple substitutions map base variables to base terms, so a simple instance of a base literal is always base, therefore $L$ is non-base too. Also, since the set $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ consists only of base clauses, the set of non-base clauses in $N_{\mathcal{A}} = \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N)$ is actually a subset of $sgi_{\mathcal{A}}(N)$, therefore each premise of any inference $I' \in \mathcal{F}(N_{\mathcal{A}} \cap (Cl_{\Sigma'} \setminus Cl_{\Sigma}))$ is a simple ground $R^{\approx}_{\mathcal{A}}$-reduced instance of some non-base clause in $N$.

As already stated in Section 3.4.2 devoted to the standard superposition for ground clauses, the inference systems $\mathcal{H}$ and $\mathcal{F}$ share the same selection function, i.e. a literal $L'$ is selected in $C' \in sgi_{\mathcal{A}}(C)$ if and only if the corresponding literal $L$ is selected in $C \in N$.

Consider a non-redundant ground **equality resolution** inference $I'$:

$$\mathcal{I}\frac{\Gamma', t' \approx t' \to \Delta'}{\Gamma' \to \Delta'}$$

with a premise $C'_1 \in N_{\mathcal{A}} \cap (Cl_{\Sigma'} \setminus Cl_{\Sigma})$, and conclusion $C'_0$, respectively. From the observation above, we know $C'_1 \in sgi_{\mathcal{A}}(C_1)$, for some non-base clause $C_1 \in N$. We show now that there is a Hierarchic Equality Resolution inference on $C_1$ by ensuring that all conditions in Definition 3.29 are satisfied. Without loss of generality, assume

$$C_1 = \Lambda \parallel \Gamma, s \approx t \to \Delta,$$

where $\Lambda = \Pi \cup \Upsilon$ consists of positive $\Pi$ and negative $\Upsilon$ base literals. As $C'_1 \in sgi_{\mathcal{A}}(C_1)$, there exists an $R^{\approx}_{\mathcal{A}}$-reduced substitution $\theta$ such that $C'_1 = C_1\theta$, particularly

- $\Gamma' = \Pi\theta \cup \Gamma\theta$,
- $\Delta' = \Upsilon\theta \cup \Delta\theta$, and
- $(t' \approx t') = (s \approx t)\theta$.

The atom $(s \approx t)$ appears in the free part of $C_1$, as according to condition (i) in the definition of the ground rule the atom $t' \approx t'$ is maximal, and, therefore, by the observation we have made in the beginning of the proof, $s \approx t$ is non-base as well. The substitution $\theta$ is a simple ground $R^{\approx}_{\mathcal{A}}$-reduced unifier of $s$ and $t$, hence there exists a simple most general unifier $\sigma = mgu(s,t)$ of $s$ and $t$. This satisfies

[2]Recall that $Cl_{\Sigma'}$ denotes the class of all clauses over the body signature $\Sigma'$, and $Cl_{\Sigma}$ the class of all clauses over the base signature $\Sigma$. Thus, $Cl_{\Sigma}$ is the class of all base clauses, and $Cl_{\Sigma'} \setminus Cl_{\Sigma}$ stands for the class of all non-base clauses.

condition (i) of the hierarchic rule. Moreover, $\theta = \sigma\psi$ for some simple grounding substitution $\psi$; this fact and conditions (i) and (ii) of the ground rule directly yield condition (ii) of the hierarchic rule is satisfied as well. Thus, there is a Hierarchic Equality Resolution inference $I$

$$\mathcal{I}\frac{\Lambda \parallel \Gamma, s \approx t \rightarrow \Delta}{(\Lambda \parallel \Gamma \rightarrow \Delta)\sigma}$$

with premise $C_1$ and conclusion $C_0$ such that $C'_i = C_i\theta$, for every $i \in \{0,1\}$, therefore $I' \in sgi_{\mathcal{A}}(I)$. Moreover,

$$\begin{array}{rll} & I' \notin \mathcal{R}^{\mathcal{F}}_I(N_{\mathcal{A}}) & \text{// by lemma's cond.}\\ \Rightarrow & sgi_{\mathcal{A}}(I) \nsubseteq \mathcal{R}^{\mathcal{F}}_I(N_{\mathcal{A}}) & \text{// as } I' \in sgi_{\mathcal{A}}(I)\\ \Rightarrow & I \notin \mathcal{R}^{\mathcal{H}}_I(N) & \text{// by Def. 3.64 of } \mathcal{R}^{\mathcal{H}}_I \end{array}$$

Consequently, $I \in \mathcal{H}(N) \setminus \mathcal{R}^{\mathcal{H}}_I(N)$.

Let $I'$ be a non-redundant ground **superposition left** inference

$$\mathcal{I}\frac{\Gamma'_1 \rightarrow \Delta'_1, l'' \approx r' \qquad s'[l''] \approx t', \Gamma'_2 \rightarrow \Delta'_2}{s'[r'] \approx t', \Gamma'_1, \Gamma'_2 \rightarrow \Delta'_1, \Delta'_2}$$

with premises $C'_1, C'_2 \in N_{\mathcal{A}} \cap (Cl_{\Sigma'} \setminus Cl_{\Sigma})$, and conclusion $C'_0$, respectively. By the same argumentation as in the case of an equality resolution inference, we learn $C'_1 \in sgi_{\mathcal{A}}(C_1)$ and $C'_2 \in sgi_{\mathcal{A}}(C_2)$, for some non-base variable-disjoint clauses $C_1, C_2 \in N$. Without loss of generality, assume

$$\begin{aligned} C_1 &= \Lambda_1 \parallel \Gamma_1 \rightarrow \Delta_1, l \approx r,\\ C_2 &= \Lambda_2 \parallel s[l'] \approx t, \Gamma_2 \rightarrow \Delta_2, \end{aligned}$$

where $\Lambda_i = \Pi_i \cup \Upsilon_i$ consists of positive $\Pi_i$ and negative $\Upsilon_i$ base literals, for every $i \in \{1,2\}$. As $C'_1 \in sgi_{\mathcal{A}}(C_1)$ and $C'_2 \in sgi_{\mathcal{A}}(C_2)$, there exist two simple grounding $R^{\approx}_{\mathcal{A}}$-reduced substitutions $\theta_1$ and $\theta_2$ such that $C'_1 = C_1\theta_1$ and $C'_2 = C_2\theta_2$. As $C_1$ and $C_2$ are variable-disjoint, there exists a simple grounding $R^{\approx}_{\mathcal{A}}$-reduced substitution $\theta$ such that $C'_1 = C_1\theta$ and $C'_2 = C_2\theta$, particularly

- $\Gamma'_i = \Pi_i\theta \cup \Gamma_i\theta$,
- $\Delta'_i = \Upsilon_i\theta \cup \Delta_i\theta$,

for every $i \in \{1,2\}$, and

- $(l'' \approx r') = (l \approx r)\theta$,
- $(s'[l''] \approx t') = (s[l'] \approx t)\theta$.

We have to distinguish two cases:

- the overlap in $I'$ takes place at or below a variable position, i.e. $l''$ occurs in $s'$ at a position $p = p_1p_2$, and $s/p_1$ is some variable $x$. Let $s''$ be the sub-term of $s'$ at the position $p_1$, that is $s'' = s'/p_1$, then $s''/p_2 = l''$. We define a substitution $\tau$ as follows:

$$\tau(y) = \begin{cases} s''[r']_{p_2}, & \text{if y = x,}\\ \theta(y), & \text{oth.,}\end{cases}$$

  i.e. $\tau$ is a modification of the $\theta$, such that for every variable different from $x$, we take the respective value of $\theta$, and for $x$ we take $s''$ with the occurrence of $l''$ at the position $p_2$ in it replaced with $r'$ (note that there can be more than one occurrence of $l''$ in $s''$, but we replace only the one at $p_2$). Evidently, $C_2\tau \prec C_2\theta$ and $C_2\tau \in sgi_{\mathcal{A}}(C_2)$. It is easy to see that $C'_1, C_2\tau \models C'_0$. These mean that $I'$ is actually redundant for $N_{\mathcal{A}}$, which contradicts the lemma's assumption, hence the case considered is impossible.

– otherwise, $l''$ occurs in $s'$ at some position $p$, such that $p$ is also a position in $s$ and $s/p = l$ is not a variable. The substitution $\theta$ is a simple ground $R^{\approx}_{\mathcal{A}}$-reduced unifier of $l$ and $l'$, hence there exists a simple most general unifier $\sigma = mgu(l, l')$ of $l$ and $l'$. This satisfies condition (i) of the hierarchic rule. By assumption, $l'$ is not a variable, satisfying condition (ii). Moreover, $\theta = \sigma\psi$ for some simple grounding substitution $\psi$; this fact and conditions (i)-(v) of the ground rule directly yield conditions (iii)-(vii) of the hierarchic rule are satisfied as well. Thus, there is a Hierarchic Superposition Left inference $I$

$$\mathcal{I}\frac{\Lambda_1 \parallel \Gamma_1 \to \Delta_1, l \approx r \qquad \Lambda_2 \parallel s[l'] \approx t, \Gamma_2 \to \Delta_2}{(\Lambda_1, \Lambda_2 \parallel s[r] \approx t, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2)\sigma}$$

with premises $C_1$, $C_2$ and conclusion $C_0$ such that $C'_i = C_i\theta$, for every $i \in \{0,1,2\}$, therefore $I' \in sgi_{\mathcal{A}}(I)$. Analogously to the case of an equality resolution inference, we also conclude $I \in \mathcal{H}(N) \setminus \mathcal{R}^{\mathcal{H}}_I(N)$.

■

THEOREM 3.72 ► Hierarchic Saturation Theorem

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $\mathcal{A} \in \mathscr{C}$ be a base algebra, and $N$ a set of abstracted clauses*[1]. *If $\mathcal{A}$ satisfies the set $N \cap Cl_\Sigma$ of all base*[2] *clauses within $N$, and $N$ is saturated with respect to $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$, then $N_{\mathcal{A}}$ is saturated with respect to $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$.*

PROOF ► We have to show that if $N$ is saturated then every $\mathcal{F}$-inference from clauses in $N_{\mathcal{A}}$ is redundant with respect to $N_{\mathcal{A}}$, i.e. that it is contained in $\mathcal{R}^{\mathcal{F}}_I(N_{\mathcal{A}})$.

Let us split the set $N_{\mathcal{A}} = \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(N)$ into three subsets $M_1$, $M_2$, and $M_3$ defined as follows:

$$\begin{array}{ll}
M_1 = N_{\mathcal{A}} \cap (Cl_{\Sigma'} \setminus Cl_\Sigma) & \text{(the subset of non-base clauses)} \\
M_2 = N_{\mathcal{A}} \cap (Cl_\Sigma \setminus (\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})) & \text{(the subset of base clauses} \\
 & \text{ not contained in } \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \\
M_3 = N_{\mathcal{A}} \cap (\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) & \text{(the subset of base clauses} \\
\quad\;\; = \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \text{ contained in } \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})
\end{array}$$

Evidently, $N_{\mathcal{A}} = M_1 \cup M_2 \cup M_3$. Consider the following three possible types of $\mathcal{F}$-inferences on $N_{\mathcal{A}}$:

1. inferences with *every* premise in $M_1$,

[1]Please, recall Definition 3.20 of an abstracted clause, page 41; Definition 3.25 of the entailment relation $\models_{\mathscr{C}}$, page 45; Definition 3.32 of the hierarchic superposition inference system $\mathcal{H}$, page 49; Definition 3.39 of the standard superposition inference system $\mathcal{F}$, page 56; Definition 3.46 of the rewrite system $R^{\approx}_{\mathcal{A}}$, page 61; Definition 3.49 of clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, page 62; Definition 3.55 of sets of all simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances $sgi_{\mathcal{A}}$, page 67; Definition 3.56 of a clause set $N_{\mathcal{A}}$, page 67; Definitions 3.64 and 3.40 of the hierarchic $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}^{\mathcal{H}}_F, \mathcal{R}^{\mathcal{H}}_I)$ and the standard ground $\mathcal{R}^{\mathcal{F}} = (\mathcal{R}^{\mathcal{F}}_F, \mathcal{R}^{\mathcal{F}}_I)$ redundancy criterions, pages 74 and 56, respectively.

[2]Recall that $Cl_\Sigma$ denotes the class of all clauses over the base signature $\Sigma$, and $Cl_{\Sigma'}$ the class of all clauses over the body signature $\Sigma$. Thus, $Cl_\Sigma$ is the class of all base clauses, and $Cl_{\Sigma'} \setminus Cl_\Sigma$ stands for the class of all non-base clauses.

2. inferences with *at least one* premise in $M_2$, and
3. inferences with *at least one* premise in $M_3$.

From Lemma 3.71 it follows that if there is a non-redundant $\mathcal{F}$-inference with premises in $M_1 = N_{\mathcal{A}} \cap (Cl_{\Sigma'} \setminus Cl_{\Sigma})$, then there also exists a non-redundant $\mathcal{H}$-inference with premises in $N$. Hence, any $\mathcal{F}$-inference with *non-base* premises is redundant, as for otherwise it would be contradictory to the assumption that $N$ is saturated.

The set $M_2 = N_{\mathcal{A}} \cap (Cl_{\Sigma} \setminus (\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}))$ is obviously a subset of $sgi(N) \cap Cl_{\Sigma} = sgi(N \cap Cl_{\Sigma})$, which is true under $\mathcal{A}$ by assumption. According to Lemma 3.51, every base clause, that is true in $\mathcal{A}$ and not contained in $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, is entailed by strictly smaller clauses in $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, thus it is in $\mathcal{R}^{\mathcal{F}}_{F}(N_{\mathcal{A}})$; any inference involving such a clause is in $\mathcal{R}^{\mathcal{F}}_{I}(N_{\mathcal{A}})$. Therefore, any inference with a premise from $M_2$ is redundant.

So, what is left to consider, is the case of an $\mathcal{F}$-inference with a premise from $M_3 = \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$. We demonstrate this in detail for the equality resolution (Definition 3.37) and the left superposition rules (Definition 3.35); the analysis of the other rules is quite similar.

The ground **equality resolution** rule is not applicable to clauses from $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, because every clause from $\mathsf{E}_{\mathcal{A}}$ is a *positive* unit clause, and every clauses from $\mathsf{D}_{\mathcal{A}}$ is a negative unit clause, whose the only literal is a disequation between two *distinct ground* $\Sigma$-terms (Definition 3.49 of $\mathsf{D}_{\mathcal{A}}$), hence, not unifiable.

Let $I'$ be a ground **superposition left** inference

$$\mathcal{I}\,\frac{\Gamma'_1 \to \Delta'_1, l'' \approx r' \qquad s'[l''] \approx t', \Gamma'_2 \to \Delta'_2}{s'[r'] \approx t', \Gamma'_1, \Gamma'_2 \to \Delta'_1, \Delta'_2}$$

with premises $C'_1$ and $C'_2$, and conclusion $C'_0$, respectively. Obviously, a clause from $\mathsf{D}_{\mathcal{A}}$ cannot be the first premise of $I$. Assume the first premise is a clause from $\mathsf{E}_{\mathcal{A}}$, say

$$C'_1 = (\quad \to l'' \approx r'),$$

then, according to Definition 3.49 of $\mathsf{E}_{\mathcal{A}}$, we know $l'' \neq r'$ and $r' = l''{\downarrow_{R^{\approx}_{\mathcal{A}}}}$, thus $l'' \neq l''{\downarrow_{R^{\approx}_{\mathcal{A}}}}$, i.e. $l''$ is not $R^{\approx}_{\mathcal{A}}$-reduced (we shall use this fact later on in the proof). The second premise $C'_2$ must be then a base clause. Indeed, suppose $C'_2$ is a non-base clause. Then there exists a non-base clause $C_2 \in N$ defined as

$$C_2 = \Lambda_2 \parallel s[l'] \approx t, \Gamma_2 \to \Delta_2$$

where $\Lambda_2 = \Pi \cup \Upsilon$ consists of positive $\Pi$ and negative $\Upsilon$ base literals, such that $C'_2 = C_2\theta$, for some simple grounding $R^{\approx}_{\mathcal{A}}$-reduced substitution $\theta$, particularly:

- $\Gamma'_2 = \Pi\theta \cup \Gamma_2\theta$,
- $\Delta'_2 = \Upsilon\theta \cup \Delta_2\theta$,
- $(s'[l''] \approx t') = (s[l'] \approx t)\theta$.

The equation $s[l'] \approx t$ does not belong to $\Lambda_2$, because (i) $s' \approx t'$ is maximal in the *non-base* clause $C'_2$, (ii) any ground base literal is strictly smaller than any ground non-base literal, hence $s' \approx t'$ is non-base, and (iii) no simple substitution can introduce free symbols to a base term, therefore $s[l'] \approx t$ is non-base. Since $C_2$ is an abstracted clause, every base-sorted symbol occurring in $s$ is either a base variable or a free operator. Since $\theta$ is $R^{\approx}_{\mathcal{A}}$-reduced, every base subterm occurring in $s' = s\theta$ is also $R^{\approx}_{\mathcal{A}}$-reduced, and so is $l''$, contradicting the fact that $l''$ is not reduced, shown just above. For the same reason, $C_2$ is not a clause from $\mathsf{D}_{\mathcal{A}}$, because by the definition of $\mathsf{D}_{\mathcal{A}}$ the hand-sides of the only disequation in each clause from $\mathsf{D}_{\mathcal{A}}$ are in their normal forms, or, equivalently speaking, $R^{\approx}_{\mathcal{A}}$-reduced. Thus, $C_2'$ is a base clause not contained in $\mathsf{E}_{\mathcal{A}}$, nor in $\mathsf{D}_{\mathcal{A}}$. By the observation above regarding inferences involving base clauses, we conclude the inference $I'$ is redundant.

Now, assume the second premise $C_2'$ of $I'$ is a clause from $\mathsf{D}_{\mathcal{A}}$ (obviously, $C_2'$ cannot be a clause from $\mathsf{E}_{\mathcal{A}}$), say

$$C_2' = (s'[l''] \approx t' \rightarrow \quad).$$

As the term $l''$ is base, $l'' \approx r'$ is strictly maximal in the first premise $C_1'$ of $I'$, and $l'' \succ r'$, we conclude $C_1'$ is a base clause. If $C_1' \in \mathsf{E}_{\mathcal{A}}$ then the case considered is impossible (as we have shown just above), for otherwise $C_1' \in M_2$, which implies the inference $I'$ is redundant, as we have also shown above. ■

### 3.4.6 Weak Algebras

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

In Sections 3.4.4-3.4.5 we have shown how the mutual saturation condition[1] is satisfied for the hierarchic SUP(T) calculus. Besides the mutual saturation condition, we have also to ensure the mutual hierarchic model existence condition. One of its preconditions is that a model of a set $N_{\mathcal{A}} = sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ has to be a model of $N$, where $\mathcal{A} \in \mathscr{C}$ is a base algebra. To support the condition, we have restricted the class of admissible clauses sets to ones possessing the sufficient completeness property[2], according to which in every model of $sgi(N)$ each non-base term[3] $t'$ of a base sort $\mathsf{S} \in \mathcal{S}$ has to be interpreted as some base term $t$:

$$\forall\, t' \in T_{\Omega'}(\mathcal{S}) \setminus T_{\Omega}. \exists\, t \in T_{\Omega} : sgi(N) \models t' \approx t.$$

However, such formulation of the property is superfluous and too restrictive, because we need this property to hold only for those models of $sgi(N)$ that satisfy also the set $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, and it suffices to ensure the property only for ground smooth extension terms. Thus, the formulation can be refined as follows:

$$\forall\, t' \in T^{E}_{\Omega'}. \exists\, t \in T_{\Omega} : t' \text{ smooth } \Rightarrow sgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \models t' \approx t,$$

for all base algebras $\mathcal{A} \in \mathscr{C}$, where $T^{E}_{\Omega'}$ is the set of all ground extension terms.

Here, we introduce a notion of an algebra weak relative to the base class $\mathscr{C}$. A $\Sigma'$-algebra $\mathcal{A}'$ is weak relative to $\mathscr{C}$ if it *monomorphically* extends some base algebra $\mathcal{A} \in \mathscr{C}$, Definition 3.73, meaning that $\mathcal{A}'$ has to extend $\mathcal{A}$ in a structure-

[1]The mutual saturation condition and the mutual hierarchic model existence condition are discussed in the overview to Section 3.4 on pages 50-51

[2]The sufficient completeness property has been informally discussed in the overview to Section 3.4; it will be formally defined in Section 3.4.7 entirely dedicated to presenting the property.

[3]Please, recall Definitions 3.16 and 3.17 of different kinds of terms (base, non-base, etc.), page 39.

preserving way mapping distinct elements of the universe $U_{\mathcal{A}}$ of $\mathcal{A}$ to distinct elements of own universe $U_{\mathcal{A}'}$. In the *Weak Algebra Theorem* we assert that an algebra is weak if and only if it is a model of the set $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ for some base algebra $\mathcal{A}$, Theorem 3.76. Thus, the sufficient completeness criterion can be reformulated in terms of weak algebras as follows:

$$\forall \mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}}. \forall t' \in T^{E}_{\Omega'}. \exists t \in T_{\Omega}(\mathsf{S}) : \mathcal{A}' \models sgi(N),\ t' \text{ smooth } \Rightarrow \mathcal{A}' \models t' \approx t,$$

where $\mathscr{W}_{\mathsf{HSp}}$ is the class of all weak algebras relative to $\mathscr{C}$.

◄ DEFINITION 3.73
$\mathscr{C}$-Weak Algebra

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

A $\Sigma'$-algebra $\mathcal{A}' \in \mathscr{A}_{\Sigma'}$ is[1] called ***weak relative to*** $\mathscr{C}$ (or, shortly, $\mathscr{C}$-weak algebra), if there exists a base algebra $\mathcal{A} \in \mathscr{C}$ monomorphic[2] to $\mathcal{A}'$. We write $\mathscr{W}_{\mathsf{HSp}}$ to denote the class of all algebras weak with respect to $\mathscr{C}$:

$$\mathscr{W}_{\mathsf{HSp}} \overset{\text{def}}{=} \{\mathcal{A}' \in \mathscr{A}_{\Sigma'} \mid \exists \mathcal{A} \in \mathscr{C} : \mathcal{A} \text{ monomorphic to } \mathcal{A}'\}.$$

■

From now on, as the class $\mathscr{C}$ of base algebras is given beforehand, if $\mathcal{A}'$ is a weak algebra with respect to $\mathscr{C}$, we call it just *weak*, for the sake of conciseness.

◄ DEFINITION 3.74
$\mathscr{W}_{\mathsf{HSp}}$-consistent
$\mathscr{W}_{\mathsf{HSp}}$-entailment

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

Let $N$ and $M$ be two sets of $\Sigma'$-clauses. We call $N$ ***consistent relative to*** $\mathscr{W}_{\mathsf{HSp}}$ (or, shortly, $\mathscr{W}_{\mathsf{HSp}}$***-consistent***), if there exists a weak model of $N$:

$$N\ \mathscr{W}_{\mathsf{HSp}}\text{-consistent} \overset{\text{def}}{\iff} \exists \mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}} : \mathcal{A}' \models N,$$

and otherwise we call it ***inconsistent relative to*** $\mathscr{W}_{\mathsf{HSp}}$ (or, shortly, $\mathscr{W}_{\mathsf{HSp}}$***-inconsistent***), written $N \models_{\mathscr{W}} \bot$.

We say $N$ ***implies/entails*** $M$ ***relative to*** $\mathscr{W}_{\mathsf{HSp}}$, written $N \models_{\mathscr{W}} M$, if every weak model of $N$ is also a model of $M$:

$$N \models_{\mathscr{W}} M \overset{\text{def}}{\iff} \forall \mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}} : \mathcal{A}' \models N \Rightarrow \mathcal{A}' \models N.$$

■

Clearly, the set of all hierarchic algebras is a subset of the set of all weak algebras, where the corresponding monomorphism is the identity. Consequently, $N \models_{\mathscr{W}} M$ implies $N \models_{\mathscr{C}} M$, but not the other way around, in general.

Since the class $\mathscr{C}$ consists of base algebras that all are term-generated, from the definition above it follows that an algebra $\mathcal{A}'$ is weak with respect to $\mathscr{C}$, iff there exists a base algebra $\mathcal{A} \in \mathscr{C}$ and a monomorphism $h : U_{\mathcal{A}} \to U_{\mathcal{A}'}$ from $\mathcal{A}$ to $\mathcal{A}'$, such that

$$f_{\mathcal{A}'}\big(h((t_1)_{\mathcal{A}}), \ldots, h((t_n)_{\mathcal{A}})\big) = h\big((f(t_1, \ldots, t_n))_{\mathcal{A}}\big)$$

for all ground base terms $f(t_1, \ldots, t_n) \in T_{\Omega}$, where $f \in \Omega$ a base operator symbol of arity $n \geq 0$.

[1] $\mathscr{A}_{\Sigma'}$ denotes the class of all $\Sigma'$-algebras.
[2] Recall the definition of a monomorphic algebra in Definition 2.31.

Note, that our definition of a weak algebra significantly differs from the one given in [AKW09b], where we required for a weak algebra $\mathcal{A}'$ to have a *homomorphism* (not a *monomorphism*!) $h : U_{\mathcal{A}} \to U_{\mathcal{A}'}$ for *each* base algebra $\mathcal{A} \in \mathscr{C}$, which is actually a very strong requirement. In Definition 3.73 we soften the requirement to existence of *at least one* base algebra *monomorphic to* $\mathcal{A}'$.

Our main aim regarding weak algebras is to prove that an algebra $\mathcal{A}'$ is weak relative to $\mathscr{C}$ *if and only if* $\mathcal{A}'$ is a model of the clause set $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, for some base algebra $\mathcal{A} \in \mathscr{C}$. For doing so we need one further property on interpretation of base terms under weak algebras given in the following proposition.

PROPOSITION 3.75 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*Let* $\mathcal{A}'$ *be a weak algebra,* $\mathcal{A} \in \mathscr{C}$ *a base algebra homomorphic to* $\mathcal{A}'$*, and* $h$ *a homomorphism from* $\mathcal{A}$ *to* $\mathcal{A}'$*. For all ground base terms* $t \in T_\Omega$ *it holds that* $t_{\mathcal{A}'} = h(t_{\mathcal{A}})$*.*

PROOF ► We give a proof by induction on depth of $t$.
**Induction base**. If $depth(t) = 0$, then $t$ is a base constant, say $a \in \Omega$. Thus,

$$\begin{aligned} t_{\mathcal{A}'} &= a_{\mathcal{A}'} \\ &= h(a_{\mathcal{A}}) && \text{// as } h \text{ is a hom. from } \mathcal{A} \text{ to } \mathcal{A}' \\ &= h(t_{\mathcal{A}}). \end{aligned}$$

**Induction hypothesis**. Assume the statement holds for any ground base term $t \in T_\Omega$, such that $depth(t) \le k$, for some $k \ge 0$.

**Induction step**. Consider a term $t$ with $depth(t) = k+1$. Without loss of generality, assume $t = f(t_1, \ldots, t_n)$, for some base operator symbol $f \in \Omega$ of arity $n \ge 0$, then

$$\begin{aligned} t_{\mathcal{A}'} &= f_{\mathcal{A}'}\big((t_1)_{\mathcal{A}'}, \ldots, (t_n)_{\mathcal{A}'}\big) \\ &= f_{\mathcal{A}'}\big(h((t_1)_{\mathcal{A}}), \ldots, h((t_n)_{\mathcal{A}})\big) && \text{// by ind. hyp.} \\ &= h\big(f_{\mathcal{A}}\big((t_1)_{\mathcal{A}}, \ldots, (t_n)_{\mathcal{A}}\big)\big) && \text{// as } h \text{ is a hom. from } \mathcal{A} \text{ to } \mathcal{A}' \\ &= h(t_{\mathcal{A}}) \end{aligned}$$

■

THEOREM 3.76 ► Weak Algebra Theorem

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*A* $\Sigma'$*-algebra* $\mathcal{A}'$ *is weak if and only if*[1] $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ *is satisfiable in* $\mathcal{A}'$*, for some base algebra*[2] $\mathcal{A} \in \mathscr{C}$*:*

$$\mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}} \Leftrightarrow \exists \mathcal{A} \in \mathscr{C} : \mathcal{A}' \models \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}.$$

PROOF ► **The "⇒" direction**. Assume, $\mathcal{A}'$ is a weak algebra. According to Definition 3.73, there exists a base algebra $\mathcal{A} \in \mathscr{C}$ that is monomorphic to $\mathcal{A}'$, hence homomorphic

[1] Please, recall Definition 3.49 of a clause set $\mathsf{E}_{\mathcal{A}}$, page 62.

[2] Please, note that the base algebra $\mathcal{A}$ is only used for construction of clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$ and is not a subject to any logical entailment condition, in contrast to $\mathcal{A}'$ which has to entail $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ with respect to the general entailment relation $\models$.

to $\mathcal{A}'$. Assume $h$ is a monomorphism from $\mathcal{A}$ to $\mathcal{A}'$, hence a homomorphism from $\mathcal{A}$ to $\mathcal{A}'$.

Let $C$ be an arbitrary clause from $\mathsf{E}_{\mathcal{A}}$; without loss of generality, represent it as $C = (\ \rightarrow l \approx r)$, for some ground base terms $l, r \in T_\Omega$; then

$$\begin{aligned} l_{\mathcal{A}'} &= h(l_{\mathcal{A}}) && \text{// by Prop. 3.75} \\ &= h(r_{\mathcal{A}}) && \text{// as } \mathcal{A} \models C \text{, hence } l_{\mathcal{A}} = r_{\mathcal{A}} \\ &= r_{\mathcal{A}'} && \text{// by Prop. 3.75} \end{aligned}$$

Let $D$ be an arbitrary clause from $\mathsf{D}_{\mathcal{A}}$; without loss of generality, represent it as $D = (l' \approx r' \rightarrow\ )$, for some ground base terms $l', r' \in T_\Omega$. As $\mathcal{A} \models \mathsf{D}_{\mathcal{A}}$, we know $l'_{\mathcal{A}} = e_1 \neq e_2 = r'_{\mathcal{A}}$, for some $e_1, e_2 \in U_{\mathcal{A}}$. Consequently,

$$\begin{aligned} l'_{\mathcal{A}'} &= h(l'_{\mathcal{A}}) && \text{// by Prop. 3.75} \\ &\neq h(r'_{\mathcal{A}}) && \text{// as } l'_{\mathcal{A}} \neq r'_{\mathcal{A}} \text{ and } h \text{ injective} \\ &= r'_{\mathcal{A}'} && \text{// by Prop. 3.75} \end{aligned}$$

Thus, $\mathcal{A}' \models C, D$. As $C$ and $D$ have been picked arbitrarily, it follows $\mathcal{A}' \models \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$.

**The "$\Leftarrow$" direction**. Assume, $\mathcal{A}'$ is a $\Sigma'$-algebra that satisfies $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, for some base algebra $\mathcal{A} \in \mathscr{C}$. We show that $\mathcal{A}'$ is a weak algebra by finding a mapping $h$ that is a monomorphism (injective homomorphism) from $\mathcal{A}$ to $\mathcal{A}'$.

Every algebra in $\mathscr{C}$ is term-generated, hence for every element $e \in U_{\mathcal{A}}$ in the universe of $\mathcal{A}$ there exists a ground base term $t \in T_\Omega$ such that $\mathcal{A}(t) = e$. According to Definition 3.45, $[t]_{\mathcal{A}}$ denotes the equivalence class of $t$ with respect to $\mathcal{A}$; more formally:

$$[t]_{\mathcal{A}} \overset{\text{def}}{=} \{s \in T_\Omega \mid \mathcal{A}(s) = \mathcal{A}(t)\}.$$

As usual, we also write $[t]_{\mathcal{A}}$ to denote a single representative of the class. Recall, that for any $s, s', t \in T_\Omega$, if $s, s' \in [t]_{\mathcal{A}}$ then $\mathsf{E}_{\mathcal{A}} \models s \approx s'$, which is a consequence of Corollary 3.50. Recall also, that interpretation of the base sorts $\mathcal{S}$ constitute the universe of $\mathcal{A}$:

$$U_{\mathcal{A}} = \bigcup_{\mathsf{S} \in \mathcal{S}} \mathsf{S}_{\mathcal{A}}.$$

We define a mapping $h : U_{\mathcal{A}} \mapsto U_{\mathcal{A}'}$ as follows: for every base sort $\mathsf{S} \in \mathcal{S}$, and every element $e \in \mathsf{S}_{\mathcal{A}}$:

$$h(e) \overset{\text{def}}{=} \mathcal{A}'([t]_{\mathcal{A}}),$$

where[1] $t \in T_\Omega$ is a ground base term such that $\mathcal{A}(t) = e$. That is, for every element $e$ in $U_{\mathcal{A}}$, the mapping $h$ takes an arbitrary ground base term, whose interpretation under $\mathcal{A}$ equals $e$, and returns its interpretation under $\mathcal{A}'$. Since $\mathcal{A}$ is term-generated, $h$ is total, i.e. $h$ is defined for every $e \in U_{\mathcal{A}}$. We next show that $h$ is well-defined, that is $h$ returns the same value regardless of a particular choice of a representative of the equivalence class $[t]_{\mathcal{A}}$. Let $s, s' \in [t]_{\mathcal{A}}$ be arbitrary distinct representatives of the class. As we have observed above, $\mathsf{E}_{\mathcal{A}} \models s \approx s'$. Since $\mathcal{A}' \models \mathsf{E}_{\mathcal{A}}$, we have $\mathcal{A}' \models s \approx s'$, consequently $\mathcal{A}'(s) = \mathcal{A}'(s') = e'$ for some $e' \in U_{\mathcal{A}'}$. As $s$ and $s'$ have been picked arbitrarily, all representatives of $[t]_{\mathcal{A}}$ have the same

[1] Please, note that $[t]_{\mathcal{A}}$ denotes the equivalence class of $t$ under $\mathcal{A}$ or an arbitrary representative of the class, whereas $t_{\mathcal{A}}$ denotes the interpretation of $t$ under $\mathcal{A}$; thus, the notions $[t]_{\mathcal{A}}$ and $t_{\mathcal{A}}$ should not be confused!

value $e'$ under $\mathcal{A}'$, thus $h$ is well-defined. Moreover, all terms in $[t]_{\mathcal{A}}$ are of the same sort $\mathsf{S} = sort(t)$, hence $h(e) \in \mathsf{S}_{\mathcal{A}'}$, consequently, $h(\mathsf{S}_{\mathcal{A}}) \subseteq \mathsf{S}_{\mathcal{A}'}$ for every base sort $\mathsf{S} \in \mathcal{S}$.

Now, we show $h$ is structure-preserving. Let $f \in \Omega$ be an arbitrary base operator symbol of arity $n \geq 0$, such that $f : \mathsf{S}_1 \times \ldots \times \mathsf{S}_n \to \mathsf{S}$, and $e_1, \ldots, e_n$ arbitrary elements of the universe of $\mathcal{A}$, such that $e_i \in (\mathsf{S}_i)_{\mathcal{A}}$ for every $i \in \{1, \ldots, n\}$, then

$$\begin{aligned}
& f_{\mathcal{A}'}\big(h(e_1), \ldots, h(e_n)\big) \\
&\quad = f_{\mathcal{A}'}\big(\mathcal{A}'([t_1]_{\mathcal{A}}), \ldots, \mathcal{A}'([t_n]_{\mathcal{A}})\big) && \text{// by def. of } h \text{, for some } t_1, \ldots, t_n \in T_\Omega \\
&&& \quad \text{s.t. } \mathcal{A}(t_i) = e_i \text{ for all } i \in \{1, \ldots, n\} \\
&\quad = f_{\mathcal{A}'}(\mathcal{A}'(t_1), \ldots, \mathcal{A}'(t_n)) && \text{// as } \forall s, s' \in [t_i]_{\mathcal{A}} : \mathcal{A}'(s) = \mathcal{A}'(s') \\
&\quad = \mathcal{A}'(t) && \text{// where } t = f(t_1, \ldots, t_n) \in T_\Omega \\
&\quad = \mathcal{A}'([t]_{\mathcal{A}}) && \text{// as } \forall s, s' \in [t]_{\mathcal{A}} : \mathcal{A}'(s) = \mathcal{A}'(s') \\
&\quad = h(e) && \text{// by def. of } h \text{, where } e = t_{\mathcal{A}} \\
&\quad = h(t_{\mathcal{A}}) \\
&\quad = h\big(f_{\mathcal{A}}((t_1)_{\mathcal{A}}, \ldots, (t_n)_{\mathcal{A}})\big) \\
&\quad = h\big(f_{\mathcal{A}}(e_1, \ldots, e_n)\big)
\end{aligned}$$

Thus, $h$ is a homomorphism from $\mathcal{A}$ to $\mathcal{A}'$, and $\mathcal{A}$ is homomorphic to $\mathcal{A}'$.

The only point left to show is injectivity of $h$. Let $e_1$ and $e_2$ be two distinct elements of the universe of $\mathcal{A}$: $e_1, e_2 \in U_{\mathcal{A}}$ and $e_1 \neq e_2$. As $\mathcal{A}$ is term-generated, there exist terms $t_1, t_2 \in T_\Omega$ such that $\mathcal{A}(t_1) = e_1$, $\mathcal{A}(t_2) = e_2$. Put $t_i' = t_i{\downarrow}_{R^{\approx}_{\mathcal{A}}}$ be the normal form of each $t_i$ with respect to the rewrite system[1] $R^{\approx}_{\mathcal{A}}$, for every $i \in \{1,2\}$. As[2] $\mathcal{A} \models \mathsf{E}_{\mathcal{A}} \models t_i \approx t_i'$, for every $i \in \{1,2\}$, we have $t_1' \neq t_2'$, as for otherwise $e_1 = \mathcal{A}(t_1) = \mathcal{A}(t_1') = \mathcal{A}(t_2') = \mathcal{A}(t_2) = e_2$ contradicting $e_1 \neq e_2$. By Definition 3.49 of $\mathsf{D}_{\mathcal{A}}$, we know $(t_1' \approx t_2' \rightarrow) \in \mathsf{D}_{\mathcal{A}}$. Consequently,

$$\begin{aligned}
& & \mathcal{A}'(t_1') &\neq \mathcal{A}'(t_2') && \text{// as } \mathcal{A}' \models \mathsf{D}_{\mathcal{A}} \\
&\Rightarrow & \mathcal{A}'(t_1) &\neq \mathcal{A}'(t_2) && \text{// as } \mathcal{A}' \models \mathsf{E}_{\mathcal{A}} \text{, and } \mathsf{E}_{\mathcal{A}} \models t_i \approx t_i' \text{, for each } i \in \{1,2\} \\
&\Rightarrow & h(e_1) &\neq h(e_2) && \text{// by Prop. 3.75}
\end{aligned}$$

Thus, $h$ is an injective homomorphism from $\mathcal{A}$ to $\mathcal{A}'$, hence a monomorphism from $\mathcal{A}$ to $\mathcal{A}'$. By Definition 3.73 we conclude $\mathcal{A}'$ is a weak algebra. ■

COROLLARY 3.77 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*Let* $\Sigma'$*-algebra* $\mathcal{A}'$ *be a weak algebra. Two base algebras* $\mathcal{A}_1, \mathcal{A}_2 \in \mathscr{C}$ *are monomorphic to* $\mathcal{A}'$ *if and only if* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are isomorphic.*

## 3.4.7 Sufficient Completeness Criterion

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

In this section we discuss the main component contributing to refutational completeness of the SUP(T) calculus – the *sufficient completeness criterion.* The

[1] Please, recall Definition 3.46 of the rewrite system $R^{\approx}_{\mathcal{A}}$, page 61.

[2] Please, recall Definition 3.49 of the clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, page 62.

criterion imposes on an input set $N$ of general $\Sigma'$-clauses a condition, according to which every ground smooth extension term must have the same interpretation as some ground base term in all weak models of the set $sgi(N)$ of all simple ground instances of $N$. The criterion helps to gain two goals, discussed in the overview to the section "Completeness of SUP(T)": on one hand, it assures every weak model of all simple ground instances $sgi(N)$ of the set $N$ to be a model of all its ground instances $gi(N)$, hence a model of $N$, and, on the other hand, it provides a sufficient condition for a model of the set[1] $N_{\mathcal{A}} = sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ to be *hierarchic*, hence a *hierarchic model* of $N$.

First, we give a definition of a sufficiently complete clause set, Definition 3.78. Then, in the *Hierarchic Model Lemma* we assert that a given clause set $N$ is theory-consistent if and only if the ground clause set $N_{\mathcal{A}}$ has a $\Sigma'$-model, where $\mathcal{A}$ is a base algebra from $\mathscr{C}$.

◄ DEFINITION 3.78 Sufficiently Complete Clause Set

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively.

A set $N$ of $\Sigma'$-clauses is called ***sufficiently complete with respect to simple instances***, if for every weak model $\mathcal{A}'$ of $sgi(N)$ and every ground smooth extension term[1] $t'$ there exists a ground base term $t$ such that $\mathcal{A}' \models t' \approx t$. We write $\mathcal{SC}_{\mathsf{HSp}}$ to denote the class of all such sets of clauses:

$$\mathcal{SC}_{\mathsf{HSp}} \overset{\text{def}}{=} \{N \in Cl_{\Sigma'} \mid (\forall \mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}})(\forall t' \in T^E_{\Omega'})(\exists t \in T_\Omega) : \\ t' \text{ smooth}, \mathcal{A}' \models sgi(N) \Rightarrow \mathcal{A}' \models t' \approx t\}.$$

■

For the sake of conciseness, we call clause sets in $\mathcal{SC}_{\mathsf{HSp}}$ just *sufficiently complete*. Expanding the notion of a weak model, the definition of the class of all sufficiently complete clause sets can be reformulated as follows:

$$\mathcal{SC}_{\mathsf{HSp}} \overset{\text{def}}{=} \{N \in Cl_{\Sigma'} \mid (\forall \mathcal{A} \in \mathscr{C})(\forall t' \in T^E_{\Omega'})(\exists t \in T_\Omega) : \\ t' \text{ smooth} \Rightarrow sgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \models t' \approx t\}.$$

◄ LEMMA 3.79 Hierarchic Model Lemma

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*Let* $N \in \mathcal{SC}_{\mathsf{HSp}}$ *be a sufficiently complete clause set, and* $\mathcal{A}' \in \mathscr{A}_{\Sigma'}$ *an arbitrary* $\Sigma'$*-algebra. The algebra* $\mathcal{A}'$ *is a hierarchic model of* $N$ *if and only if it satisfies the clause set* $N_{\mathcal{A}}$*, for some base algebra*[2] $\mathcal{A} \in \mathscr{C}$*:*

$$\forall N \in \mathcal{SC}_{\mathsf{HSp}}. \forall \mathcal{A}' \in \mathscr{A}_{\Sigma'} : (\mathcal{A}' \models_{\mathscr{C}} N \Leftrightarrow \exists \mathcal{A} \in \mathscr{C} : \mathcal{A}' \models N_{\mathcal{A}})$$

◄ PROOF

**The "⇒" direction**. Assume $\mathcal{A}' \models N$, for some $\Sigma'$-algebra $\mathcal{A}' \in \mathscr{A}_{\Sigma'}$. As $\mathcal{A}'$ is a hier-

[1]Please, recall Definitions 3.16 and 3.17 of different kinds of terms (base, non-base, etc.), page 39; Definition 3.23 of a hierarchic algebra/model, page 44; Definition 3.25 of the entailment relation $\models_{\mathscr{C}}$, page 45; Definition 3.46 of the rewrite system $R^{\approx}_{\mathcal{A}}$, page 61; Definition 3.49 of clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, page 62; Definition 3.55 of sets of all simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances $sgi_{\mathcal{A}}$, page 67; Definition 3.56 of a clause set $N_{\mathcal{A}}$, page 67; Definition 3.73 of a weak algebra, page 89.

[2]Please, note that the base algebra $\mathcal{A}$ is only used for construction of the clause set $N_{\mathcal{A}} = sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ and is not a subject to any logical entailment condition, in contrast to $\mathcal{A}'$ which has to entail $N_{\mathcal{A}}$.

archic model of $N$, we know $\mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}}$, consequently $\mathcal{A}'|_\Sigma = \mathcal{A}$ for some base algebra $\mathcal{A} \in \mathscr{C}$. Therefore, $\mathcal{A}' \models \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$. From $\mathcal{A}' \models N$ it follows that $\mathcal{A}' \models sgi_\mathcal{A}(N)$. Summing up, $\mathcal{A}' \models \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A} \cup sgi_\mathcal{A}(N) = N_\mathcal{A}$.

**The "$\Leftarrow$" direction**. Let $\mathcal{A} \in \mathscr{C}$ be an arbitrary base algebra. Assume $N_\mathcal{A}$ is satisfiable, then there exists a term-generated $\Sigma'$-algebra $\mathcal{A}'$ that satisfies $N_\mathcal{A}$. According to Lemma 3.61 (ii), $N_\mathcal{A} \models sgi(N)$, hence $\mathcal{A}' \models sgi(N)$. Moreover, as $\mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A} \subseteq N_\mathcal{A}$, we know $\mathcal{A}' \models \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$, yielding by Theorem 3.76 $\mathcal{A}'$ is a weak algebra. Therefore, from the fact that $N$ is sufficiently complete, and the Extension Terms Lemma (Lemma 3.19), it follows that for every ground non-base term $t'$ of a base sort there exists a ground base term $t$ such that $\mathcal{A}' \models t' \approx t$. It is easy to see that from this and the fact that $\mathcal{A}' \models sgi(N)$, it follows that $\mathcal{A}' \models gi(N)$, consequently $\mathcal{A}' \models N$.

Our next claim is that the restriction $\mathcal{A}'|_\Sigma$ of $\mathcal{A}'$ to the base signature $\Sigma$ is a base algebra from the class $\mathscr{C}$. To prove $\mathcal{A}'|_\Sigma \in \mathscr{C}$, we show that $\mathcal{A}'|_\Sigma$ is isomorphic to the base algebra $\mathcal{A}$ by providing a mapping $h$ from the universe $U_\mathcal{A}$ of $\mathcal{A}$ to the universe $U_{\mathcal{A}'|_\Sigma}$ of $\mathcal{A}'|_\Sigma$ and proving $h$ to be a bijective homomorphism, i.e. that $h$ and $h^{-1}$ are homomorphisms from $\mathcal{A}$ to $\mathcal{A}'|_\Sigma$ and from $\mathcal{A}'|_\Sigma$ to $\mathcal{A}$, respectively. Let $h$ be defined analogous to that in the proof of Theorem 3.76, that is, for every base sort $\mathsf{S} \in \mathcal{S}$, and every element $e \in \mathsf{S}_\mathcal{A} \subseteq U_\mathcal{A}$,

$$h(e) \overset{\text{def}}{=} \mathcal{A}'|_\Sigma([t]_\mathcal{A}),$$

where[1] $t \in T_\Omega$ is a ground base term such that $\mathcal{A}(t) = e$. That is, for every element $e$ in $U_\mathcal{A}$, the mapping $h$ takes an arbitrary ground base term, whose interpretation under $\mathcal{A}$ equals $e$, and returns its interpretation under $\mathcal{A}'|_\Sigma$. As all base algebras in $\mathscr{C}$ are term-generated, the mapping $h$ is total. Since the algebra $\mathcal{A}'|_\Sigma$ is the $\Sigma$-restriction of $\mathcal{A}'$, the two algebras agree on base terms $T_\Omega$ and have the same interpretation of base sorts $\mathcal{S}$, i.e. :

$$\begin{aligned}\forall\, s \in T_\Omega &: \mathcal{A}'|_\Sigma(s) = \mathcal{A}'(s),\\ \forall\, \mathsf{S} \in \mathcal{S} &: \mathcal{A}'|_\Sigma(\mathsf{S}) = \mathcal{A}'(\mathsf{S}).\end{aligned}$$

In the proof of Theorem 3.76 we have shown that all terms in the equivalence class[2] $[t]_\mathcal{A}$ have the same interpretation under $\mathcal{A}'$. As $[t]_\mathcal{A}$ contains only base terms, and the algebras $\mathcal{A}'|_\Sigma$ and $\mathcal{A}'$ agree on every ground base term, we conclude that $\mathcal{A}'|_\Sigma$ interprets all terms in $[t]_\mathcal{A}$ equally as well, hence $h$ is well-defined:

$$\forall\, s, s' \in [t]_\mathcal{A} : \mathcal{A}'|_\Sigma(s) = \mathcal{A}'|_\Sigma(s'),$$

Thus, for all $\mathsf{S} \in \mathcal{S}$, $e \in \mathsf{S}_A$, and $t \in T_\Omega$, such that $\mathcal{A}(t) = e$, we have:

$$\begin{aligned}h(e) &= \mathcal{A}'|_\Sigma([t]_\mathcal{A})\\ &= \mathcal{A}'([t]_\mathcal{A})\end{aligned}$$

Now, we show $h$ is structure-preserving. Let $f \in \Omega$ be an arbitrary base operator symbol of arity $n \geq 0$, such that $f : \mathsf{S}_1 \times \ldots \times \mathsf{S}_n \to \mathsf{S}$, and $e_1, \ldots, e_n$ arbitrary elements of the universe of $\mathcal{A}$, such that $e_i \in (\mathsf{S}_i)_\mathcal{A}$ for every $i \in \{1, \ldots, n\}$, then

$$\begin{aligned}&f_{\mathcal{A}'|_\Sigma}\big(h(e_1), \ldots, h(e_n)\big)\\ &\quad = f_{\mathcal{A}'|_\Sigma}\big(\mathcal{A}'|_\Sigma([t_1]_\mathcal{A}), \ldots, \mathcal{A}'|_\Sigma([t_n]_\mathcal{A})\big) \qquad \text{// by def. of } h\text{, for some } t_1, \ldots, t_n \in T_\Omega\end{aligned}$$

[1]Please, note that $[t]_\mathcal{A}$ denotes the equivalence class of $t$ under $\mathcal{A}$ or an arbitrary representative of the class, whereas $t_A$ denotes the interpretation of $t$ under $\mathcal{A}$; thus, the notions $[t]_\mathcal{A}$ and $t_A$ should not be confused!

[2]Please, recall Definition 3.45 of a equivalence class $[t]_\mathcal{A}$, page 60.

$$\begin{aligned}
& && \text{s.t. } \mathcal{A}(t_i) = e_i \text{ for all } i \in \{1,\dots,n\} \\
&= f_{\mathcal{A}'|_\Sigma}\big(\mathcal{A}'|_\Sigma(t_1),\dots,\mathcal{A}'|_\Sigma(t_n)\big) && \text{// as } \forall\, s,s' \in [t_i]_\mathcal{A} : \mathcal{A}'|_\Sigma(s) = \mathcal{A}'|_\Sigma(s') \\
&= \mathcal{A}'|_\Sigma(t) && \text{// where } t = f(t_1,\dots,t_n) \in T_\Omega \\
&= \mathcal{A}'|_\Sigma([t]_\mathcal{A}) && \text{// as } \forall\, s,s' \in [t]_\mathcal{A} : \mathcal{A}'|_\Sigma(s) = \mathcal{A}'|_\Sigma(s') \\
&= h(e) && \text{// by def. of } h, \text{ where } e = t_\mathcal{A} \\
&= h(t_\mathcal{A}) \\
&= h\big(f_\mathcal{A}((t_1)_\mathcal{A},\dots,(t_n)_\mathcal{A})\big) \\
&= h\big(f_\mathcal{A}(e_1,\dots,e_n)\big)
\end{aligned}$$

Thus, $h$ is a homomorphism from $\mathcal{A}$ to $\mathcal{A}'|_\Sigma$, and $\mathcal{A}$ is homomorphic to $\mathcal{A}'|_\Sigma$.

The task now falls into two subgoals, where we have to show that

1. $h^{-1}$ exists, for which we have to ensure that
   - (i) $h$ is surjective, that is that every element of the $h$'s co-domain is mapped to by at least one element of the $h$'s domain, or more formally: for every base sort $\mathsf{S} \in \mathcal{S}$ and every element $e' \in \mathsf{S}_{\mathcal{A}'|_\Sigma}$ there exists an $e \in \mathsf{S}_\mathcal{A}$ such that $h(e) = e'$;
   - (ii) $h$ is injective, meaning that every element of the $h$'s co-domain is mapped to by at most one element of the $h$'s domain, or more formally: for any $e_1, e_2 \in U_\mathcal{A}$ if $h(e_1) = h(e_2)$ then $e_1 = e_2$, or equivalently, if $e_1 \neq e_2$ then $h(e_1) \neq h(e_2)$;
2. $h^{-1}$ is a homomorphism from $\mathcal{A}'|_\Sigma$ to $\mathcal{A}$.

For the surjection part, assume $e' \in \mathsf{S}_{\mathcal{A}'|_\Sigma}$, where $\mathsf{S} \in \mathcal{S}$ is an arbitrary base sort. As $\mathsf{S}_{\mathcal{A}'|_\Sigma} = \mathsf{S}_{\mathcal{A}'}$ and $\mathcal{A}'$ is term-generated, there exists a ground term $s \in T_{\Omega'}(\mathsf{S})$ such that $\mathcal{A}'(s) = e'$. We distinguish two cases:

- $s$ is a ground *base* term, i.e. $s \in T_\Omega(\mathsf{S}) \subseteq T_{\Omega'}(\mathsf{S})$. Thus,

$$\begin{aligned}
e' &= \mathcal{A}'(s) \\
&= \mathcal{A}'|_\Sigma(s) && \text{// as } \forall\, s \in T_\Omega : \mathcal{A}'(s) = \mathcal{A}'|_\Sigma(s) \\
&= \mathcal{A}'|_\Sigma([s]_\mathcal{A}) && \text{// as } \forall\, t,t' \in [s]_\mathcal{A} : \mathcal{A}'|_\Sigma(t) = \mathcal{A}'|_\Sigma(t') \\
&= h(e) && \text{// by def. of } h, \text{ where } e = \mathcal{A}(s)
\end{aligned}$$

- $s$ is a ground *non-base* term, i.e. $s \in T_{\Omega'}(\mathsf{S}) \setminus T_\Omega$. Since $\mathcal{A}'$ is a weak algebra and $N$ sufficiently complete with respect to simple instances, we know from Definition 3.78 and the Extension Terms Lemma (Lemma 3.19) that there exists a ground base term $t \in T_\Omega(\mathsf{S})$ such that $\mathcal{A}' \models s \approx t$, therefore $e' = \mathcal{A}'(t)$, which reduces the case to the previous one.

For the injection part, let $e_1$ and $e_2$ be two distinct elements of the universe of $\mathcal{A}$: $e_1, e_2 \in U_\mathcal{A}$ and $e_1 \neq e_2$. As $\mathcal{A}$ is term-generated, there exist terms $t_1, t_2 \in T_\Omega$ such that $\mathcal{A}(t_1) = e_1$, $\mathcal{A}(t_2) = e_2$. Put $t'_i = t_i{\downarrow}_{R^\approx_\mathcal{A}}$ be the normal form of each $t_i$ with respect to the rewrite system[1] $R^\approx_\mathcal{A}$, for every $i \in \{1,2\}$. As $\mathcal{A} \models \mathsf{E}_\mathcal{A} \models t_i \approx t'_i$, for every $i \in \{1,2\}$, we have $t'_1 \neq t'_2$, as for otherwise $e_1 = \mathcal{A}(t_1) = \mathcal{A}(t'_1) = \mathcal{A}(t'_2) = \mathcal{A}(t_2) = e_2$ contradicting $e_1 \neq e_2$. By Definition 3.49 of $\mathsf{D}_\mathcal{A}$, we know $(t'_1 \approx t'_2 \rightarrow\ ) \in \mathsf{D}_\mathcal{A}$. Consequently,

$$\mathcal{A}'(t'_1) \neq \mathcal{A}'(t'_2) \qquad \text{// as } \mathcal{A}' \models N_\mathcal{A} \text{ and } \mathsf{D}_\mathcal{A} \subseteq N_\mathcal{A}$$

[1]Please, recall Definition 3.46 of the rewrite system $R^\approx_\mathcal{A}$, page 61.

$$\begin{array}{lcl}
\Rightarrow & \mathcal{A}'(t_1) \neq \mathcal{A}'(t_2) & \text{// as } \mathcal{A}' \models N_\mathcal{A},\ \mathsf{E}_\mathcal{A} \subseteq N_\mathcal{A}, \text{ and } \mathsf{E}_\mathcal{A} \models t_i \approx t_i' \\
\Rightarrow & \mathcal{A}'|_\Sigma(t_1) \neq \mathcal{A}'|_\Sigma(t_2) & \text{// as } \forall\, t \in T_\Omega : \mathcal{A}'(t) = \mathcal{A}'|_\Sigma(t) \\
\Rightarrow & \mathcal{A}'|_\Sigma([t_1]_\mathcal{A}) \neq \mathcal{A}'|_\Sigma([t_2]_\mathcal{A}) & \text{// as } \forall\, t \in T_\Omega. \forall\, s, s' \in [t]_\mathcal{A} : \mathcal{A}'|_\Sigma(s) = \mathcal{A}'|_\Sigma(s') \\
\Rightarrow & h(e_1) \neq h(e_2) & \text{// by def. of } h, \text{ where } e_i = \mathcal{A}(t_i), \\
 & & \quad \text{for each } i \in \{1,2\}
\end{array}$$

Thus, $h$ is injective and surjective, hence bijective, consequently the inverse mapping $h^{-1}$ exists.

The final task is to show that $h^{-1}$ is a homomorphism from $\mathcal{A}'|_\Sigma$ to $\mathcal{A}$. Since $h$ is a bijective mapping from $U_\mathcal{A}$ to $\mathcal{A}'|_\Sigma$ (and, hence, $h^{-1}$ is total and well-defined) and a homomorphism from $\mathcal{A}$ to $\mathcal{A}'|_\Sigma$ (and, hence, $h(\mathsf{S}_\mathcal{A}) \subseteq \mathsf{S}_{\mathcal{A}'|_\Sigma}$ for every base sort $\mathsf{S} \in \mathcal{S}$), it follows that $h^{-1}(\mathsf{S}_{\mathcal{A}'|_\Sigma}) \subseteq \mathsf{S}_\mathcal{A}$ for every base sort $\mathsf{S} \in \mathcal{S}$. So, the only thing left to show is that $h^{-1}$ is structure-preserving, for which we have to ensure that

$$h^{-1}(f_{\mathcal{A}'|_\Sigma}(e_1', \ldots, e_n')) = f_\mathcal{A}(h^{-1}(e_1'), \ldots, h^{-1}(e_n'))$$

holds for every base operator symbol $(f : \mathsf{S}_1 \times \ldots \times \mathsf{S}_n \to \mathsf{S}) \in \Omega$, and all elements of the $\mathcal{A}'|_\Sigma$'s universe $e_1', \ldots, e_n'$, such that $e_i' \in (\mathsf{S}_i)_{\mathcal{A}'|_\Sigma}$ for every $i \in \{1, \ldots, n\}$. Assume $f$ and all $e_i'$ are arbitrary operator symbol and elements of the $\mathcal{A}'|_\Sigma$'s universe, respectively, satisfying the conditions right above, then:

$$\begin{array}{ll}
f_\mathcal{A}\big(h^{-1}(e_1'), \ldots, h^{-1}(e_n')\big) & \\
\quad = f_\mathcal{A}\big(h^{-1}(h(e_1)), \ldots, h^{-1}(h(e_n))\big) & \text{// by surjectivity of } h, \\
 & \quad \text{where all } e_i \in U_\mathcal{A} \text{ s.t. } h(e_i) = e_i' \\
 & \quad \text{and } e_i \in (\mathsf{S}_i)_\mathcal{A}, \text{ for all } i \in \{1, \ldots, n\} \\
\quad = f_\mathcal{A}(e_1, \ldots, e_n) & \text{// as } h^{-1} \text{ is the inverse of } h \\
\quad = f_\mathcal{A}\big(\mathcal{A}(t_1), \ldots, \mathcal{A}(t_n)\big) & \text{// as } \mathcal{A} \text{ term-generated,} \\
 & \quad \text{where all } t_i \in T_\Omega \text{ s.t. } \mathcal{A}(t_i) = e_i \\
\quad = \mathcal{A}(f(t_1, \ldots, t_n)) & \\
\quad = \mathcal{A}(t) & \text{// where } t = f(t_1, \ldots, t_n) \\
\quad = h^{-1}\big(h(\mathcal{A}(t))\big) & \\
\quad = h^{-1}\big(\mathcal{A}'|_\Sigma([t]_\mathcal{A})\big) & \text{// by def. of } h \\
\quad = h^{-1}\big(\mathcal{A}'|_\Sigma(t)\big) & \text{// as } \forall\, s, s' \in [t]_\mathcal{A} : \mathcal{A}'|_\Sigma(s) = \mathcal{A}'|_\Sigma(s') \\
\quad = h^{-1}\big(\mathcal{A}'|_\Sigma(f(t_1, \ldots, t_n))\big) & \\
\quad = h^{-1}\Big(f_{\mathcal{A}'|_\Sigma}\big(\mathcal{A}'|_\Sigma(t_1), \ldots, \mathcal{A}'|_\Sigma(t_n)\big)\Big) & \\
\quad = h^{-1}\Big(f_{\mathcal{A}'|_\Sigma}\big(\mathcal{A}'|_\Sigma([t_1]_\mathcal{A}), \ldots, \mathcal{A}'|_\Sigma([t_n]_\mathcal{A})\big)\Big) & \text{// as } \forall\, s, s' \in [t_i]_\mathcal{A} : \mathcal{A}'|_\Sigma(s) = \mathcal{A}'|_\Sigma(s') \\
\quad = h^{-1}\Big(f_{\mathcal{A}'|_\Sigma}\big(h(e_1), \ldots, h(e_n)\big)\Big) & \text{// by def. of } h, \text{ where } e_1, \ldots, e_n \text{ as} \\
 & \quad \text{defined above} \\
\quad = h^{-1}\big(f_{\mathcal{A}'|_\Sigma}(e_1', \ldots, e_n')\big) & \text{// as } h(e_i) = e_i' \text{ for each } i \in \{1, \ldots, n\}
\end{array}$$

Thus, $h^{-1}$ is a homomorphism from $\mathcal{A}'|_\Sigma$ to $\mathcal{A}$; and we are done. ■

**Comparison to sufficient completeness criteria of [BGW94] and [BW13]**

The most recent work by Baumgartner and Waldmann [BW13] offers another sufficient completeness criterion very much similar to the one proposed by us, namely:

> A set $N$ of $\Sigma'$-clauses is sufficiently complete with respect to simple instances iff for every $\Sigma'$-model $\mathcal{A}'$ of *sgi*$(N) \cup \mathsf{Gnd}(\mathscr{C})$ and every ground non-base term $t'$ of a base sort there is a ground base term $t$ such that $\mathcal{A}' \models t' \approx t$, where $\mathsf{Gnd}(\mathscr{C})$ is the set of all ground base formulae that are satisfied by every base algebra $\mathcal{A} \in \mathscr{C}$.
>
> (Baumgartner, Waldmann [BW13])

On one hand, from the Extension Terms Lemma (Lemma 3.19) it follows that every ground non-base term $t'$ of a base sort is equal under a $\Sigma'$-algebra to some ground base term $t$, if and only if each ground smooth extension term $s'$ is equal under the same algebra to some ground base term $s$. Thus, it does not make any difference if a completeness criterion enjoins all non-base terms of a base sort or only smooth extension terms to be sufficiently defined. On the other hand, from Lemma 3.51 it follows that the set $\mathsf{Gnd}(\mathscr{C})$ is entailed by $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, for every base algebra $\mathcal{A} \in \mathscr{C}$, but $\mathsf{Gnd}(\mathscr{C})$ does not entail $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, in general. Consequently, the sufficient completeness criterion proposed in Definition 3.78 is stronger than (or, in general, is at least as strong as) the one from [BW13].

The original sufficient completeness criterion of Bachmair, Ganzinger, and Waldmann [BGW94] imposes even a stronger condition:

> A set $N$ of $\Sigma'$-clauses is sufficiently complete with respect to simple instances iff for every $\Sigma'$-model $\mathcal{A}'$ of *sgi*$(N)$ and every ground non-base term $t'$ of a base sort there is a ground base term $t$ such that $\mathcal{A}' \models t' \approx t$. (Bachmair, Ganzinger, Waldmann [BGW94])

Evidently, the set of models of either *sgi*$(N) \cup \mathsf{Gnd}(\mathscr{C})$ (as in [BW13]), or *sgi*$(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ (as suggested by us) is in general smaller than the set of models of *sgi*$(N)$. Therefore, the sufficient completeness criterion originally proposed in [BGW94] is weaker than the one of [BW13] or the one suggested by us in Definition 3.78.

### 3.4.8 Refutational Completeness

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

In Section 3.1.4 we have introduced the concept of calculi approximation, Definition 3.13, aimed to serve as an abstract mechanism for proving refutational completeness of a calculus $(\mathcal{I}_1, \mathcal{R}_1)$, the approximating one, by relating it to another calculus $(\mathcal{I}_2, \mathcal{R}_2)$, the approximated one, that is a-priori known to be refutationally complete. Each calculus $(\mathcal{I}_i, \mathcal{R}_i)$, $i \in \{1,2\}$, comes with implicitly given:

- a class $\mathcal{N}_i$ of formula sets, saturating which the calculus is aimed at, and
- an underlying entailment relation $\models_i$, with respect to which a redundancy criterion $\mathcal{R}_i$ is defined.

The approximation is carried out via so called approximation function $\alpha$ – a mapping from $\mathcal{N}_1$ to $\mathcal{N}_2$. According to the definition of calculi approximation, the calculus $(\mathcal{I}_1, \mathcal{R}_1)$ approximates $(\mathcal{I}_2, \mathcal{R}_2)$ if each of the following conditions is satisfied for every formula set $N \in \mathcal{N}_1$:

(i) if $N$ is saturated with respect to $(\mathcal{I}_1, \mathcal{R}_1)$, then $\alpha(N)$ is saturated with respect to $(\mathcal{I}_2, \mathcal{R}_2)$,

(ii) if $N \models_1 \bot_1$, then $\alpha(N) \models_2 \bot_2$, and

(iii) if $\bot_2 \in \alpha(N)$ and $N$ is saturated, then $\bot_1 \in N$,

where $\bot_i$ is a contradictory formula with respect to the entailment relation $\models_i$, for every $i \in \{1,2\}$. In Theorem 3.14 we have shown that if the approximated calculus $(\mathcal{I}_2, \mathcal{R}_2)$ is complete for $\mathcal{N}_2$, then the approximating calculus $(\mathcal{I}_1, \mathcal{R}_1)$ is complete for $\mathcal{N}_1$.

We want to lift the completeness result of the standard superposition[1] $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$ for ground clauses (with respect to the general entailment relation $\models$) to the case of the hierarchic superposition $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$ for sufficiently complete clauses (with respect to the entailment relation $\models_{\mathscr{C}}$), using the calculi approximation mechanism. To this end, we aim at approximating $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$ by $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$. As the flat superposition is refutationally complete, the approximation of $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$ by $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$ yields by Theorem 3.14 that the hierarchic superposition is refutationally complete as well. For this we need to find an approximation function confirming with the conditions listed above. Let this be the function $\alpha_{\mathcal{H}\text{-}\mathcal{F}}$ defined as follows.

DEFINITION 3.80 ► $\mathcal{H}$-$\mathcal{F}$ Approximation Function $\alpha_{\mathcal{H}\text{-}\mathcal{F}}$

The function $\alpha_{\mathcal{H}\text{-}\mathcal{F}}$ defined as[1]

$$\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N) \overset{\text{def}}{=} \begin{cases} N_{\mathcal{A}}, & \text{if (i) } N \text{ an } (\mathcal{H}, \mathcal{R}^{\mathcal{H}})\text{-saturated set} \\ & \quad\text{of abstracted } \Sigma'\text{-clauses, and} \\ & \quad\text{(ii) } \exists \mathcal{A} \in \mathscr{C} : \mathcal{A} \models N \cap Cl_{\Sigma} \\ \{\Box\}, & \text{oth.} \end{cases}$$

is[2] called the $\mathcal{H}$-$\mathcal{F}$ ***approximation function***. ■

Informally speaking, if the given set $N$ is saturated and the subset $N \cap Cl_{\Sigma}$ of all base clauses within $N$ is theory-consistent, then the function returns a set $N_{\mathcal{A}}$, where $\mathcal{A}$ is any base algebra that satisfies the base subset of $N$; otherwise, the function returns a set $\{\Box\}$ consisting of a single empty clause. In the following theorem we prove that SUP(T) approximates the ground superposition via the function $\alpha_{\mathcal{H}\text{-}\mathcal{F}}$.

THEOREM 3.81 ► $\mathcal{H}$-$\mathcal{F}$ Approximation

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$. *Assume* $\mathsf{Sp}$ *is compact.*

*The hierarchic superposition calculus* $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$ *approximates the flat ground superposition calculus* $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$ *via the approximation function* $\alpha_{\mathcal{H}\text{-}\mathcal{F}}$.

PROOF ► Let $\mathcal{SC}_{\mathsf{HSp}}$ be the class of sufficiently complete clause sets. We have to prove that

[1] Please, recall Definition 3.20 of an abstracted clause, page 41; Definition 3.25 of the entailment relation $\models_{\mathscr{C}}$, page 45; Definition 3.32 of the hierarchic superposition inference system $\mathcal{H}$, page 49; Definition 3.39 of the standard superposition inference system $\mathcal{F}$, page 56; Definitions 3.64 and 3.40 of the hierarchic $\mathcal{R}^{\mathcal{H}} = (\mathcal{R}_F^{\mathcal{H}}, \mathcal{R}_I^{\mathcal{H}})$ and the standard ground $\mathcal{R}^{\mathcal{F}} = (\mathcal{R}_F^{\mathcal{F}}, \mathcal{R}_I^{\mathcal{F}})$ redundancy criterions, pages 74 and 56, respectively; Definition 3.56 of a clause set $N_{\mathcal{A}}$, page 67.

[2] Recall that $Cl_{\Sigma'}$ denotes the class of all clauses over the body signature $\Sigma'$, and $Cl_{\Sigma}$ the class of all clauses over the base signature $\Sigma$. Thus, $Cl_{\Sigma}$ is the class of all base clauses, and $Cl_{\Sigma'} \setminus Cl_{\Sigma}$ stands for the class of all non-base clauses.

the following three conditions of Definition 3.13 hold for the current context: (i) if $N \in \mathcal{SC}_{\mathsf{HSp}}$ is saturated with respect to $(\mathcal{H},\mathcal{R}^{\mathcal{H}})$, then $\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N)$ is saturated with respect to $(\mathcal{F},\mathcal{R}^{\mathcal{F}})$; (ii) if $N \models_{\mathscr{C}} \bot$, then $\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N) \models \bot$; (iii) if $\square \in \alpha_{\mathcal{H}\text{-}\mathcal{F}}(N)$ and $N$ is saturated then $\square \in N$.

Assume $N$ is saturated with respect to $(\mathcal{H},\mathcal{R}^{\mathcal{H}})$. If there exists some base algebra $\mathcal{A} \in \mathscr{C}$ that satisfies the set $N \cap Cl_{\Sigma}$ of all base clauses within $N$, then, according to Definition 3.80, $\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N) = N_{\mathcal{A}}$. Saturation of $\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N)$ follows in this case by Theorem 3.72. If there is no base algebra $\mathcal{A} \in \mathscr{C}$ satisfying $sgi(N) \cap Cl_{\Sigma}$, then $\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N) = \{\square\}$, which is trivially saturated. This proves condition (i).

Next we show condition (iii), whereupon continue with (ii). Assume $N$ is saturated with respect to the hierarchic calculus $(\mathcal{H},\mathcal{R}^{\mathcal{H}})$ and $\square \in \alpha_{\mathcal{H}\text{-}\mathcal{F}}(N)$. We have to consider two cases[1]:

- there exists a base algebra $\mathcal{A} \in \mathscr{C}$ that satisfies $N \cap Cl_{\Sigma}$. Then $\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N) = N_{\mathcal{A}}$, and $\square \in N_{\mathcal{A}}$. Obviously, $\square \notin \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, hence $\square \in N_{\mathcal{A}} \setminus (\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) = sgi_{\mathcal{A}}(N)$, consequently $\square \in N$;
- there is no algebra $\mathcal{A} \in \mathscr{C}$ satisfying $N \cap Cl_{\Sigma}$, i.e. $N \cap Cl_{\Sigma}$ is theory-inconsistent. As the base specification $\mathsf{Sp}$ is compact, there is a finite theory-inconsistent subset $M$ of $N \cap Cl_{\Sigma}$. A Constraint Refutation inference with premises $M$ derives $\square$. Since $N$ is saturated by assumption, the inference must be redundant with respect to $\mathcal{R}^{\mathcal{H}}$, which holds, according to Definition 3.64 of $\mathcal{R}^{\mathcal{H}}$, if and only if an empty clause $\square$ is already in $N$.

We prove (ii) by contraposition, wherefore we have to show that if $\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N)$ is satisfiable then $N$ is consistent relative to[2] $\mathscr{C}$, or equivalently, $N$ has a hierarchic model. Suppose $\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N)$ is satisfiable. Obviously, $\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N)$ is different from $\{\square\}$, hence, according to the definition of the approximation function $\alpha_{\mathcal{H}\text{-}\mathcal{F}}$, the set $\alpha_{\mathcal{H}\text{-}\mathcal{F}}(N)$ equals $N_{\mathcal{A}}$ for some base algebra $\mathcal{A} \in \mathscr{C}$. As $N$ is sufficiently complete and $N_{\mathcal{A}}$ satisfiable, we conclude by the Hierarchic Model Lemma (Lemma 3.79) that $N$ has a hierarchic model. This completes the proof. ■

◄ THEOREM 3.82
Hierarchic Completeness Theorem

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$. *Assume* $\mathsf{Sp}$ *is compact.*

*The hierarchic superposition calculus* $(\mathcal{H},\mathcal{R}^{\mathcal{H}})$ *is refutationally complete for all sets of clauses that are sufficiently complete.*

◄ PROOF

For the proof we set:

- $(\mathcal{I}_1,\mathcal{R}_1)$ equals $(\mathcal{H},\mathcal{R}^{\mathcal{H}})$, and $\models_1$ and $\mathcal{N}_1$ equal $\models_{\mathscr{C}}$ and $\mathcal{SC}_{\mathsf{HSp}}$, respectively, where $\mathcal{SC}_{\mathsf{HSp}}$ is the class of all clause sets that are sufficiently complete;
- $(\mathcal{I}_2,\mathcal{R}_2)$ equal $(\mathcal{F},\mathcal{R}^{\mathcal{F}})$, and $\models_2$ and $\mathcal{N}_2$ equal $\models$ and $\mathcal{GC}_{\Sigma'}$, respectively, where $\mathcal{GC}_{\Sigma'}$ is the class of all ground $\Sigma'$-clause sets,
- $\alpha = \alpha_{\mathcal{H}\text{-}\mathcal{F}}$, where $\alpha_{\mathcal{H}\text{-}\mathcal{F}}$ is the $\mathcal{H}$-$\mathcal{F}$ approximation function, Definition 3.80.

[1]Please, recall Definition 3.49 of clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, page 62; Definition 3.55 of sets of all simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances $sgi_{\mathcal{A}}$, page 67; Definition 3.56 of a clause set $N_{\mathcal{A}}$, page 67

[2]Please, recall Definition 3.23 of a hierarchic algebra/model, page 44; Definition 3.25 of consistency relative to $\mathscr{C}$, page 45.

According to Theorem 3.81, $(\mathcal{I}_1, \mathcal{R}_1)$ approximates $(\mathcal{I}_2, \mathcal{R}_2)$ via $\alpha$. Moreover, according to Theorem 3.42, $(\mathcal{I}_2, \mathcal{R}_2)$ is refutationally complete for $\mathcal{N}_2$. Consequently, by Theorem 3.14, $(\mathcal{I}_1, \mathcal{R}_1)$ is refutationally complete for all sets in $\mathcal{N}_1$, i.e. $(\mathcal{H}, \mathcal{R}^{\mathcal{H}})$ is refutationally complete for all sufficiently complete clause sets. ■

## 3.5 Local Sufficient Completeness

Thus far we have shown that the hierarchic superposition calculus SUP(T) is refutationally complete for all sets of clauses enjoying the sufficient completeness criterion according to which a clause set has to sufficiently define all ground extension terms. Next we show that for achieving completeness it suffices to sufficiently define only those extension terms that occur in irredundant clauses of a given clause set. We call clause sets, possessing the property that all extension terms occurring in their irredundant clauses are sufficiently defined, *locally sufficiently complete*, Definition 3.84.

**Overview.** Let $M = N_\infty$ be the limit of a fair SUP(T)-derivation $N = N_0 \vdash N_1 \vdash \ldots$, where $N$ is a locally sufficiently complete set of abstracted $\Sigma'$-clauses. Since $N_0 \vdash N_1 \vdash \ldots$ is fair, it follows that $M$ is saturated, by Lemma 3.9. Assume $M$ does not contain an empty clause $\square$. According to Theorem 3.72, the ground clause set $M_{\mathcal{A}} = sgi_{\mathcal{A}}(M) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ is saturated with respect to the standard SUP calculus for ground clauses $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$, where $\mathcal{A} \in \mathscr{C}$ is a base algebra satisfying $sgi(M) \cap Cl_\Sigma$, the base subset of $sgi(M)$. By Lemma 3.41, $M_{\mathcal{A}}$ has a Herbrand model $\mathcal{I}_{M_{\mathcal{A}}} = T_{\Omega'}/R_{M_{\mathcal{A}}}$, where $R_{M_{\mathcal{A}}}$ is a rewrite system constructed from the maximal literals of productive clauses in $M_{\mathcal{A}}$, and $T_{\Omega'}/R_{M_{\mathcal{A}}}$ the quotient for ground $\Sigma'$-terms $T_{\Omega'}$ by the smallest congruence relation containing $R_{M_{\mathcal{A}}}$ (see Definition 2.29 for further details regarding construction of $\mathcal{I}_{M_{\mathcal{A}}}$ and $R_{M_{\mathcal{A}}}$). In general, the Herbrand interpretation $\mathcal{I}_{M_{\mathcal{A}}}$ is not a hierarchic algebra, as it may allow “junks” into base sorts (that is, elements of the base part of the universe, that no base terms are equal to).

We tackle this issue by constructing a new rewrite system $R'$ based on the Herbrand model $\mathcal{I}_{M_{\mathcal{A}}}$. The main purpose of the rewrite system $R'$ is to sufficiently define those terms which are not sufficiently defined by $\mathcal{I}_{M_{\mathcal{A}}}$ by equating such terms to (arbitrary) base terms. Moreover, $R'$ has to be constructed in a safe way such that the corresponding Herbrand interpretation $\mathcal{I}' = T_{\Omega'}/R'$ would be a hierarchic model of the initial clause set $N$. To this end, $R'$ has to preserve the interpretation of (i) those extension terms that occur in $N$, and (ii) non-base terms that contain no extension subterms. Besides, in order to ease a proof of $\mathcal{I}' \models N$, we want the resulting rewrite system $R'$ to be *convergent*.

**Outline.** The current section is split into three parts. In Section 3.5.1, we give a formal definition of a locally sufficient complete clause set, Definition 3.84. As a prerequisite, we first present the notions of *smooth substitutions* and *instances*, Definition 3.83.

In Section 3.5.2, we formally describe the three constituting parts $R^{EF}$, $R^{E}$, $R^{SD}$ of the desired rewrite system $R'$, Definition 3.85, and study the most essential properties of the rewrite system $R' = R^{EF} \cup R^{E} \cup R^{SD}$, namely: (i) sufficient completeness of $R'$, Proposition 3.88; (ii) convergency, Proposition 3.89; (iii) and preservation of the original meaning of all extension terms occurring in an initial locally sufficiently complete clause set $N$, Proposition 3.91.

In Section 3.5.3, we prove the main property of locally sufficiently complete sets of abstracted clauses that any such clause set $N$ has a hierarchic model whenever a fair SUP(T) derivation does not produce an empty clause $\square$ and the set of all base clauses within the limit of the derivation is theory-consistent, Theorem 3.93. And we finish the discussion of local sufficient completeness by establishing refutational completeness of SUP(T) for all locally sufficiently complete set of abstracted clauses provided the base specification Sp is compact, Theorem 3.94.

### 3.5.1 Locally Sufficiently Complete Clause Sets

We begin with the definition of *smooth substitutions* and *smooth instances*. These notions are needed to formally define the class of locally sufficiently complete clause sets.

DEFINITION 3.83 ► Smooth Substitution Smooth Instance

Assume HSp = (Sp, Sp′) is a hierarchic specification with the base specification Sp = $(\Sigma, \mathscr{C})$ and the body Sp′ = $(\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

A substitution $\sigma$ is called ***smooth*** if no term in its image contains an extension subterm:

$$\sigma \textit{ smooth} \overset{\text{def}}{\iff} \forall t \in \textit{im}(\sigma). \forall p \in \rho(t) : t/p \notin T^{E}_{\Omega'}(\mathcal{X}').$$

A term $t'$ is called a ***smooth instance*** of a term $t$, if $t' = t\sigma$ for some smooth substitution $\sigma$. The set of all smooth ground instances of $t$ is denoted by $\mathit{smgi}(t)$:

$$\mathit{smgi}(t) \overset{\text{def}}{=} \{t\sigma \mid \sigma \text{ smooth}, t\sigma \in T_{\Omega'}\}.$$

■

The notion of a smooth (ground) instance is naturally lifted to inferences and all expressions, particularly to literals, clauses, and sets thereof. Note that any smooth substitution is simple, but not the other way around, in general. Moreover, given an expression $e$ and its smooth instance $e' = e\sigma$, where $\sigma$ is a smooth substitution, a subterm $t' = e'/p$ of $e'$ at a position $p \in \rho(e')$ is an extension term *if and only if* the subterm $t = e/p$ of $e$ at the same position $p \in \rho(e)$ is so; in other words, the application of a smooth substitution does not introduce new extension symbol occurrences.

DEFINITION 3.84 ► Locally Sufficiently Complete Clause Sets

Assume HSp = (Sp, Sp′) is a hierarchic specification with the base specification Sp = $(\Sigma, \mathscr{C})$ and the body Sp′ = $(\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively.

A set $N$ of $\Sigma'$-clauses is called ***locally sufficiently complete with respect to smooth instances*** if for every weak model $\mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}}$ of the set $\mathit{sgi}(N)$ of all simple ground instances of $N$, every base algebra $\mathcal{A} \in \mathscr{C}$ monomorphic[1] to $\mathcal{A}'$, and every ground extension term[2] $t'$ occurring in the set[3] $\mathit{smgi}(N) \setminus \mathcal{R}^{\mathcal{F}}_{F}(\mathit{smgi}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})$ of all smooth ground instances of $N$ non-redundant for $\mathit{smgi}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ there

[1]The existence of such base algebra $\mathcal{A}$ is guaranteed, because, by definition, for any weak algebra $\mathcal{A}'$ there is a base algebra $\mathcal{A}$ which is monomorphic to $\mathcal{A}'$; see Definition 3.73 of a weak algebra, page 89.

[2]Please, recall Definitions 3.16 and 3.17 of different kinds of terms, page 39.

[3]Please recall Definition 3.49 of clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, page 62, and Definition 3.40 of the standard redundancy criterion $\mathcal{R}^{\mathcal{F}}$ for ground clauses, page 56.

exists a ground base term $t$ such that $t'$ equals $t$ under $\mathcal{A}'$. We write $\mathcal{LSC}_{\mathsf{HSp}}$ to denote the class of all such sets of clauses:

$$\mathcal{LSC}_{\mathsf{HSp}} \overset{\text{def}}{=} \{ N \in Cl_{\Sigma'} \mid (\forall \mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}})(\forall \mathcal{A} \in \mathscr{C})(\forall t' \in T^{E}_{\Omega'})(\exists t \in T_{\Omega}) :$$

- $\mathcal{A}$ is monomorphic to $\mathcal{A}'$,
- $t'$ occurs in $smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_{F}(smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})$, and
- $\mathcal{A}' \models sgi(N) \Rightarrow \mathcal{A}' \models t' \approx t \}$

■

For the sake of conciseness, if a clause set $N$ is *locally sufficiently complete with respect to smooth instances*, we simply say that $N$ is ***locally sufficiently complete***.

Expanding the notion of a weak model and using the Weak Algebra Theorem (Theorem 3.76), the definition of the class of all locally sufficiently complete clause sets can be equivalently reformulated as follows:

$$\mathcal{LSC}_{\mathsf{HSp}} = \{N \in Cl_{\Sigma'} \mid (\forall \mathcal{A} \in \mathscr{C})(\forall t' \in T^{E}_{\Omega'})(\exists t \in T_{\Omega}) : \\ t' \text{ occ. in } smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_{F}(smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \Rightarrow sgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \models t' \approx t\}$$

Using Lemma 3.61(ii) and the fact that $sgi_{\mathcal{A}}(N) \subseteq sgi(N)$, the definition can be equivalently rewritten even further:

$$\mathcal{LSC}_{\mathsf{HSp}} = \{N \in Cl_{\Sigma'} \mid (\forall \mathcal{A} \in \mathscr{C})(\forall t' \in T^{E}_{\Omega'})(\exists t \in T_{\Omega}) : \\ t' \text{ occ. in } smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_{F}(smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \Rightarrow N_{\mathcal{A}} \models t' \approx t\},$$

where $N_{\mathcal{A}} = sgi_{\mathcal{A}}(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, and $sgi_{\mathcal{A}}(N)$ is the set of all simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances of $N$ (see Definitions 3.55 and 3.56 of (the set of all) simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances and a set $N_{\mathcal{A}}$, respectively).

## 3.5.2 Sufficiently Defining Rewrite Systems

As discussed at the very beginning of the current section, our concern is to construct a rewrite system $R'$, such that the corresponding Herbrand interpretation $\mathcal{I}' = T_{\Omega'}/R'$ would be a hierarchic model of an input locally sufficiently complete clause set $N$. Next we formally define the constituents $R^{EF}$, $R^{E}$, $R^{SD}$ of a required rewrite system $R' = R^{EF} \cup R^{E} \cup R^{SD}$. The purpose of the rewrite system $R^{E}$ is to preserve the interpretation of those extension terms that occur in $N$, whereas $R^{EF}$ is aimed at preserving the interpretation of terms that contain no extension subterms (*extension-free* terms). The purpose of $R^{SD}$ is to sufficiently define those extension terms that are not sufficiently defined by $N$.

We define the rewrite systems $R^{EF}$, $R^{E}$, $R^{SD}$ with respect to an abstract $\Sigma'$-algebra $\mathcal{A}'$ and a reduction ordering $\succ$ total on ground $\Sigma'$-terms. Afterwards, given a locally sufficiently complete clause set $N$, we instantiate $\mathcal{A}'$ with an appropriate weak model of the set $sgi(N)$, that agrees with conditions of the definition of a locally sufficiently complete clause set.

To define a rewrite system $R^{EF}$ we make use of an auxiliary rewrite system $\tilde{R}^{EF}$ which contains all rewrite rules between every ground extension-free term $t$ and the minimal element $m^{\succ}_{\mathcal{A}'}(t)$ in the equivalence class $[t]_{\mathcal{A}'}$ of $t$ in $\mathcal{A}'$; the class $[t]_{\mathcal{A}'}$ consists of all ground $\Sigma'$-terms whose interpretation under $\mathcal{A}'$ equals that of

$t$:

$$[t]_{\mathcal{A}'} \stackrel{\text{def}}{=} \{s \in T_{\Omega'} \mid s_{\mathcal{A}'} = t_{\mathcal{A}'}\}$$
$$m^{>}_{\mathcal{A}'}(t) \stackrel{\text{def}}{=} min_{>}([t]_{\mathcal{A}'})$$

A rewrite system $R^{EF}$ relates to $\tilde{R}^{EF}$ same as $R^{\approx}_{\mathcal{A}}$ relates to $\tilde{R}^{\approx}_{\mathcal{A}}$ (see Section 3.4.3, Definition 3.46).

DEFINITION 3.85 ► Rewrite systems: $\tilde{R}^{EF}$ $R^{EF}$ $R^{E}$ $R^{SD}$

Assume HSp = (Sp, Sp′) is a hierarchic specification with the base specification Sp = $(\Sigma, \mathscr{C})$ and the body Sp′ = $(\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $>$ be a reduction ordering total on ground $\Sigma'$-terms, and $\mathcal{A}' \in \mathscr{A}_{\Sigma'}$ a $\Sigma'$-algebra.

We define a ground rewrite system $\tilde{R}^{EF}$ as the set of all rewrite rules between every ground extension-free term $t \in T_{\Omega'}$ and the smallest term in its equivalence class $[t]_{\mathcal{A}'}$ different from $t$:

$$\tilde{R}^{EF} \stackrel{\text{def}}{=} \{t \to m^{>}_{\mathcal{A}'}(t) \mid t \in T_{\Omega'}, \forall p \in \rho(t) : t/p \notin T^{E}_{\Omega'}, t \neq m^{>}_{\mathcal{A}'}(t)\}$$

We define a rewrite system $R^{EF}$ to be the reduced subset of $\tilde{R}^{EF}$, that is, the set of all rules $(l \to r) \in \tilde{R}^{EF}$ such that $l$ is not reducible by $\tilde{R}^{EF}_{A} \setminus \{l \to r\}$:

$$R^{EF} \stackrel{\text{def}}{=} \{(l \to r) \in \tilde{R}^{EF}_{A} \mid l = l{\downarrow}_{\tilde{R}^{EF}_{A} \setminus \{l \to r\}}\}.$$

We define a ground rewrite system $R^{E}$ as the set of all rewrite rules $(l \to r)$ with unique left-hand-sides, such that $l$ is a ground smooth extension term reduced with respect to the rewrite system $R^{EF}$, and $r$ is an arbitrary ground base term that equal under $\mathcal{A}'$ to $l$:

$(l \to r) \in R^{E} \stackrel{\text{def}}{\iff} l \in T^{E}_{\Omega'}$ and $r \in T_{\Omega}$ such that:

- $\forall (l' \to r') \in R^{E} : l = l' \Rightarrow r = r'$,
- $\mathcal{A}' \models l \approx r$,
- $l$ smooth, and
- $l = l{\downarrow}_{R^{EF}}$

We define a ground rewrite system $R^{SD}$ as the set of all rewrite rules $(l \to r)$ with unique left-hand-sides, such that $l$ is a ground smooth extension term, that is reduced with respect to the rewrite system $R^{EF}$ and *not equal* under $\mathcal{I}$ to any base term, and $r$ is an arbitrary ground base term:

$(l \to r) \in R^{SD} \stackrel{\text{def}}{\iff} l \in T^{E}_{\Omega'}$ and $r \in T_{\Omega}$ such that:

- $\forall (l' \to r') \in R^{SD} : l = l' \Rightarrow r = r'$,
- $\forall t \in T_{\Omega} : \mathcal{A}' \not\models l \approx t$,
- $l$ smooth, and
- $l = l{\downarrow}_{R^{EF}}$

■

Please note, that the rewrite systems $\tilde{R}^{EF}$, $R^{EF}$, $R^{E}$, and $R^{SD}$ are always well-defined irregardless of the underlying algebra $\mathcal{A}'$. Some of the rewrite systems $\tilde{R}^{EF}$, $R^{EF}$, $R^{E}$, or $R^{SD}$ might be empty for a particular $\mathcal{A}'$, but $R^{E} \cup R^{SD}$ can be empty only if the enrichment operator set $\Omega'' = \Omega' \setminus \Omega$ contains no extension symbol (i.e. no free function symbol ranging into a base sort).

Next we assert two important properties of rewrite systems $\tilde{R}^{EF}$ and $R^{EF}$, namely: they define the same set of normal forms and induce the same equality relation on ground $\Sigma'$-terms. The properties will be required to show that the Herbrand interpretation $\mathcal{I}' = T_{\Omega'}/R'$ is a hierarchic model of a locally sufficiently complete clause set $N$, where $R' = R^{EF} \cup R^{E} \cup R^{SD}$.

◄ PROPOSITION 3.86

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Rewrite systems* $\tilde{R}^{EF}$ *and* $R^{EF}$ *define the same set of normal forms.*

◄ PROOF

Analogous to the proof of Lemma 3.47. ■

◄ COROLLARY 3.87

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

$\tilde{R}^{EF}$ *and* $R^{EF}$ *induce the same equality relation on ground* $\Sigma'$*-terms.*

◄ PROOF

Follows from Definition 3.85 of $\tilde{R}^{EF}$ and $R^{EF}$, Proposition 3.86, and Birkhoff's Theorem (Theorem 2.28). ■

Next we show that a rewrite system $R' = R^{EF} \cup R^{E} \cup R^{SD}$ is itself sufficiently complete, in the sense that it rewrites any extension term to a base term, and convergent, if underlied by an appropriate ordering.

◄ PROPOSITION 3.88

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Assume* $\succ$ *is a reduction ordering total on ground terms which orients any extension term greater than any extension-free one.*

*If* $\succ$ *is the ordering underlying a rewrite system* $R' = R^{EF} \cup R^{E} \cup R^{SD}$, *then in any model of* $R'$ *all ground smooth extension terms are equal to some base terms:*

$$\forall\, t \in T^{E}_{\Omega'}.\, \exists\, s \in T_{\Omega} : \ t \text{ smooth } \Rightarrow R' \models t \approx s.$$

◄ PROOF

Since in $\succ$ any term containing an extension subterm is greater than any extension-free term, any ground extension-free term can be rewritten in $R'$ only to a ground extension-free term. Moreover, ground extension-free terms can be rewritten in $R'$ only by rules from $R^{EF}$. Consequently, given a ground smooth extension term $t$, any term $t'$ obtained from $t$ as the result of a rewriting $t \to^{*}_{R^{EF}} t'$ is also a ground smooth extension term. Thus, for any such $t$, its normal form $t{\downarrow}_{R^{EF}}$ is a ground smooth extension term. By Definition 3.85, there exists a rewrite rule $(l \to r) \in R^{E} \cup R^{SD}$, such that $l = t{\downarrow}_{R^{EF}}$ and $r \in T_{\Omega}$. Thus, $t \to^{*}_{R'} r$. By Birkhoff's Theorem (Theorem 2.28), it implies that $R' \models t \approx r$. ■

From the Extension Terms Lemma (Lemma 3.19) it follows that all models of $R' = R^{EF} \cup R^{E} \cup R^{SD}$ interpret any ground non-base term of a base sort the same as some base term:

$$\forall\, t \in T_{\Omega'}(\mathcal{S}) \setminus T_{\Omega}.\, \exists\, s \in T_{\Omega} : R' \models t \approx s.$$

◄ PROPOSITION 3.89

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$. *Assume* $\succ$ *is a reduction ordering total on* *ground $\Sigma'$-terms which orients any extension term greater than any extension-free one, and any non-base term greater than any base one.*

*The rewrite system $R' = R^{EF} \cup R^{E} \cup R^{SD}$ is convergent, if $\succ$ is the ordering underlying $R'$.*

PROOF ► According to Theorem 2.21, a rewrite system is convergent if it is terminating and left-reduced. According to Theorem 2.20, a rewrite system is terminating iff the left-hand-side of every rule in it is greater than its right-hand-side with respect to some reduction ordering.

According to Definition 3.85, the rewrite system $R^{EF}$ is a subset of a rewrite system $\tilde{R}^{EF}$. The right-hand-side $r$ of any rule $(l \to r) \in \tilde{R}^{EF}$ is the smallest term in the equivalence class $[l]_{\mathcal{A}'}$ of $l$, for the underlying algebra $\mathcal{A}'$, and $l \neq r$, therefore $l \succ r$. Consequently, $l \succ r$ for any rule $(l \to r) \in R^{EF}$. The left-hand-side $l$ of any rule $(l \to r) \in R^{E} \cup R^{SD}$ is a ground extension term, hence non-base, and the right-hand-side $r$ a ground base term. In the reduction ordering $\succ$, any ground non-base term is greater than any ground base one, hence $l \succ r$ for every rule $(l \to r) \in R^{E} \cup R^{SD}$. These implies $R' = R^{EF} \cup R^{E} \cup R^{SD}$ is terminating.

The rewrite systems $R^{EF}$, $R^{E}$, $R^{SD}$ are left-reduced apart: $R^{EF}$ is left-reduced by construction, each of $R^{E}$ and $R^{SD}$ is left-reduced, because the left-hand-side of any rule in them is a *unique smooth* extension term. We are to show that the union of the rewrite systems is left-reduced as well. Consider three arbitrary rules $(l_1 \to r_1) \in R^{EF}$, $(l_2 \to r_2) \in R^{E}$, and $(l_3 \to r_3) \in R^{SD}$:

- the left-hand-sides $l_2$ and $l_3$ are in its normal form with respect to $R^{EF}$, therefore, no rewriting from $l_2$ or $l_3$ by $l_1 \to r_1$ is possible;
- a rewriting from $l_1$ by $(l_2 \to r_2)$ or $(l_3 \to r_3)$ is impossible, because $l_2$ and $l_3$ are extension terms and $l_1$ contains no extension subterm;
- a rewriting from $l_2$ by $(l_3 \to r_3)$ or from $l_3$ by $(l_2 \to r_2)$ below the top position is impossible, because no strict subterm of $l_2$ or $l_3$ is an extension term since $l_2$ and $l_3$ are smooth.
- a rewriting from $l_2$ by $(l_3 \to r_3)$ or from $l_3$ by $(l_2 \to r_2)$ at the top position is impossible, because $l_2$ is equal under the underlying algebra $\mathcal{A}'$ to some base term (namely, $r_2$) whereas $l_3$ is equal under $\mathcal{A}'$ to no base term, hence $l_2 \neq l_3$.

Since the rules $(l_1 \to r_1)$, $(l_2 \to r_2)$, and $(l_3 \to r_3)$ have been selected from the respective rewrite systems arbitrarily, we conclude that the left-hand-side of any rule from one of $R^{EF}$, $R^{E}$, or $R^{SD}$ is in its normal form with respect to the other two rewrite systems. Consequently, $R' = R^{EF} \cup R^{E} \cup R^{SD}$ is left-reduced. ■

In the following two propositions, we show that given a $\Sigma'$-algebra $\mathcal{A}'$, a rewrite system $R' = R^{EF} \cup R^{E} \cup R^{SD}$ built with respect to $\mathcal{A}'$ agrees with $\mathcal{A}'$:

1. on all extension-free terms, and
2. on all extension terms occurring in the set $smgi(N) \setminus \mathcal{R}_F^{\mathcal{F}}(smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})$ of all smooth ground instances of locally sufficiently complete clause set $N$ that are non-redundant for $smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, whenever $\mathcal{A}'$ is a model of $sgi(N)$.

◂ PROPOSITION 3.90

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Assume* $\succ$ *is a reduction ordering total on ground* $\Sigma'$*-terms which orients any extension term greater than any extension-free one, and any non-base term greater than any base one.*

*Let* $\mathcal{A}'$ *be a* $\Sigma'$*-algebra, and* $R' = R^{EF} \cup R^{E} \cup R^{SD}$ *a rewrite system built with respect to* $\mathcal{A}'$ *and the ordering* $\succ$*. Any two ground extension-free terms* $t, t' \in T_{\Omega'}$ *are equal under* $\mathcal{A}'$ *if and only if their normal forms in* $R'$ *coincide:*

$$\forall t, t' \in T_{\Omega'} : t, t' \text{ extension-free} \ \Rightarrow\ (\mathcal{A}' \models t \approx t' \ \Leftrightarrow\ t\!\downarrow_{R'} = t'\!\downarrow_{R'})$$

◂ PROOF

First, let us make the following observation regarding rewritings in $R' = R^{EF} \cup R^{E} \cup R^{SD}$. Let $t \in T_{\Omega'}$ be an arbitrary ground extension-free $\Sigma'$-term. Since extension terms are greater in $\succ$ than any extension-free term, any rewriting $t \rightarrow^{*}_{R'} t'$, for some $n \geq 1$, can only be performed using rules $l \rightarrow r$ such that $l$ and $r$ contain no extension subterms. Therefore, all such rules come from the rewrite system $R^{EF} \subseteq R'$. Since the left-hand-side of any rule $(l' \rightarrow r') \in R^{E} \cup R^{SD}$ is an extension term, it follows that any rewriting $t \rightarrow^{*}_{R'} t'$ is identical to the corresponding rewriting $t \rightarrow^{*}_{R^{EF}} t'$. Consequently, the normal forms of $t$ in $R'$ and $R^{EF}$ are the same: $t\!\downarrow_{R'} = t\!\downarrow_{R^{EF}}$.

**The "⇐" direction.** Assume $t, t' \in T_{\Omega'}$ are ground extension-free terms, then:

$$\begin{array}{rrcll}
 & t\!\downarrow_{R'} & = & t'\!\downarrow_{R'} & \\
\Leftrightarrow & t\!\downarrow_{R^{EF}} & = & t'\!\downarrow_{R^{EF}} & \text{// by the observ. above} \\
\Leftrightarrow & R^{EF} & \models & t \approx t' & \text{// by Birkhoff's Theorem (Thm. 2.28),} \\
 & & & & \text{// as } R^{EF} \subseteq R' \text{ convergent, Prop. 3.89} \\
\Rightarrow & \mathcal{A}' & \models & t \approx t' & \text{// as } \mathcal{A}' \models R^{EF} \text{ by construction of } R^{EF}
\end{array}$$

**The "⇒" direction.** Assume $\mathcal{A}' \models t \approx t'$. Let[1] $t'' = m^{\succ}_{\mathcal{A}'}(t) = m^{\succ}_{\mathcal{A}'}(t')$ be the minimal term in the equivalence class $[t]_{\mathcal{A}'} = [t']_{\mathcal{A}'}$ of $t$ and $t'$ in $\mathcal{A}'$. Since $t$ and $t'$ are extension-free, the rewrite system $\tilde{R}^{EF}$ contains the rewrite rules $(t \rightarrow t'')$ and $(t' \rightarrow t'')$. Consequently:

$$\begin{array}{rrcll}
 & t\!\downarrow_{\tilde{R}^{EF}} & = & t'\!\downarrow_{\tilde{R}^{EF}} = t'' & \text{// as } \tilde{R}^{EF} \text{ is right-reduced} \\
\Leftrightarrow & t\!\downarrow_{R^{EF}} & = & t'\!\downarrow_{R^{EF}} & \text{// by Prop. 3.86} \\
\Leftrightarrow & t\!\downarrow_{R'} & = & t'\!\downarrow_{R'} & \text{// by the observ. above}
\end{array}$$

■

◂ PROPOSITION 3.91

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Assume* $\succ$ *is a reduction ordering total on ground*

[1]Please recall that for any ground $\Sigma'$-term $t$, we write $m^{\succ}_{\mathcal{A}'}(t)$ to denote the minimal term in $[t]_{\mathcal{A}'}$; where $[t]_{\mathcal{A}'}$ stands for the equivalence class of $t$ in $\mathcal{A}'$; the class $[t]_{\mathcal{A}'}$ consists of all ground terms whose interpretation under $\mathcal{A}'$ equals that of $t$:

$$\begin{aligned}
[t]_{\mathcal{A}'} &\overset{\text{def}}{=\!=} \{s \in T_{\Omega'} \mid s_{\mathcal{A}'} = t_{\mathcal{A}'}\} \\
m^{\succ}_{\mathcal{A}'}(t) &\overset{\text{def}}{=\!=} min_{\succ}([t]_{\mathcal{A}'})
\end{aligned}$$

*$\Sigma'$-terms which orients any extension term greater than any extension-free one, and any non-base term greater than any base one.*

*Let $\mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}}$ be a weak algebra, and $R' = R^{EF} \cup R^{E} \cup R^{SD}$ a rewrite system built with respect to $\mathcal{A}'$ and the ordering $\succ$. Let $N \in \mathcal{LSC}_{\mathsf{HSp}}$ be a locally sufficiently complete clause set. If $\mathcal{A}'$ is a model of the set $sgi(N)$ of all simple ground instances of $N$, then every ground extension term $t \in T^{E}_{\Omega'}$, occurring in the set $smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_{F}(smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})$ of all smooth ground instances of $N$ non-redundant for $smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, rewrites in $R'$ to a ground base term $s \in T_{\Omega}$ such that $\mathcal{A}' \models t \approx s$:*

$$\mathcal{A}' \models sgi(N) \Rightarrow \big(\forall t \in T^{E}_{\Omega'} : t \text{ occurs in } smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_{F}(smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \Rightarrow \\ \exists s \in T_{\Omega} : t{\downarrow}_{R'} = s \text{ and } \mathcal{A}' \models t \approx s\big).$$

PROOF ► First, let us make the following observation regarding rewritings in $R' = R^{EF} \cup R^{E} \cup R^{SD}$. Let $t \in T_{\Omega'}$ be an arbitrary ground extension-free $\Sigma'$-term. Since extension terms are greater in $\succ$ than any extension-free term, any rewriting $t \rightarrow^{*}_{R'} t'$, for some $n \geq 1$, can only be performed using rules $l \rightarrow r$ such that $l$ and $r$ contain no extension subterms. Therefore, all such rules come from the rewrite system $R^{EF} \subseteq R'$. Since the left-hand-side of any rule $(l' \rightarrow r') \in R^{E} \cup R^{SD}$ is an extension term, it follows that any rewriting $t \rightarrow^{*}_{R'} t'$ is identical to the corresponding rewriting $t \rightarrow^{*}_{R^{EF}} t'$. Consequently, the normal forms of $t$ in $R'$ and $R^{EF}$ are the same: $t{\downarrow}_{R'} = t{\downarrow}_{R^{EF}}$. Moreover, any rule in $R^{EF}$ rewrites an extension-free term to another extension-free term, because left-hand-sides of all rules in $R^{EF}$ are extension-free terms, and any extension-free term is smaller in $\succ$ than any extension term.

Let $t \in T^{E}_{\Omega'}$ be an arbitrary ground extension term occurring in the set $smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_{F}(smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})$. As $N$ is locally sufficiently complete and $\mathcal{A}' \models sgi(N)$, we know from Definition 3.84 of a locally sufficiently complete clause set that $\mathcal{A}' \models t \approx s'$, for some ground base term $s' \in T_{\Omega}$.

Next we show that $t{\downarrow}_{R'} = s = s'{\downarrow}_{R'}$, for some ground base term $s \in T_{\Omega}$ such that $\mathcal{A}' \models s \approx s'$, by induction on the number $occ_{E}(t)$ of occurrences of extension subterms in $t$, formally defined as follows:

$$occ_{E}(t) \overset{\text{def}}{=} |\{p \in \rho(t) \mid t/p \in T^{E}_{\Omega'}\}|.$$

*Induction base.* Assume $occ_{E}(t) = 1$. Then $t$ is a smooth extension term. Let $t'$ be the normal form of $t$ in $R^{EF}$:

$$\begin{array}{lcl} & t' = t{\downarrow}_{R^{EF}} & \\ \Rightarrow & \mathcal{A}' \models t' \approx t & \text{// as } R^{EF} \subseteq \tilde{R}^{EF} \text{ and } \mathcal{A}' \models \tilde{R}^{EF}, \\ & & \text{// and by Birkhoff's Theorem} \\ \Leftrightarrow & \mathcal{A}' \models t' \approx s' & \text{// as } \mathcal{A}' \models t \approx s' \\ \Rightarrow & (t' \rightarrow s'') \in R^{E} & \text{// by Def. 3.85 of } R^{E}, \\ & & \text{// for some } s'' \in T_{\Omega} \text{ such that } \mathcal{A}' \models s' \approx s'' \\ \Rightarrow & t' \rightarrow_{R^{E}} s'' & \end{array}$$

Let $s = s''{\downarrow}_{R'}$ be the normal form of $s''$ in $R'$. Moreover, since $s''$ is extension-free, we conclude by the observation above that $s = s''{\downarrow}_{R'} = s''{\downarrow}_{R^{EF}}$. Thus, we have:

$$t \rightarrow^{*}_{R^{EF}} t' \rightarrow_{R^{E}} s'' \rightarrow^{*}_{R^{EF}} s,$$

Since $R^{EF} \subseteq \tilde{R}^{EF}$, $\mathcal{A}' \models \tilde{R}^{EF}$ by construction, and $\mathcal{A}' \models t \approx s' \approx s''$, we conclude $\mathcal{A}' \models s'' \approx s$ and therefore $\mathcal{A}' \models t \approx s$. As $R'$ is convergent, Proposition 3.89, the term $s$ is the unique normal form of $t$ in $R'$. As any ground base term is smaller in $\succ$ than any non-base term, and $s''$ is a ground base term, we conclude that $s$ is also base. Altogether, $t{\downarrow_{R'}} = s$, $\mathcal{A}' \models t \approx s$, and $s \in T_\Omega$.

*Induction hypothesis.* Suppose the assertion holds for all terms $t \in T^E_{\Omega'}$ occurring in $smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_F(smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})$ such that $occ_E(t) \leq n$, for some $n \geq 1$.

*Induction step.* Assume $occ_E(t) = n+1$. Let $t'_1, \ldots, t'_m$ be all the outermost strict extension subterms of $t$, that is, $t'_i = t/p_i \in T^E_{\Omega'}$ for some *non-top* positions $p_i \in \rho(t)$ such that:

$$\forall q \in \rho(t) : p_i > q > \varepsilon \Rightarrow t/q \notin T^E_{\Omega'}.$$

for all $i \in \{1, \ldots, m\}$. Because all $t'_i$ are strict subterms of $t$, we know that each $t'_i$ occurs in the set $smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_F(smgi(N) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})$, and $1 \leq occ_E(t'_i) \leq n$. By Induction hypothesis, there exist ground base terms $s'_i \in T_\Omega$ such that:

$$\begin{aligned} t'_i{\downarrow_{R'}} &= s'_i, \text{ and} \\ \mathcal{A}' \models t'_i &\approx s'_i \end{aligned}$$

for every $i \in \{1, \ldots, m\}$. Consequently, there is a rewriting:

$$t[t'_1, \ldots, t'_m] \rightarrow^*_{R'} t[s'_1, \ldots, s'_m]$$

The term $t[s'_1, \ldots, s'_m]$, obtained from $t = t[t'_1, \ldots, t'_m]$ by rewriting in $R'$ the outermost occurrences of $t'_1, \ldots, t'_m$ to $s'_1, \ldots, s'_m$, respectively, is a ground smooth extension term, i.e. the only extension symbol occurring in $t[s'_1, \ldots, s'_m]$ is the top symbol. Let $t''$ be the normal form of $t[s'_1, \ldots, s'_m]$ with respect to the rewrite system $R^{EF}$, that is

$$t'' = t[s'_1, \ldots, s'_m]{\downarrow_{R^{EF}}}$$

Since $\mathcal{A}' \models R^{EF}$, we conclude

$$\mathcal{A}' \models t[s'_1, \ldots, s'_m] \approx t''$$

Moreover, every single rewrite step in a rewriting $t[s'_1, \ldots, s'_m] \rightarrow^*_{R^{EF}} t''$ takes place below the top position, because any rule in $R^{EF}$ rewrites an extension-free term to another extension-free term (according to the observation above). Hence, $t''$ is a smooth extension term. Besides,

$$\begin{array}{lll} & \mathcal{A}' \models t'_i \approx s'_i & \text{// for every } i \in \{1, \ldots, m\} \\ \Rightarrow & \mathcal{A}' \models t \approx t[s'_1, \ldots, s'_m] & \\ \Leftrightarrow & \mathcal{A}' \models t[s'_1, \ldots, s'_m] \approx s' & \text{// as } \mathcal{A}' \models t \approx s' \\ \Leftrightarrow & \mathcal{A}' \models t'' \approx s' & \text{// as } \mathcal{A}' \models R^{EF} \text{ and } t[s'_1, \ldots, s'_m] \rightarrow^*_{R^{EF}} t'' \\ \Rightarrow & (t'' \rightarrow s'') \in R^E & \text{// by Def. 3.85 of } R^E, \\ & & \text{// for some } s'' \in T_\Omega \text{ such that } \mathcal{A}' \models s' \approx s'' \\ \Rightarrow & t'' \rightarrow_{R^E} s'' & \end{array}$$

Let $s = s''{\downarrow_{R'}}$ be the normal form of $s''$ in $R'$. Since $s''$ is extension-free, we conclude by the observation above that $s = s''{\downarrow_{R'}} = s''{\downarrow_{R^{EF}}}$. Thus, we have:

$$t \rightarrow^*_{R'} t[s'_1, \ldots, s'_m] \rightarrow_{R^{EF}} t'' \rightarrow_{R^E} s'' \rightarrow^*_{R^{EF}} s$$

From $\mathcal{A}' \models R^{EF}$, $\mathcal{A}' \models \{t \approx t[s'_1, \ldots, s'_m], t'' \approx s', s' \approx s''\}$, and $s'' \rightarrow^*_{R^{EF}} s$, it follows that $\mathcal{A}' \models t \approx s$. As $R'$ is convergent, the term $s$ is the unique normal form of $t$ in $R'$. As any ground base term is smaller in $\succ$ than any non-base term, and $s''$ is a ground base term, we conclude that $s$ is also base. Altogether, $t{\downarrow_{R'}} = s$, $\mathcal{A}' \models t \approx s$, and $s \in T_\Omega$. ∎

Note that in the above Proposition, the normal form $s$ of $t$ in the rewrite system $R' = R^{EF} \cup R^E \cup R^{SD}$ is obtained only by applying rewrite rules from $R^{EF} \cup R^E$ and no rule from $R^{SD}$, that is:

$$s = t{\downarrow_{R'}} = t{\downarrow_{R^{EF} \cup R^E}} .$$

### 3.5.3 Model Existence and Refutational Completeness

Assume $M = N_\infty$ is the limit of a fair SUP(T)-derivation from a locally sufficiently complete set $N$ of abstracted clauses, and $\square \notin M$. Since $M$ is the limit of a fair derivation, the clause set $M$ is saturated. By the Hierarchic Saturation Theorem (Theorem 3.72), the clause set[1] $M_\mathcal{A} = sgi_\mathcal{A}(M) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$ is saturated with respect to the flat superposition calculus $(\mathcal{F}, \mathcal{R}^\mathcal{F})$, for a base algebra $\mathcal{A} \in \mathscr{C}$ that satisfies the base subset of $M$. From $\square \notin M$ it follows that $\square \notin M_\mathcal{A}$. According to Lemma 3.41, $M_\mathcal{A}$ has a Herbrand model $\mathcal{I}_{M_\mathcal{A}} = T_{\Omega'}/R_{M_\mathcal{A}}$. We want to show that the Herbrand interpretation $\mathcal{I}' = T_{\Omega'}/R'$ is a hierarchic model of the initial clause set $N$, where $R' = R^{EF} \cup R^E \cup R^{SD}$ and $R^{EF}$, $R^E$, and $R^{SD}$ are constructed with regard to the Herbrand model $\mathcal{I}_{M_\mathcal{A}}$.

In the next proposition, we assert that every weak model $\mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}}$ of the set of all simple ground instances of the limit $N_\infty$ is a model of the set of all simple ground instances of the initial clause set $N$. We need this property to ensure that the candidate Herbrand model $\mathcal{I}_{M_\mathcal{A}}$ does also satisfy the set $sgi(N)$. Note that soundness of the $\mathcal{H}$-inference rules and of the hierarchic redundancy criterion $\mathcal{R}^\mathcal{H}$ with respect to the $\mathscr{C}$-entailment relation $\models_\mathscr{C}$ is not sufficient to guarantee $\mathcal{I}_{M_\mathcal{A}} \models sgi(N)$, as the Herbrand model $\mathcal{I}_{M_\mathcal{A}}$ is not hierarchic, in general.

PROPOSITION 3.92 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let* $N_0$ *be a set of abstracted* $\Sigma'$*-clauses,* $N_0 \vdash N_1 \vdash \ldots$ *an* $\mathcal{H}$*-derivation, and* $N_\infty$ *the limit of the derivation. Then*[2] $sgi(N_\infty) \models_\mathscr{W} sgi(N_0)$, *i.e. every weak model of the set of all simple ground instance of the derivation limit* $N_\infty$ *entails the set of all simple ground instance of the initial clause set* $N_0$.

PROOF ► Put $N = N_0$ and $M = N_\infty$. Let $\mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}}$ be an arbitrary weak model of $sgi(M) = sgi(N_\infty)$. According to the Weak Algebra Theorem (Theorem 3.76), there exists a base algebra $\mathcal{A} \in \mathscr{C}$ such that[3] $\mathcal{A}' \models \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$.

Obviously, $sgi(M) \models_\mathscr{W} sgi(N \cap M)$, and thus $\mathcal{A}' \models sgi(N \cap M)$. Consider the set $N \setminus M$. Every clause in $N \setminus M$ has been deleted at some point in the derivation $N_0 \vdash$

[1]Please, recall Definition 3.49 of clause sets $\mathsf{E}_\mathcal{A}$ and $\mathsf{D}_\mathcal{A}$, page 62; Definition 3.55 of sets of all simple ground $R^\approx_\mathcal{A}$-reduced instances $sgi_\mathcal{A}$, page 67; Definition 3.56 of a clause set $N_\mathcal{A}$, page 67.

[2]Please, recall Definitions 3.73 and 3.74 of a weak model and the $\models_\mathscr{W}$-entailment, respectively, page 89.

[3]Please, recall Definition 3.49 of clause sets $\mathsf{E}_\mathcal{A}$ and $\mathsf{D}_\mathcal{A}$, page 62; Definition 3.55 of sets of all simple ground $R^\approx_\mathcal{A}$-reduced instances $sgi_\mathcal{A}$, page 67; Definition 3.56 of a clause set $N_\mathcal{A}$, page 67; Definitions 3.40 and 3.64 of the redundancy criteria $\mathcal{R}^\mathcal{F}$ and $\mathcal{R}^\mathcal{H}$, pages 56 and 74, respectively.

$N_1 \vdash \ldots$, therefore all clauses in $N \setminus M$ are redundant for the closure $N^* = \bigcup_i N_i$ of the derivation:

$$\begin{array}{rll}
 & N \setminus M \subseteq \mathcal{R}_F^{\mathcal{H}}(N^*) & \text{// by Def. 3.3 of der. relation } \vdash, \\
 & & \text{// and Def. 3.7 of der. limit } N_\infty \\
 & = \mathcal{R}_F^{\mathcal{H}}(M) & \text{// as } M = N_\infty \text{ and by Lemma 3.9(i)} \\
\Rightarrow & sgi_{\mathcal{A}}(N \setminus M) \subseteq \mathcal{R}_F^{\mathcal{F}}(M_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \text{// by Def. 3.64 of } \mathcal{R}_F^{\mathcal{H}} \\
\Rightarrow & M_{\mathcal{A}} \models sgi_{\mathcal{A}}(N \setminus M) & \text{// from Def. 3.40 of } \mathcal{R}_F^{\mathcal{F}}, \\
 & & \text{// where } M_{\mathcal{A}} = sgi_{\mathcal{A}}(M) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \\
\Leftrightarrow & M_{\mathcal{A}} \models sgi_{\mathcal{A}}(N \setminus M) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \text{// as } \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \subseteq M_{\mathcal{A}} \\
\Leftrightarrow & M_{\mathcal{A}} \models sgi(N \setminus M) & \text{// by Lemma 3.61(ii)} \\
\Rightarrow & \mathcal{A}' \models sgi(N \setminus M) & \text{// as } M_{\mathcal{A}} = sgi_{\mathcal{A}}(M) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}, \\
 & & \text{// } sgi_{\mathcal{A}}(M) \subseteq sgi(M),\ \mathcal{A}' \models sgi(M), \\
 & & \text{// and } \mathcal{A}' \models \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}
\end{array}$$

From $\mathcal{A}' \models sgi(N \cap M)$ and $\mathcal{A}' \models sgi(N \setminus M)$ it follows that $\mathcal{A}' \models sgi(N)$. As $\mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}}$ has been picked arbitrarily, we conclude[1] $sgi(M) \models_{\mathscr{W}} sgi(N)$. ■

Having disposed of all the preliminary steps, we prove now the main property of locally sufficiently complete sets of abstracted clauses, namely that any such clause set $N \in \mathcal{LSC}_{\mathsf{HSp}}$ has a hierarchic model whenever a fair SUP(T) derivation does not produce an empty clause □ and the set of all base clauses within the limit of the derivation is theory-consistent.

◂ THEOREM 3.93

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Assume* $\succ$ *is a reduction ordering total on ground* $\Sigma'$*-terms which orients any extension term greater than any extension-free one, and any non-base term greater than any base one.*

*Let* $N_\infty$ *equal the limit of a fair SUP(T)-derivation* $N_0 \vdash N_1 \vdash \ldots$ *with* $\succ$ *as the underlying ordering, where* $N_0 \in \mathcal{LSC}_{\mathsf{HSp}}$ *is a locally sufficiently complete set of abstracted* $\Sigma'$*-clauses. Assume there exists a base algebra* $\mathcal{A} \in \mathscr{C}$ *satisfying the set* $N_\infty \cap Cl_\Sigma$ *of all base clauses within* $N_\infty$*. If* $N_\infty$ *does not contain an empty clause* □*, then* $N_0$ *has a hierarchic model.*

◂ PROOF

Put $N = N_0$ and $M = N_\infty$. Since $M$ is the limit of a fair derivation $N_0 \vdash N_1 \vdash \ldots$, the clause set $M$ is saturated, Lemma 3.9. By the Hierarchic Saturation Theorem (Theorem 3.72), the clause set[2] $M_{\mathcal{A}} = sgi_{\mathcal{A}}(M) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ is saturated with respect to the flat superposition calculus $(\mathcal{F}, \mathcal{R}^{\mathcal{F}})$. From $\square \notin M$ it follows that $\square \notin M_{\mathcal{A}}$. According to Lemma 3.41, $M_{\mathcal{A}}$ has a Herbrand model $\mathcal{I}_{M_{\mathcal{A}}} = T_{\Omega'}/R_{M_{\mathcal{A}}}$. We are to show that the Herbrand interpretation $\mathcal{I}' = T_{\Omega'}/R'$ is a hierarchic model of the initial clause set $N = N_0$, where $R' = R^{EF} \cup R^E \cup R^{SD}$ and $R^{EF}$, $R^E$, and $R^{SD}$ are constructed with regard to the Herbrand model $\mathcal{I}_{M_{\mathcal{A}}}$ in accordance to Definition 3.85.

The rest of the proof is split into three parts, in which we show that: first, $\mathcal{I}'$ is a model of the set $smgi(N)$ of all smooth ground instances of the initial clause

[1] Note that $sgi(N_{i+k}) \models_{\mathscr{W}} sgi(N_i)$ follows by analogous argumentation, where $i, k \geq 0$.

[2] Please, recall Definition 3.49 of clause sets $\mathsf{E}_{\mathcal{A}}$ and $\mathsf{D}_{\mathcal{A}}$, page 62; Definition 3.55 of sets of all simple ground $R^{\approx}_{\mathcal{A}}$-reduced instances $sgi_{\mathcal{A}}$, page 67; Definition 3.56 of a clause set $N_{\mathcal{A}}$, page 67.

set $N = N_0$; second, $\mathcal{I}'$ is a model of the set $gi(N)$ of all ground instances of $N$, consequently, a model of $N$; and, third, $\mathcal{I}'$ is hierarchic algebra.

**Subgoal** $\mathcal{I}' \models smgi(N)$. First, let us make the following observation regarding rewritings in $R' = R^{EF} \cup R^{E} \cup R^{SD}$. Let $t \in T_{\Omega'}$ be an arbitrary ground extension-free $\Sigma'$-term. Since extension terms are greater in $\succ$ than any extension-free term, any rewriting $t \rightarrow^*_{R'} t'$, for some $n \geq 1$, can only be performed using rules $l \rightarrow r$ such that $l$ and $r$ contain no extension subterms. Therefore, all such rules come from the rewrite system $R^{EF} \subseteq R'$. Since the left-hand-side of any rule $(l' \rightarrow r') \in R^{E} \cup R^{SD}$ is an extension term, it follows that any rewriting $t \rightarrow^*_{R'} t'$ is identical to the corresponding rewriting $t \rightarrow^*_{R^{EF}} t'$. Consequently, the normal forms of $t$ in $R'$ and $R^{EF}$ are the same: $t{\downarrow_{R'}} = t{\downarrow_{R^{EF}}}$.

First, we show that $\mathcal{I}_{M_\mathcal{A}}$ is a model of the set $sgi(N)$ of all simple ground instances of the *initial* clause set $N = N_0$. Consider the Herbrand interpretation $\mathcal{I}_{M_\mathcal{A}}$ more elaborately:

$$\begin{array}{rlll} & \mathcal{I}_{M_\mathcal{A}} & \models M_\mathcal{A} & \\ & & \supseteq \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A} & \text{// as } M_\mathcal{A} = sgi_\mathcal{A}(M) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A} \\ \Rightarrow & \mathcal{I}_{M_\mathcal{A}} & \in \mathscr{W}_{\mathsf{HSp}} & \text{// by the Weak Algebra Theorem (Thm. 3.76)} \end{array}$$

Moreover

$$\begin{array}{rlll} & M_\mathcal{A} & \models sgi(M) & \text{// by Lemma 3.61(ii)} \\ \Rightarrow & \mathcal{I}_{M_\mathcal{A}} & \models sgi(N) & \text{// by Prop. 3.92} \end{array}$$

Next, we show that $\mathcal{I}' \models smgi(N)$. Consider an arbitrary clause $C \in smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_{F}(smgi(N) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A})$ from the set of all simple ground instances of $N$ non-redundant for $smgi(N) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$. The clause $C$ is satisfied by $\mathcal{I}_{M_\mathcal{A}}$ as $C \in smgi(N) \subseteq sgi(N)$. We know $\mathcal{I}_{M_\mathcal{A}} \models C$ if and only if $\mathcal{I}_{M_\mathcal{A}}$ satisfies some literal $L = (t_1 \mathrel{\dot{\approx}} t_2) \in C$, where $\dot{\approx} \in \{\approx, \not\approx\}$. Consider two possible case regarding the sign of $L$:

1. *L is positive*, i.e. $L = (t_1 \approx t_2)$. Let $t^i_1, \ldots, t^i_{n_i}$ be all the *outermost*[1] occurrences of extension subterms of each $t_i$, where $n_i \geq 0$ for each $i \in \{1,2\}$. As $N$ is a locally sufficiently complete clause set, and, as shown above, $\mathcal{I}_{M_\mathcal{A}} \models sgi(N)$ and $\mathcal{I}_{M_\mathcal{A}} \in \mathscr{W}_{\mathsf{HSp}}$, we know from Proposition 3.91 that for every term $t^i_j$ there exists a ground base term $s^i_j \in T_\Omega$ such that the normal form of $t^i_j$ in $R'$ equals $s^i_j$, and $t^i_j$ and $s^i_j$ are equal under $\mathcal{I}_{M_\mathcal{A}}$:

$$\begin{array}{r} t^i_j{\downarrow_{R'}} = s^i_j \text{ and} \\ \mathcal{I}_{M_\mathcal{A}} \models t^i_j \approx s^i_j, \end{array}$$

for each $i \in \{1,2\}$ and $j \in \{1,\ldots,n_i\}$. Therefore, for $t_1$ and $t_2$ there are the following rewritings in $R'$:

$$\begin{array}{rcl} t_1 & = & t_1[t^1_1,\ldots,t^1_{n_1}] \rightarrow^*_{R'} t_1[s^1_1,\ldots,s^1_{n_1}] = t'_1 \\ t_2 & = & t_2[t^2_1,\ldots,t^2_{n_2}] \rightarrow^*_{R'} t_2[s^2_1,\ldots,s^2_{n_2}] = t'_2 \end{array}$$

where each term $t'_i = t_i[s^i_1,\ldots,s^i_{n_i}]$ is obtained from $t_i = t_i[t^i_1,\ldots,t^i_{n_i}]$ by rewriting the outermost occurrences of the extension terms $t^i_1,\ldots,t^i_{n_i}$ to the base terms $s^i_1,\ldots,s^i_{n_i}$, respectively. The obtained terms $t'_1$ and $t'_2$ contain no ex-

[1]An occurrence of an extension subterm $t' = t/p$ of a term $t$ at a position $p \in \rho(t)$ is *outermost*, if no subterm of $t$ at a position $q < p$ is an extension term: $\forall q \in \rho(t) : q < p \Rightarrow t/q \notin T^E_{\Omega'}(\mathcal{X}')$.

tension subterms, hence they are extension-free. Thus, we have:

$$\begin{array}{lll}
 & \mathcal{I}_{M_\mathcal{A}} \models t_1 \approx t_1',\ t_2 \approx t_2' & \text{// as } \mathcal{I}_{M_\mathcal{A}} \models t^i_j \approx s^i_j \text{ for each } i \in \{1,2\},\ j \in \{1,\dots,n_i\} \\
\Rightarrow & \mathcal{I}_{M_\mathcal{A}} \models t_1' \approx t_2' & \text{// as } \mathcal{I}_{M_\mathcal{A}} \models t_1 \approx t_2 \\
\Leftrightarrow & t_1'{\downarrow_{R'}} = t_2'{\downarrow_{R'}} & \text{// by Prop. 3.90, as } t_1' \text{ and } t_2' \text{ extension-free} \\
\Leftrightarrow & t_1{\downarrow_{R'}} = t_2{\downarrow_{R'}} & \text{// as } t_i \rightarrow^*_{R'} t_i', \text{ for each } i \in \{1,2\} \\
 & & \text{// and } R' \text{ convergent by Prop. 3.89} \\
\Leftrightarrow & \mathcal{I}' \models t_1 = t_2 & \text{// by Birkhoff's Theorem (Thm. 2.28)} \\
 & & \text{// as } R' \text{ convergent}
\end{array}$$

2. *L is negative*, i.e. $L = t_1 \not\approx t_2$. By the same arguments as used for the case of positive $L$, we conclude that the terms $t_1$ and $t_2$ can be rewritten in $R'$ to some extension-free terms $t_1'$ and $t_2'$ such that $\mathcal{I}_{M_\mathcal{A}} \models t_1 \approx t_1'$ and $\mathcal{I}_{M_\mathcal{A}} \models t_2 \approx t_2'$. On the other hand:

$$\begin{array}{lll}
 & \mathcal{I}_{M_\mathcal{A}} \models t_1 \not\approx t_2 & \\
\Leftrightarrow & \mathcal{I}_{M_\mathcal{A}} \models t_1' \not\approx t_2' & \text{// as } \mathcal{I}_{M_\mathcal{A}} \models t_1 \approx t_1', t_2 \approx t_2' \\
\Leftrightarrow & \mathcal{I}_{M_\mathcal{A}} \not\models t_1' \approx t_2' & \\
\Leftrightarrow & t_1'{\downarrow_{R'}} \neq t_2'{\downarrow_{R'}} & \text{// by Prop. 3.90, as } t_1' \text{ and } t_2' \text{ extension-free} \\
\Leftrightarrow & t_1{\downarrow_{R'}} \neq t_2{\downarrow_{R'}} & \text{// as } t_i \rightarrow^*_{R'} t_i', \text{ for each } i \in \{1,2\} \\
 & & \text{// and } R' \text{ convergent by Prop. 3.89} \\
\Leftrightarrow & \mathcal{I}' \models t_1 \neq t_2 & \text{// by Birkhoff's Theorem as } R' \text{ convergent}
\end{array}$$

Thus, $\mathcal{I}' \models L$, consequently $\mathcal{I}' \models C$. As $C$ has been picked arbitrarily, we conclude $\mathcal{I}' \models smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_F(smgi(N) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A})$.

Next step is to show that $\mathcal{I}' \models smgi(N)$. Let $D = \{t_1 \mathrel{\dot{\approx}} t_2\}$ be an arbitrary clause from $\mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$, where $\dot{\approx}\ \in \{\approx, \not\approx\}$. Since $\mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$ are sets of ground base clauses, the terms $t_1$ and $t_2$ are extension-free terms. Consequently:

$$\begin{array}{lll}
 & \mathcal{I}_{M_\mathcal{A}} \models t_1 \mathrel{\dot{\approx}} t_2 & \text{// as } \mathcal{I}_{M_\mathcal{A}} \models M_\mathcal{A} \supseteq \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A} \\
\Leftrightarrow & t_1{\downarrow_{R'}} \doteq t_2{\downarrow_{R'}} & \text{// by Prop. 3.90, as } t_1' \text{ and } t_2' \text{ extension-free,} \\
 & & \text{// where } \doteq \text{ is } =, \text{ if } \dot{\approx} \text{ is } \approx, \\
 & & \text{// and } \doteq \text{ is } \neq, \text{ if } \dot{\approx} \text{ is } \not\approx \\
\Leftrightarrow & \mathcal{I}' \models t_1 \mathrel{\dot{\approx}} t_2 & \text{// by Birkhoff's Theorem as } R' \text{ convergent}
\end{array}$$

Since $D$ has been picked arbitrarily from $\mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$, we conclude $\mathcal{I}' \models \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$. By the Weak Algebra Theorem (Theorem 3.76), the algebra $\mathcal{I}'$ is weak. On other hand, we have:

$$\begin{array}{lll}
 & \mathcal{I}' \models smgi(N) \setminus \mathcal{R}^{\mathcal{F}}_F(smgi(N) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}) & \text{// as shown above} \\
\Leftrightarrow & \mathcal{I}' \models \big(smgi(N) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}\big) \setminus \mathcal{R}^{\mathcal{F}}_F(smgi(N) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}) & \text{// as } \mathcal{I}' \models \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A} \\
 & \quad \models smgi(N) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A} & \text{// by Thm. 3.42, and} \\
 & & \text{// by Def. 3.1 of } \mathcal{R}_F \\
\Rightarrow & \mathcal{I}' \models smgi(N) &
\end{array}$$

**Subgoal** $\mathcal{I}' \models N$. Assume $\mathcal{A}'$ is an arbitrary model of $R'$. According to Proposition 3.88, any ground extension term is interpreted under $\mathcal{A}'$ as some base term. Consequently, by the Extension Terms Lemma (Lemma 3.19), any ground non-base term of a base sort is interpreted under $\mathcal{A}'$ as some base term:

$$\forall\, t \in T_{\Omega'}(\mathcal{S}) \setminus T_\Omega . \exists\, s \in T_\Omega : \mathcal{A}'(t) = \mathcal{A}'(s).$$

From this we conclude that for any ground substitution $\sigma$ there exists an $\mathcal{A}'$-

equivalent smooth ground substitution $\sigma'$, in the sense that $dom(\sigma') = dom(\sigma)$, and $\forall x \in dom(\sigma') : \mathcal{A}'(x\sigma') = \mathcal{A}'(x\sigma)$. This implies that for any clause $D \in N$, each ground instance $D\sigma \in gi(N)$ is equivalent under $\mathcal{A}'$ to some smooth ground instance $D\sigma' \in smgi(N)$ of the same clause $D$, that is, $\mathcal{A}' \models D\sigma'$ iff $\mathcal{A}' \models D\sigma$. Therefore, $\mathcal{A}' \models smgi(N)$ iff $\mathcal{A}' \models gi(N)$ iff $\mathcal{A}' \models N$. As $\mathcal{A}'$ has been picked arbitrarily, $\mathcal{I}'$ is a model of $R'$ by construction, and $\mathcal{I}' \models smgi(N)$, we conclude $\mathcal{I}' \models N$.

**Subgoal** $\mathcal{I}' \in \mathscr{H}_{\mathsf{HSp}}$. Put $N'$ equal to the set of all ground positive unit clauses constructed from the equations (rewrite rules) in $R'$:

$$N' \overset{\text{def}}{=} \{\rightarrow s \approx t \mid (s \rightarrow t) \in R'\}.$$

Evidently, $\mathcal{I}' \models N'$. As we have shown above, $\mathcal{I}' \models \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$. Hence, since $N'$ is ground, we conclude $\mathcal{I}' \models N' \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} = sgi_{\mathcal{A}}(N') \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} = N'_{\mathcal{A}}$. From Proposition 3.88 and Definition 3.78, it follows that $N'$ is sufficiently complete. According to the Hierarchic Model Lemma (Lemma 3.79), the Herbrand interpretation $\mathcal{I}'$ is a hierarchic model of $N'_{\mathcal{A}}$, hence a hierarchic algebra.

From $\mathcal{I}' \models N$ and $\mathcal{I}' \in \mathscr{H}_{\mathsf{HSp}}$, we conclude $\mathcal{I}'$ is a hierarchic model of $N$. ■

We end up with a refutational completeness result of the SUP(T) calculus for all locally sufficiently complete sets of abstracted clauses provided the base specification Sp is compact.

THEOREM 3.94 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Assume* Sp *is compact. Assume* $\succ$ *is a reduction ordering total on ground* $\Sigma'$*-terms which orients any extension term greater than any extension-free one, and any non-base term greater than any base one.*

*The SUP(T) calculus with* $\succ$ *as the underlying ordering is refutationally complete for all locally sufficiently complete sets of abstracted* $\Sigma'$*-clauses.*

PROOF ► Let $N \in \mathcal{LSC}_{\mathsf{HSp}}$ be an arbitrary locally sufficiently complete set of abstracted $\Sigma'$-clauses. Let $N_0 \vdash N_1 \vdash \ldots$ be a fair SUP(T) derivation from $N = N_0$, and $N_\infty$ the limit of the derivation. By Lemma 3.9, the limit $N_\infty$ is saturated. Consider two possible cases regarding $N_\infty$:

- If $\square \in N_\infty$, then $N_0 \models_{\mathscr{C}} \bot$, as SUP(T) is sound with respect to the $\mathscr{C}$-entailment relation $\models_{\mathscr{C}}$.
- Otherwise, let $\mathcal{A} \in \mathscr{C}$ be a base algebra that satisfies the set $N_\infty \cap Cl_\Sigma$ of all base clauses within $N_\infty$. The existence of $\mathcal{A}$ follows from the compactness of the base specification Sp.

  Indeed, suppose for the sake of contradiction, that $N_\infty \cap Cl_\Sigma$ is satisfiable by no base algebra in $\mathscr{C}$, then by compactness there exists finitely many base clauses $C_1, \ldots, C_n \in N_\infty \cap Cl_\Sigma$, for some $n \geq 1$, such that $C_1, \ldots, C_n \models_{\mathscr{C}} \bot$, where $C_i = (\Lambda_i \parallel \rightarrow)$, for each $i \in \{1, \ldots, n\}$. Hence, there is a Constraint Refutation inference, Definition 3.31:

$$\mathcal{I}\,\frac{\Lambda_1 \parallel \rightarrow \quad \ldots \quad \Lambda_n \parallel \rightarrow}{\square}$$

As $N_\infty$ is saturated, the inference is redundant. According to Definition 3.64 of the hierarchic redundancy criterion $\mathcal{R}^{\mathcal{H}}$, a Constraint Refutation inference is redundant if and only if an empty clause is present in the clause set, a contradiction.

According to Theorem 3.93, $N_0$ has a hierarchic model.

■

## 3.6 Hierarchic Reduction Rules

We want to extend the two rules tautology and subsumption deletion, which are the primary reduction rules in the superposition SUP framework for general FOL clauses [Wei01], onto the case of the hierarchic superposition SUP(T) calculus for abstracted FOL(T) clauses. The extended rules are to be applied in two stages: in first stage the free parts of abstracted clauses are subject to the conditions of the standard rules like if they were general FOL clauses, and in the second stage the constraints of the abstracted clauses are checked to meet extra conditions, sufficient to recognize the whole clauses redundant. Such schema reflects the structure of abstracted clauses and reuses the standard well-studied redundancy detection techniques developed for the FOL clausal fragment.

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$. Consider an abstracted clause $C$:

$$\begin{aligned} C &= \Lambda \parallel \Gamma \to \Delta \\ &= \neg \bigwedge \Lambda \vee \neg \bigwedge \Gamma \vee \bigvee \Delta \end{aligned}$$

The clause $C$ is a tautology relative to $\mathscr{C}$ (valid in all hierarchic algebras), if the free part $\neg \bigwedge \Gamma \vee \bigvee \Delta$ or the base part $\neg \bigwedge \Lambda$ of it is so. The condition on the free part can be approximated by the standard tautology check: indeed, $\models \neg \bigwedge \Gamma \vee \bigvee \Delta$ implies $\models_{\mathscr{C}} \neg \bigwedge \Gamma \vee \bigvee \Delta$. The base part $\neg \bigwedge \Lambda$ is a tautology relative to $\mathscr{C}$ if and only if

$$\begin{array}{lll} & \forall \mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}} : \mathcal{A}' \models gi(\neg \bigwedge \Lambda) & \\ \Leftrightarrow & \mathcal{A}' \models sgi(\neg \bigwedge \Lambda) & \\ \Leftrightarrow & \forall \mathcal{A} \in \mathscr{C} : \mathcal{A} \models sgi(\neg \bigwedge \Lambda) & \text{// as } sgi(\neg \bigwedge \Lambda) \text{ a base formula} \\ \Leftrightarrow & \mathcal{A} \models \forall \vec{x}. \neg \bigwedge \Lambda & \text{// as all } \mathcal{A} \in \mathscr{C} \text{ term-gen.,} \\ & & \text{// where } \vec{x} = var(\Lambda) \\ \Leftrightarrow & \mathcal{A} \not\models \exists \vec{x}. \bigwedge \Lambda & \\ \Leftrightarrow & \exists \vec{x}. \bigwedge \Lambda \models_{\mathscr{C}} \bot & \end{array}$$

i.e. if the existential closure of the constraint is unsatisfiable in the base theory.

Consider two abstracted clauses $C_1$ and $C_2$ such that the free part of $C_1$ subsumes the free part of $C_2$:

$$\begin{aligned} C_1 &= \Lambda_1 \parallel \Gamma_1 \to \Delta_1 = \neg \bigwedge \Lambda_1 \vee \neg \bigwedge \Gamma_1 \vee \bigvee \Delta_1, \\ C_2 &= \Lambda_2 \parallel \Gamma_2 \to \Delta_2 = \neg \bigwedge \Lambda_2 \vee \neg \bigwedge \Gamma_2 \vee \bigvee \Delta_2, \\ & \Gamma_1 \sigma \subseteq \Gamma_2 \text{ and } \Delta_1 \sigma \subseteq \Delta_2, \end{aligned}$$

for some matcher $\sigma$. Next, we determine conditions which have to be imposed on the constraints of the clauses, so that the clause $C_1$ entails $C_2$ relative to $\mathscr{C}$, under the assumption that the free part of $C_1$ subsumes the free part of $C_2$. Suppose $C_1 \not\models_{\mathscr{C}} C_2$, then there exists a hierarchic algebra $\mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}}$ such that $\mathcal{A}' \models C_1$ and $\mathcal{A}' \not\models C_2$. So,

$$\begin{array}{lll} & \mathcal{A}' \not\models C_2 & \\ \Leftrightarrow & \mathcal{A}' \not\models gi(C_2) & \\ \Leftrightarrow & \mathcal{A}' \not\models sgi(C_2) & \text{// as } \mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}} \\ \Leftrightarrow & \mathcal{A}' \not\models C_2 \phi & \text{// for some simple ground} \\ & & \text{// subst. } \phi \text{ with } dom(\phi) = var(C_2) \\ \Leftrightarrow & \mathcal{A}' \not\models \neg \bigwedge \Lambda_2 \phi \text{ and} & \\ & \mathcal{A}' \not\models \neg \bigwedge \Gamma_2 \phi \vee \bigvee \Delta_2 \phi & \end{array}$$

As $\Gamma_1\sigma \subseteq \Gamma_2$ and $\Delta_1\sigma \subseteq \Delta_2$, from $\mathcal{A}' \not\models \neg \bigwedge \Gamma_2\phi \vee \bigvee \Delta_2\phi$ we conclude $\mathcal{A}' \not\models \neg \bigwedge \Gamma_1\sigma\phi \vee \bigvee \Delta_1\sigma\phi$, which under the assumption that $\mathcal{A}' \models C_1$ implies

$$\begin{array}{lll} & \mathcal{A}' \models \neg \bigwedge \Lambda_1\sigma\phi & \\ \Leftrightarrow & \mathcal{A}' \models gi(\neg \bigwedge \Lambda_1\sigma\phi) & \\ \Leftrightarrow & \mathcal{A}' \models sgi(\neg \bigwedge \Lambda_1\sigma\phi) & \text{// as } \mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}} \\ \Leftrightarrow & \mathcal{A}' \models \neg \bigwedge \Lambda_1\sigma\phi\delta & \text{// for every simple ground subst. } \delta \\ & & \text{// with } dom(\delta) = var(\Lambda_1\sigma) \setminus var(C_2) \end{array}$$

Thus, if $\Gamma_1\sigma \subseteq \Gamma_2$ and $\Delta_1\sigma \subseteq \Delta_2$ for some matcher $\sigma$, then $C_1 \not\models_{\mathscr{C}} C_2$ *if and only if* there exist *some* hierarchic algebra $\mathcal{A}'$, *some* simple ground $\phi$ with $dom(\phi) = var(C_2)$ such that for *every* simple ground $\delta$ with $dom(\delta) = var(\Lambda_1\sigma) \setminus var(C_2)$ the following holds

$$\mathcal{A}' \not\models (\Gamma_2 \to \Delta_2)\phi, \text{ and } \mathcal{A}' \not\models \neg \bigwedge \Lambda_2\phi, \text{ and } \mathcal{A}' \models \neg \bigwedge \Lambda_1\sigma\phi\delta.$$

Consequently, $C_1 \models_{\mathscr{C}} C_2$ *if and only if* for *every* hierarchic algebra $\mathcal{A}'$ and *every* simple ground $\phi$ there exists *some* simple ground $\delta$ with domains as defined above such that

$$\mathcal{A}' \models (\Gamma_2 \to \Delta_2)\phi, \text{ or } \mathcal{A}' \models \neg \bigwedge \Lambda_2\phi, \text{ or } \mathcal{A}' \not\models \neg \bigwedge \Lambda_1\sigma\phi\delta,$$

or equivalently,

$$\text{if } \mathcal{A}' \not\models (\Gamma_2 \to \Delta_2)\phi, \text{ then } \mathcal{A}' \models (\bigwedge \Lambda_2\phi \to \bigwedge \Lambda_1\sigma\phi\delta),$$

that is, every hierarchic algebra $\mathcal{A}'$, that does not entail a ground instance of the $C_2$'s free part, validates the implication between the respective ground instance of the $C_2$'s constraint and some corresponding ground instance of $C_1\sigma$'s constraint. This criterion is not practically tractable, in general. We weaken it to fit the theory T fragment (so that it could be checked by a T-solver alone) as follows:

- First of all, due to the principle of modularity, according to which the free reasoning and theory reasoning are hidden from each other, for validating the constraints implication we have to consider all hierarchic algebras $\mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}}$, and not only those ones which do not entail $(\Gamma_2 \to \Delta_2)\phi$.

- Besides, the subsumption matcher $\sigma$ may map a base variable occurring in $\Lambda_1$ to a non-base term, turning thus $\Lambda_1\sigma$ to a non-base formula. In order to avoid such cases, we restrict the free part subsumption by *simple* matchers[1].

- With a simple matcher $\sigma$, the implication $\bigwedge \Lambda_2\phi \to \bigwedge \Lambda_1\sigma\phi\delta$ becomes a base formula. Checking validity of a base formula with respect to all hierarchic algebras $\mathcal{A}' \in \mathscr{H}_{\mathsf{HSp}}$ reduces to checking validity with respect to all base algebras $\mathcal{A} \in \mathscr{C}$.

- Since all base algebras in $\mathscr{C}$ are term-generated, the application of substitutions $\phi$ and $\delta$ can be replaced by the appropriate quantification of the domain variables of $\phi$ and $\delta$.

[1]As the free part of $C_1$ may contain base variables which do not appear in its constraint $\Lambda_1$, a more effective solution is to prohibit the (non-simple) matcher $\sigma$ from mapping the variables occurring in $\Lambda_1$ to non-base terms (all other variables, including base ones, may be mapped by $\sigma$ arbitrarily). All current implementations of SPASS(T) [AKW09a, EKS+11, FHW10, FW11, FKW12b] use this softer subsumption matcher requirement.

Thus, we obtain the following *sufficient* condition for $C_1 \models_{\mathscr{C}} C_2$:

(3.1) ► $$\models_{\mathscr{C}} \forall \vec{x}.\exists \vec{y}.(\bigwedge \Lambda_2 \rightarrow \bigwedge \Lambda_1\sigma),$$

where $\sigma$ a simple free parts subsumption matcher, $\vec{x} = var(C_2) \cap \mathcal{X}$ and $\vec{y} = var(\Lambda_1\sigma) \setminus var(C_2)$. Please, observe the structure of the variable vectors $\vec{x}$ and $\vec{y}$: the vectors correspond to the substitutions $\phi$ and $\delta$ respectively, thus $\vec{x}$ consists of all base variables occurring in $C_2$ (not only of those contained in the constraint $\Lambda_2$), and $\vec{y}$ consists only of those $\Lambda_1\sigma$'s variables that do not appear in $C_2$. Defining the vectors as $\vec{x} = var(\Lambda_2)$ and $\vec{y} = var(\Lambda_1\sigma) \setminus var(\Lambda_2)$ would be wrong, because the clause $C_2$ may contain base variables which do not occur in its constraint $\Lambda_2$, but occur in $\Lambda_1\sigma$; quantifying such variables existentially may yield deleting non-redundant clauses. As an illustration, consider the following example.

EXAMPLE 3.95 ► Let the background theory T be rational linear arithmetic. Consider two abstracted clauses:

$$\begin{aligned} C_1 &= x + y > 0 \parallel \ \rightarrow P(x, y) \\ C_2 &= \quad\ \ x' > 0 \parallel \ \rightarrow P(x', y') \end{aligned}$$

It is easy to see that neither $C_1 \not\models_{\mathscr{C}} C_2$ nor $C_2 \not\models_{\mathscr{C}} C_1$. Nevertheless, for a free parts subsumption matcher $\sigma = [x \mapsto x', y \mapsto y']$, the following formula is LA-valid:

$$\forall x'.\exists y'.(x' > 0 \rightarrow x' + y' > 0),$$

whereas

$$\forall x', y'.(x' > 0 \rightarrow x' + y' > 0),$$

is LA-inconsistent. The former corresponds to the wrong quantification schema, according to which $\vec{x} = var(\Lambda_2) = \{x'\}$ and $\vec{y} = var(\Lambda_1\sigma) \setminus var(\Lambda_2) = \{y'\}$; the latter corresponds to the correct one, according to which $\vec{x} = var(C_2) = \{x', y'\}$ and $\vec{y} = var(\Lambda_1\sigma) \setminus var(C_2) = \emptyset$. ■

Next, we formally define the hierarchic reduction rules and prove that they agree with the hierarchic redundancy criterion.

DEFINITION 3.96 ► The ***Hierarchic Tautology Deletion*** rule is

Hierarchic Tautology Deletion

$$\mathcal{R}\frac{\Lambda \parallel \Gamma \rightarrow \Delta}{}$$

if

(i) $\models \Gamma \rightarrow \Delta$, or

(ii) $\exists \vec{x}. \bigwedge \Lambda \models_{\mathscr{C}} \bot$, for $\vec{x} = var(\Lambda)$.

■

Thus, the Hierarchic Tautology Deletion rule is applicable if at least one of the rule's conditions is fulfilled. The first condition requires the free part of the premise to be a tautology. According to the second condition the existential closure of the constraint of the premise has to be unsatisfiable in the base theory. Alternatively, the second condition can be equivalently reformulated as follows:

$$\begin{aligned} &\models_{\mathscr{C}} \forall \vec{x}. \neg \bigwedge \Lambda \\ &\models\!\!\!\mid \ \ \forall \vec{x}. \bigvee \overline{\Lambda} \end{aligned}$$

where $\vec{x} = var(\Lambda)$, and $\overline{\Lambda}$ the set of all $\Lambda$'s literals negated. That is, the rule applies if the negation $\neg \bigwedge \Lambda$ of the constraint $\Lambda$, or equivalently the base part[1] $\bigvee \overline{\Lambda}$ of the premise, is valid in the base theory.

◄ DEFINITION 3.97
Hierarchic Subsumption Deletion

The ***Hierarchic Subsumption Deletion*** rule is

$$\mathcal{R}\frac{\Lambda_1 \parallel \Gamma_1 \to \Delta_1 \qquad \Lambda_2 \parallel \Gamma_2 \to \Delta_2}{\Lambda_1 \parallel \Gamma_1 \to \Delta_1}$$

where, for a simple matcher $\sigma$,

(i) $\Gamma_1\sigma \subseteq \Gamma_2$, $\Delta_1\sigma \subseteq \Delta_2$,

(ii) $\models_{\mathscr{C}} \forall \vec{x}.\exists \vec{y}.(\bigwedge \Lambda_2 \to \bigwedge \Lambda_1\sigma)$, for $\vec{x} = var(C_2) \cap \mathcal{X}$ and $\vec{y} = var(\Lambda_1\sigma) \setminus var(C_2)$,

(iii) $(\Lambda_2 \parallel \Gamma_2 \to \Delta_2) \neq \square$.

■

The first two conditions of the Hierarchic Subsumption Deletion Rule are called respectively: ***free part subsumption condition*** and ***encompassment condition*** (as it requires for every variable assignment $v$ satisfying $\bigwedge \Lambda_2$ the existence of a variable assignment $v'$ that satisfies $\bigwedge \Lambda_1\sigma$, such that $v'(x) = v(x)$ for every $x \in var(\Lambda_2)$; thus, such $v'$ "*encompasses*" $v$ on all variables $var(\Lambda_2)$). The third condition of the Hierarchic Subsumption Deletion is called the ***non-emptiness condition*** and is stipulated by the fact, that the rule missing the condition would allow subsuming an empty clause by an unsatisfiable base clause. Indeed, the free parts of a base and an empty clause equal an empty set; the constraint of an empty clause is an empty set and is, thus, an empty conjunction, which equals $\top$ by construction; and, finally, the constraint of an unsatisfiable base clause is true in any base model. Consider, for instance, the hierarchic combination FOL(LA): the clause $(2 \cdot (10+1) \approx 22 \parallel \to)$ subsumes an empty clause $\square$, because $2 \cdot (10+1) \approx 22$ is true in any model of arithmetic, and $\models_{\mathscr{C}} (\top \to 2 \times (10+1) \approx 22)$ is thus true. The rule Hierarchic Subsumption Deletion can not be applied for the example due to condition (iii) of the rule. In practice, the condition does not make much sense as all/most theorem provers stop executing whenever an empty clause is derived; the condition is intended rather for theoretical purposes.

Note that $y \in var(\Lambda_1\sigma) \setminus var(C_2)$ implies $y \notin var(\Gamma_1, \Delta_1)$. If the base theory enables quantifier elimination, then every such variable $y$ could in fact be eliminated in $\Lambda_1$, and the encompassment condition would reduce to

$$\models_{\mathscr{C}} \forall \vec{x}.(\bigwedge \Lambda_2 \to \bigwedge \Lambda_1\sigma),$$

for $\vec{x} = var(C_2) \cap \mathcal{X} = var(C_1, C_2) \cap \mathcal{X}$. In Chapters 5 and 6, presenting the application of the SUP(T) calculus to reasoning in the hierarchic combinations where the background theory is linear and non-linear arithmetic, we will propose a sufficient criterion that decides the encompassment condition by determining the existence of a simple matcher $\tau$ which maps the variables $\vec{y}$ to base terms over variables of $\Lambda_2$ such that

$$\models_{\mathscr{C}} \forall \vec{x}.(\bigwedge \Lambda_2 \to \bigwedge \Lambda_1\sigma\tau),$$

holds. We call such $\tau$ an ***encompassment matcher***.

[1]Please, recall Definition 3.20 of an abstracted clause and, in particular, of its constituents.

LEMMA 3.98 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$. *Assume* $\succ$ *is a reduction ordering total on ground* $\Sigma'$*-terms which orients any non-base term greater than any base one.*

*The reduction rule Hierarchic Tautology Deletion enjoys the hierarchic redundancy criterion* $\mathcal{R}_F^{\mathcal{H}}$.

PROOF ► Let $N$ be a set of abstracted $\Sigma'$-clauses. Consider a Hierarchic Tautology Deletion reduction with a premise $C \in N$:

$$\mathcal{R}\frac{\Lambda \parallel \Gamma \to \Delta}{}$$

We are to show that $C \in \mathcal{R}_F^{\mathcal{H}}(N)$, so that deleting $C$ from $N$ is eligible. According to the definition of the rule, either

(i) $\models \Gamma \to \Delta$, or

(ii) $\exists \vec{x}. \bigwedge \Lambda \models_{\mathscr{C}} \bot$, for $\vec{x} = var(\Lambda)$,

holds.

If the first condition of the rule is satisfied, then we have:

$$\begin{aligned}
&\models \Gamma \to \Delta \\
&= \neg \bigwedge \Gamma \vee \bigvee \Delta \\
&\models \neg \bigwedge \Lambda \vee \neg \bigwedge \Gamma \vee \bigvee \Delta \\
&= \Lambda \parallel \Gamma \to \Delta \\
&= C
\end{aligned}$$

Therefore, the clause $C$ is a tautology, hence $sgi_{\mathcal{A}}(C) \subseteq \mathcal{R}_F^{\mathcal{F}}(N_{\mathcal{A}})$ for any base algebra $\mathcal{A} \in \mathscr{C}$, consequently $C \in \mathcal{R}_F^{\mathcal{H}}(N)$.

If Condition (ii) is fulfilled, then we have:

$$\begin{array}{lcl}
 & \exists \vec{x}. \bigwedge \Lambda \models_{\mathscr{C}} \bot & \text{// where } \vec{x} = var(\Lambda) \\
\Leftrightarrow & \models_{\mathscr{C}} \forall \vec{x}.(\neg \bigwedge \Lambda) & \text{// i.e. } \forall \vec{x}.(\neg \bigwedge \Lambda) \text{ a taut. for } \models_{\mathscr{C}}
\end{array}$$

Put $C' = \bigvee \overline{\Lambda}$ be a clause consisting of all $\Lambda$'s literals negated; $C' \models\!\!\!\dashv \neg \bigwedge \Lambda$. Then

$$\begin{array}{ll}
 & \models_{\mathscr{C}} C' \\
\Leftrightarrow & \forall \mathcal{A} \in \mathscr{C} : \mathcal{A} \models sgi(C')
\end{array}$$

Pick an arbitrary base algebra $\mathcal{A} \in \mathscr{C}$ and arbitrary clause $D \in sgi(C')$. From Lemma 3.51 it follows that there exist finitely many clauses $C_1, \ldots, C_n \in \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$, for some $n \geq 0$, such that $C_1, \ldots, C_n \models D$, and $C_i \preceq D$, for every $i \in \{1, \ldots, n\}$. Therefore

$$\begin{array}{ll}
 & D \in \mathcal{R}_F^{\mathcal{F}}(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \\
\Rightarrow & sgi(C') \subseteq \mathcal{R}_F^{\mathcal{F}}(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}
\end{array}$$

If $\Gamma \cup \Delta = \emptyset$, then $C = C'$ and

$$sgi(C) \subseteq \mathcal{R}_F^{\mathcal{F}}(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}.$$

Otherwise, if $\Gamma \cup \Delta \neq \emptyset$, then $C' \subset C$ and every clause $C\sigma \in sgi(C)$ is implied by the corresponding clause $C'\sigma \in sgi(C')$, where $\sigma$ is a simple ground substitution. Besides, $C'\sigma$ is smaller than $C\sigma$ with respect to $\succ$. This yields

$$sgi(C) \subseteq \mathcal{R}_F^{\mathcal{F}}(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}).$$

Combining the two above cases, we obtain:

$$\begin{array}{lll} & sgi(C) \subseteq \mathcal{R}_F^{\mathcal{F}}(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \\ \Rightarrow & sgi(C) \subseteq \mathcal{R}_F^{\mathcal{F}}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \text{// as } \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} \subseteq N_{\mathcal{A}}, \\ & & \text{and by Cond. (ii) of Def. 3.1} \\ & & \text{and Theorem 3.42} \\ \Rightarrow & sgi_{\mathcal{A}}(C) \subseteq \mathcal{R}_F^{\mathcal{F}}(N_{\mathcal{A}}) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} & \text{// as } sgi_{\mathcal{A}}(C) \subseteq sgi(C) \end{array}$$

As the base algebra $\mathcal{A} \in \mathscr{C}$ has been picked arbitrarily from $\mathscr{C}$, we conclude by Definition 3.64 of $\mathcal{R}_F^{\mathcal{H}}$ that $C \in \mathcal{R}_F^{\mathcal{H}}(N)$. ■

LEMMA 3.99 *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$. *Assume* $\succ$ *is a reduction ordering total on ground* $\Sigma'$*-terms which orients any non-base term greater than any base one.*

*Let* $N$ *and* $M$ *be two sets of abstracted clauses. If an abstracted clause* $C_2$ *is subsumed by an abstracted clause* $C_1 \in N$*, then any* $\mathcal{H}$*-inference deriving from* $M$ *the clause* $C_2$ *is redundant for* $N$*:*

$$\forall\, I \in \mathcal{H}(M) : concl(I) = C_2 \Rightarrow I \in \mathcal{R}_I^{\mathcal{H}}(N).$$

PROOF Consider a Hierarchic Subsumption Deletion reduction:

$$\mathcal{R}\frac{\Lambda_1 \parallel \Gamma_1 \to \Delta_1 \qquad \Lambda_2 \parallel \Gamma_2 \to \Delta_2}{\Lambda_1 \parallel \Gamma_1 \to \Delta_1}$$

with premises $C_1, C_2 \in N$, respectively. According to Definition 3.97 of the rule, the following conditions are satisfied:

(i) $\Gamma_1\sigma \subseteq \Gamma_2$, $\Delta_1\sigma \subseteq \Delta_2$.

(ii) $\models_{\mathscr{C}} \forall\, \vec{x}.\exists \vec{y}.(\bigwedge \Lambda_2 \to \bigwedge \Lambda_1\sigma)$, for $\vec{x} = var(C_2) \cap \mathcal{X}$ and $\vec{y} = var(\Lambda_1\sigma) \setminus var(C_2)$.

(iii) $(\Lambda_2 \parallel \Gamma_2 \to \Delta_2) \neq \square$.

Since all algebras in $\mathscr{C}$ are term-generated, condition (ii) holds if and only if for an arbitrary base algebra $\mathcal{A} \in \mathscr{C}$, and an arbitrary simple ground substitution $\phi$ with $dom(\phi) = var(C_2) \cap \mathcal{X}$, there exists a simple ground substitution $\delta$ with $dom(\delta) = var(\Lambda_1\sigma) \setminus var(C_2)$, such that:

$$\begin{array}{lll} & \mathcal{A} \models (\bigwedge \Lambda_2 \to \bigwedge \Lambda_1\sigma)\phi\delta & \\ \Leftrightarrow & \mathcal{A} \models \bigwedge \Lambda_2\phi\delta \to \bigwedge \Lambda_1\sigma\phi\delta & \\ \Leftrightarrow & \mathcal{A} \models \bigwedge \Lambda_2\phi \to \bigwedge \Lambda_1\sigma\phi\delta & \text{// as } \vec{y} \cap var(\Lambda_2) = \emptyset \\ \Leftrightarrow & \mathcal{A} \models \neg(\bigvee \overline{\Lambda}_1\sigma\phi\delta) \vee \bigvee \overline{\Lambda}_2\phi & \end{array}$$

Putting $C_1' = \bigvee \overline{\Lambda}_1\sigma$ and $C_2' = \bigvee \overline{\Lambda}_2$ (that is, letting the clauses $C_1'$ and $C_2'$ to consist of the literals in $\Lambda_1\sigma$ and $\Lambda_2$ negated, respectively), we obtain:

$$\begin{array}{lll} & \mathcal{A} \models (\neg C_1'\phi\delta \vee C_2'\phi) & \text{// where } C_1'\phi\delta \text{ and } C_2'\phi \text{ ground clauses} \\ \Leftrightarrow & \mathcal{A} \not\models C_1'\phi\delta \text{ or } \mathcal{A} \models C_2'\phi & \end{array}$$

By Corollary 3.52 we know $\mathcal{A} \not\models C_1'\phi\delta$ if and only if there exist finitely many clauses $D_1^1, \ldots, D_m^1 \in \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ such that

$$D_1^1, \ldots, D_m^1, C_1'\phi\delta \models \bot;$$

moreover, if a maximal literal in $C_1'\phi\delta$ is not an equation $s \approx t$, such that the clause $(s \approx t \to)$ is in $\mathsf{D}_{\mathcal{A}}$, then $D_i^1 \prec C_1'\phi\delta$, for every $i \in \{1, \ldots, m\}$. From Lemma 3.51 it follows that $\mathcal{A} \models C_2'\phi$ if and only if there exist finitely many clauses $D_1^2, \dots, D_n^2 \in \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ such that

$$D_1^2, \dots, D_n^2 \models C_2'\phi,$$

and $D_i^2 \preceq C_2'\phi$, for every $i \in \{1, \dots, n\}$. Putting

$$M_1 = \begin{cases} \{D_1^1, \dots, D_m^1\}, & \text{if } \mathcal{A} \not\models C_1'\phi\delta \\ \emptyset, & \text{oth.} \end{cases}$$
$$M_2 = \begin{cases} \{D_1^2, \dots, D_n^2\}, & \text{if } \mathcal{A} \models C_2'\phi \\ \emptyset, & \text{oth.} \end{cases}$$

we obtain:

$$\begin{array}{ll} & M_1, C_1'\phi\delta \models \bot \text{ or } M_2 \models C_2'\phi \\ \Rightarrow & M_1, M_2, C_1'\phi\delta \models C_2'\phi \\ \Leftrightarrow & M_1, M_2, \bigvee \overline{\Lambda}_1\sigma\phi\delta \models \bigvee \overline{\Lambda}_2\phi \\ \Leftrightarrow & M_1, M_2, \neg \bigwedge \Lambda_1\sigma\phi\delta \models \neg \bigwedge \Lambda_2\phi \\ \Rightarrow & M_1, M_2, (\neg \bigwedge \Lambda_1\sigma\phi\delta \vee \neg \bigwedge \Gamma_1\sigma\phi\phi' \vee \bigvee \Delta_1\sigma\phi\phi') \\ & \quad \models (\neg \bigwedge \Lambda_2\phi \vee \neg \bigwedge \Gamma_2\phi\phi' \vee \bigvee \Delta_2\phi\phi') \qquad \text{// from Cond. (i), for every} \\ & \qquad \text{// simple ground subst. } \phi' \\ & \qquad \text{// with } dom(\phi') = var(C_2) \cap \mathcal{X}'' \\ \Leftrightarrow & M_1, M_2, C_1\sigma\phi\delta\phi' \models C_2\phi\phi' \end{array}$$

Combined substitutions $\omega_1 = \sigma\phi\delta\phi$ and $\omega_2 = \phi\phi'$ are simple and grounding for the clauses $C_1$ and $C_2$, respectively. Let $\omega_1'$ be the $R_{\mathcal{A}}^{\approx}$-reduced $\omega_1$. By Proposition 3.59, there exist finitely many clauses $D_1^3, \dots, D_k^3 \in \mathsf{E}_{\mathcal{A}}$, for some $k \geq 0$, such that $D_1^3, \dots, D_k^3, C_1\omega_1' \models C_1\omega_1$ and $D_i^3 \preceq C_1\omega_1$, for every $i \in \{1, \dots, k\}$. Putting, $M_3 = \{D_1^3, \dots, D_k^3\}$, we have

$$M_1, M_2, M_3, C_1\omega_1' \models C_2\omega_2.$$

Since the base algebra $\mathcal{A}$ and the substitutions $\phi$ and $\phi'$ have been picked arbitrarily, we conclude that every simple ground $R_{\mathcal{A}}^{\approx}$-reduced instance of $C_2$ follows from a finite subset of $\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ and a simple ground $R_{\mathcal{A}}^{\approx}$-reduced instance of $C_1$, for any base algebra $\mathcal{A} \in \mathscr{C}$.

Let $I$ be an $\mathcal{H}$-inference deriving $C_2$, and $I' \in sgi_{\mathcal{A}}(I)$ an arbitrary simple ground $R_{\mathcal{A}}^{\approx}$-reduced instance of $I$. Without loss of generality, assume $I' = I\omega_2$. Let $C = max_{\succ}(prem(I'))$ be the maximal premise of $I'$. Since $\mathcal{H}$-inference rules admit only non-base clauses as premises, the premises of $I'$ are also non-base. Let us represent the clause $C$ as

$$C = \Pi, \Gamma \to \Upsilon, \Delta,$$

where $\Pi \to \Upsilon$ is the base part of the clause, and $\Gamma \to \Delta$ the non-base one. According to Proposition 3.44, the non-base part of $C_2\omega_2$ is smaller than the non-base part of $C$:

$$\begin{aligned} \Gamma \to \Delta &\succ (\Gamma_2 \to \Delta_2)\omega_2 \\ &\supseteq (\Gamma_1 \to \Delta_1)\sigma\omega_1 \\ &\succeq (\Gamma_1 \to \Delta_1)\omega_1' \end{aligned}$$

Consequently,

$$\begin{array}{ll} & \Gamma \to \Delta \succ (\Lambda_1 \parallel \Gamma_1 \to \Delta_1)\omega_1' \\ \Rightarrow & \Pi, \Gamma \to \Upsilon, \Delta \succ (\Lambda_1 \parallel \Gamma_1 \to \Delta_1)\omega_1' \end{array}$$

Thus, $C \succ C_1\omega_1'$. Evidently, $C$ is also greater than any clause $D \in M_1 \cup M_2 \cup M_3$. Thus, the conclusion $C_2\omega_2$ of the inference $I'$ logically follows from finitely many clauses, each of which is smaller than the maximal premise $C$ of the inference. Since $M_1 \cup M_2 \cup M_3 \cup C_1\omega_1' \subseteq N_{\mathcal{A}}$, we conclude by Definition 3.40 of $\mathcal{F}$-redundant inferences that the inference $I'$ is redundant for $N_{\mathcal{A}}$:

$$I' \in \mathcal{R}_I^{\mathcal{F}}(N_{\mathcal{A}}).$$

Since the inference $I'$ and the base algebra $\mathcal{A} \in \mathscr{C}$ have been picked arbitrarily, it follows that all simple ground $R_{\mathcal{A}}^{\approx}$-reduced instances of $I$ are redundant, and, therefore, by Definition 3.64 of $\mathcal{H}$-redundant inferences, $I$ is redundant for $N$:

$$I \in \mathcal{R}_I^{\mathcal{H}}(N),$$

and the proof is complete. ■

CHAPTER 4

# SUP(T) as a Decision Procedure

In this chapter we show that the SUP(T) calculus is a decision procedure for the ground FOL(T) fragment as well as a non-ground FOL(T) fragment where non-base variables can only be instantiated by constants. This answers the so far open question whether SUP(T) can actually decide the ground case. It was already known that SUP(T) is complete and for some cases even a decision procedure for certain non-ground fragments such as the combination of FOL and (non)linear arithmetic [AKW09a, EKS$^{+}$11], and the analysis of fragments resulting from the translation of first-order probabilistic and (extended) timed automata [FHW10, FKW12b, FKW12a, FW11]. In Section 4.1 we make an introduction into specifics of SUP(T) application to ground problems. In Section 4.2, we prove SUP(T) to be a decision procedure for the ground FOL(T) fragment. In Section 4.3 we show that SUP(T) can decide beyond the ground fragment. The paper ends with a presentation of an example application of the achieved results for reasoning in ontologies with arithmetical facts, Section 4.4.
This chapter is an extended version of [KW11, KW12].

## 4.1 Introduction

The hierarchic superposition calculus SUP(T) over a theory T enables sound reasoning on the hierarchic combination FOL(T) of a theory T with full first-order logic. If a FOL(T) clause set enjoys a sufficient completeness criterion, the calculus is even complete. Clause sets over the ground fragment of FOL(T) are not sufficiently complete, in general. In this chapter we show that any clause set over the ground FOL(T) fragment can be transformed into a sufficiently complete one, and prove that SUP(T) terminates on the transformed clause set, hence constitutes a decision procedure provided the existential fragment of the theory T is decidable. Thanks to the hierarchic design of SUP(T), the decidability result can be extended beyond the ground case. We show SUP(T) is a decision procedure for the non-ground FOL fragment plus a theory T, if every non-constant function symbol from the underlying FOL signature ranges into the sort of the theory T, and every term of the theory sort is ground. This actually closes a gap for SUP(T) that was already shown a decision procedure for certain FOL(T) fragments with variables [FW11, FKW12b], whereas the ground case has not been yet solved.

The given definition of the sufficient completeness criterion, Definition 3.78, is very strong, but still the calculus may be complete for some clause sets, that do not enjoy the criterion. In practice, we only need to sufficiently define only those non-base terms of the base sort that actually occur in a given clause set $N$. For a finite clause set $N$ sufficient completeness can be obtained by transforming $N$ into a clause set $N'$, in which every non-base term $t$ of the base sort occurs in a positive unit clause of the form $t \approx t'$, where $t'$ is a base term. If by a proper instantiation of the SUP(T) calculus the existence of such equations remains an invariant for any SUP(T) derivation $N_0 \vdash N_1 \vdash \ldots$, where $N_0$ is the clause set $N'$ abstracted, sufficient completeness is preserved and the overall approach is complete. Completeness of the calculus for clause sets over the fragments we consider in the current work is achieved exactly in this way – by sufficiently defining all non-base terms of the base sort that do occur in a given clause set (the approach is further described in the following sections).

EXAMPLE 4.1 ► Here is an example, for which the hierarchical calculus is not complete, but where completeness can be recovered by sufficiently defining present terms. Let the background theory T be rational linear arithmetic; $<, \geq, 0 \in \Omega$, and $f, a \in \Omega''$; the function symbol $f$ ranges into the base sort $\mathsf{S}$, and $a$ into the free sort $\mathsf{S}''$. Consider two clauses

$$\begin{array}{ll}(1) & \rightarrow f(a) \geq 0 \\ (2) & \rightarrow f(a) < 0\end{array}$$

which are abstracted to

$$\begin{array}{ll}(1') & x \not\geq 0 \parallel f(a) \approx x \rightarrow \\ (2') & x \not< 0 \parallel f(a) \approx x \rightarrow\end{array}$$

Recall that variables in clauses are universally quantified, hence all clauses are variable disjoint. Clearly, the set of these two clauses is unsatisfiable relative to $\mathscr{C}$, however no SUP(T) inference is possible. Note that hierarchic equality resolution is not applicable as $\sigma = \{x \mapsto f(a)\}$ is not a simple substitution. The two clauses are not sufficiently complete, because the clause set does not imply simple instances of the term $f(a)$ to be equal to some base term.

Note that the set is satisfiable in a model $\mathcal{A}'$, that extends the base sort with an extra "junk" element, which we denote by "$\diamondsuit$", exclusively generated by the term $f(a)$, i.e. $t_{\mathcal{A}'} = \diamondsuit$ iff $t = f(a)$; and interprets the predicates "$<$" and "$\geq$" such that:

- $\diamondsuit \geq_{\mathcal{A}'} 0_{\mathcal{A}'}$ and $\diamondsuit <_{\mathcal{A}'} 0_{\mathcal{A}'}$ are true; and
- $\mathcal{A}' \models s < s'$ iff $s < s'$ holds in the theory of linear arithmetic (analogously for $\geq$), for any ground base terms $s, s' T_\Omega$.

The model $\mathcal{A}'$ is not hierarchic, as its restriction $\mathcal{A}'|_\Sigma$ to the base signature is a non-standard model of linear arithmetic (because of the junk "$\diamondsuit$").

The equality of $f(a)$ to a base term can, for example, be forced by the additional clause

$$(3) \qquad \rightarrow f(a) \approx 0$$

which is abstracted to

$$(3') \qquad y \approx 0 \parallel \rightarrow f(a) \approx y$$

Now the three clauses are sufficiently complete. For the clauses there is a refutation where Hierarchic Superposition Left Inferences between the clauses (1) and (3), and (2) and (3) yield respectively (note the move of base variable assignments $y \approx x$ from the free parts of the conclusions to the constraints):

$$\begin{array}{ll} (4) & x < 0, y \approx 0, y \approx x \parallel \rightarrow \\ (5) & x \geq 0, y \approx 0, y \approx x \parallel \rightarrow \end{array}$$

which are equivalent to

$$\begin{array}{ll} (4') & 0 < 0 \parallel \rightarrow \\ (5') & 0 \geq 0 \parallel \rightarrow \end{array}$$

respectively. A Constraint Refutation application to the clause (4) (or equivalently to (4′)) results in the empty clause $\square$. ■

## 4.2 Deciding Ground FOL(T)

In this section we prove that the SUP(T) calculus is a decision procedure for the ground FOL(T) fragment.

### 4.2.1 Basification

A key idea of the approach to decide the ground FOL(T) fragment we describe here is to *sufficiently define* ground base sort terms, occurring in an input clause set and whose top symbol is a free function symbol, by extending the clause set by extra positive unit clauses consisting of an equation between every such a term and a fresh (i.e. not occurring in the clause set) base constant (parameter) uniquely introduced for the term, thus obtaining a sufficiently complete clause set which is satisfiable relative to T iff the original one is so. Here we assume that the background theory T provides such free (Skolem) constants. Semantically, they play the role of existentially quantified variables, i.e., for our decision result to be effective the base theory must provide decidability for the existential fragment. This is, for example, the case for the theory of linear arithmetic, considered as the running example. Furthermore, extending the background theory with new constant symbols may cause losing compactness of the overall approach. We show that following a certain strategy, the SUP(T) calculus may produce only *finitely many* irredundant clauses in the fragment considered, so compactness is not needed any more.

Given a set $N$ of $\Sigma'$-clauses, an extension term $t \in T^{E}_{\Omega'}(\mathcal{X}')$ is called *basified*, if the set $N$ contains a positive unit clause $C \in N$ consisting of an equation $t \approx \mathsf{a}$, where $\mathsf{a}$ is a base constant (theory parameter). The clause $C$ is called *basifying* (*for the term t*). A clause set $N$ is called *basified* if every extension term $t$, occurring in $N$, is basified by a clause in $N$.

Every set $N$ of $\Sigma'$-clauses can be transformed into an equisatisfiable basified clause set in the following way: traverse every literal in every clause in $N$ *from innermost* (i.e. discover the literal's term tree bottom up) and search for occurrences of ground extension subterms; whenever such a term $t$ is being observed, there are two possible ways of proceeding:

– if this is the *first occurrence* of $t$ considered, then: (i) introduce a fresh base constant $\mathsf{a} : \rightarrow \mathsf{S}$ of the same base sort $sort(t) = \mathsf{S} \in \mathcal{S}$, (ii) replace the occurrence of $t$ with $\mathsf{a}$, and (iii) extend the input clause set $N$ with a basifying clause $C = (\rightarrow t \approx \mathsf{a})$;

– for otherwise, replace the current occurrence of $t$ with the parameter $\mathsf{a}$, that has been introduced previously up to this point, when the first occurrence of $t$ has been discovered.

As the transformation of terms is performed from innermost to outwards, in every term $t$ basified, there is no subterm whose top symbol is an extension symbol, thus no further basification in the new basifying clauses is needed. Besides, all subsequent occurrences of the same term $t$ appearing in the input clause set $N$ are replaced with the same base parameter $\mathsf{a}$ exclusively introduced for $t$. If a clause set $N$ is basified, then every ground extension term $t$ occurring in $N$ is sufficiently defined via a basifying clause $(\rightarrow t \approx \mathsf{a}) \in N$. From a semantics perspective, the base parameters can be thought of as base-sorted variables existentially quantified on the top of the overall problem.

◀ EXAMPLE 4.2

Let $N$ be a given set of ground clauses to be basified, containing just one clause:

$$N = \{\, P(f(h(1+g(a)))) \;\rightarrow\; f(h(1)) + g(a) \geq 0 \,\}$$

where the background theory T is rational linear arithmetic; $+, \geq, 0, 1 \in \Omega$, and $P, f, g, a, h \in \Omega''$; the function symbols $f, g$ range into a base sort $\mathsf{S} \in \mathcal{S}$, and $a, h$ into a free sort $\mathsf{S}'' \in \mathcal{S}''$.

Assume the literal $\neg P(f(h(1+g(a))))$ is discovered first. Its (only) innermost extension subterm is $g(a)$. First, a fresh base parameter $\mathsf{a}$ is introduced; second, the current occurrence of $g(a)$ is replaced with $\mathsf{a}$; and then a basifying clause $(\rightarrow g(a) \approx \mathsf{a})$ is added, yielding the following clause set:

$$\begin{aligned} \{\, P(f(h(1+\mathsf{a}))) \;&\rightarrow\; f(h(1)) + g(a) \geq 0, \\ &\rightarrow\; g(a) \approx \mathsf{a} \,\} \end{aligned}$$

Next, the basification procedure processes the subterm $f(h(1+\mathsf{a}))$ by, likewise, introducing a fresh parameter, replacing the subterm with the new parameter, and adding the respective basifying clause, resulting in the following clause set:

$$\begin{aligned} \{\, P(\mathsf{b}) \;&\rightarrow\; f(h(1)) + g(a) \geq 0, \\ &\rightarrow\; g(a) \approx \mathsf{a}, \\ &\rightarrow\; f(h(1+\mathsf{a})) \approx \mathsf{b} \,\} \end{aligned}$$

At this point, every subterm of the first literal becomes basified, so the second literal $f(h(1)) + g(a) \geq 0$ is being processed. Assume the subterm $f(h(1))$ is considered first. After basifying the subterm in the above manner, we obtain:

$$\begin{aligned} \{\, P(\mathsf{b}) \;&\rightarrow\; \mathsf{c} + g(a) \geq 0, \\ &\rightarrow\; g(a) \approx \mathsf{a}, \\ &\rightarrow\; f(h(1+\mathsf{a})) \approx \mathsf{b}, \\ &\rightarrow\; f(h(1)) \approx \mathsf{c} \,\} \end{aligned}$$

In the next step, the subterm $g(a)$ is to be basified. Since another occurrence of the term has been already processed, and for which a unique base parameter has been introduced, the parameter $\mathsf{a}$, no new parameter is produced now and no basifying clause is added, and the current occurrence of $g(a)$ is replaced with the $\mathsf{a}$, yielding the following clause set $N'$:

$$\begin{aligned} N' = \{\, P(\mathsf{b}) \;&\rightarrow\; \mathsf{c} + \mathsf{a} \geq 0, \\ &\rightarrow\; g(a) \approx \mathsf{a}, \\ &\rightarrow\; f(h(1+\mathsf{a})) \approx \mathsf{b}, \\ &\rightarrow\; f(h(1)) \approx \mathsf{c} \,\} \end{aligned}$$

The obtained clause set $N'$ can be split into two parts: $N_I$ and $N_B$. The set $N_I$ is the set of the basified clauses from the initially given clause set $N$, the set $N_B$ is the set of clauses basifying those in $N_I$. So, for the example above, we obtain:

$$\begin{aligned} N_I = \{\, P(\mathsf{b}) \;&\rightarrow\; \mathsf{c} + \mathsf{a} \geq 0 \,\}, \\ N_B = \{\, \quad\;\; &\rightarrow\; g(a) \approx \mathsf{a}, \\ &\rightarrow\; f(h(1+\mathsf{a})) \approx \mathsf{b}, \\ &\rightarrow\; f(h(1)) \approx \mathsf{c} \,\} \end{aligned}$$

■

From Example 4.2, one can see that basification of a ground clause set $N$ results in a set $N' = N_I \cup N_B$, where $N_I$ is the set of the basified clauses from $N$, and $N_B$ is the set of clauses basifying those in $N_I$, such that every base sort term in $N_I$ is a ground base term possibly containing new parameters, and every base sort term in $N_B$ is either a ground smooth extension term, or a ground base term possibly containing new parameters. In fact, this observation is true for any set of clauses, in which all terms of a base sort are ground, because basification applies only to *extension* terms that are *ground*. We formalize this property in the following proposition.

PROPOSITION 4.3 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*Let $N$ be a set of clauses in which all base sort terms are ground. Let $N' = N_I \cup N_B$ be a clause set obtained from $N$ by basification, where $N_I$ is the set of the basified clauses from $N$, and $N_B$ is the set of clauses basifying those in $N_I$. Every base sort term occurring in a clause from $N_I$ is base and ground. The only literal $L = t \approx \mathsf{a}$ in any clause from $N_B$ is an equation between a base parameter* $\mathsf{a}$ *and a ground smooth extension term $t \in T^E_{\Omega'}$.*

PROOF ► Consider an arbitrary clause $( \quad \| \rightarrow t \approx \mathsf{a}) \in N_B$. The clause is basifying for some ground extension term $t$ occurring in a clause from $N$. Since basification runs from innermost in a backward depth-first-search manner, at the iteration of basification, when the first occurrence of $t$ is basified, all strict subterms of $t$ have been basified already and replaced thus by base parameters. Therefore, $t$ is a ground smooth extension term, implying that any *strict* base sort subterm of $t$ is base and ground.

Let $L = t_1 \mathrel{\dot{\approx}} t_2$ be an arbitrary literal from a clause $C \in N$, where $\dot{\approx} \in \{\approx, \not\approx\}$. We want to determine the structure of the literal $L' = t_1' \mathrel{\dot{\approx}} t_2'$, which the literal $L$ becomes after basification in the corresponding clause $C' \in N_I$. Regarding the structure of $t_1$ – analogously for $t_2$ – there are two possible cases:

1. no subterm of $t_1$ is an extension term, implying that any base sort subterm of $t_1$ is actually base and ground; in this case, there is no subterm in $t_1$ to be basified, therefore $t_1$ remains unchanged: $t_1' = t_1$.

2. $t_1$ contains an extension subterm $s$; in this case, every such a subterm $s$ is ground and, hence, replaced with a base parameter, yielding a term $t_1'$ with no extension subterm, implying that any base sort subterm of $t_1'$ is base and ground.

■

COROLLARY 4.4 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*Let $N$ be a set of clauses in which all base sort terms are ground. Let $N'$ be a clause set obtained from $N$ by basification, and $L = t_1 \mathrel{\dot{\approx}} t_2$ a literal in a clause from*

*$N'$, where $\dot{\approx} \in \{\approx, \not\approx\}$. Every subterm of each $t_i$ with a base top operator symbol is a ground base term:*

$$\forall\, p \in \rho(t_i) : top(t_i/p) \in \Omega \Rightarrow t_i/p \in T_\Omega,$$

*for every $i \in \{1,2\}$.*

◄ PROOF Immediately follows from Proposition 4.3. ■

For a given clause set $N$, basification of $N$ produces an equisatisfiable clause set $N'$; moreover, every model of $N'$ is a model of $N$ (but not the other way round, in general). Moreover, if all base sort terms occurring in clauses from $N$ are ground, then from Proposition 4.3 it follows that all extension terms occurring in $N'$ are sufficiently defined and occur only in positive literals of the form $t \approx \mathsf{a}$, where $\mathsf{a}$ is a base parameter. On other hand, basification does not allow to sufficiently define all possible extension terms over the signature of $N$, in general. Thus, $N'$ may still miss sufficient completeness. In Section 4.2.3, we will discuss this issue in detail; particularly, we will show that defining only those extension terms that occur in $N'$ suffices to guarantee refutational completeness for the *fragments considered.*

◄ LEMMA 4.5 *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $N$ be a set of clauses, and $N'$ be obtained from $N$ by basification, then:*

*(i)* *$N'$ and $N$ equisatisfiable;*

*(ii)* $N' \models N$.

The set $N'$ contains one extra clause for each unique ground extension term appearing in $N$, and its size asymptotically equals the size of $N$ in the number of symbols in $N$. Basification takes loglinear (i.e. $n \log n$) time in the number $n$ of symbols in $N$.

### 4.2.2 Derivation Invariants

Abstraction introduced in Section 3.2 is used to split a clause into base and free parts such that each part consists of base and free literals, respectively, and the only symbols shared by the parts of an abstracted clause are base variables. After a given clause set $N$ has been basified into a clause set $N' = N_I \cup N_B$, the set $N'$ is to be abstracted. In this section we investigate structural invariants of constraints and free literals in clauses derived by the SUP(T) calculus from a clause set obtained by basification and abstraction from a set $N$ of clauses in which all base sort terms are ground. Considering such sets $N$ (instead of just ground FOL(T) clause sets) allows us to directly extend obtained results onto the non-ground fragment studied in Section 4.3.

◄ EXAMPLE 4.6 Consider the basified clause set $N' = N_I \cup N_B$ from the previous example:

$$\begin{aligned} N_I = \{\, P(\mathsf{b}) &\rightarrow \mathsf{c} + \mathsf{a} \geq 0\,\}, \\ N_B = \{\, &\rightarrow g(a) \approx \mathsf{a}, \\ &\rightarrow f(h(1 + \mathsf{a})) \approx \mathsf{b}, \\ &\rightarrow f(h(1)) \approx \mathsf{c}\,\} \end{aligned}$$

Abstraction yields the following clause set (recall that all variables are universally quantified):

$$\begin{aligned}
N'_I &= \{\; x \approx \mathsf{b}, \mathsf{c}+\mathsf{a} < 0 \parallel P(x) \;\rightarrow\; \},\\
N'_B &= \{\; x \approx \mathsf{a} \parallel \;\rightarrow\; g(a) \approx x,\\
&\quad x \approx \mathsf{b}, y \approx 1+\mathsf{a} \parallel \;\rightarrow\; f(h(y)) \approx x,\\
&\quad x \approx \mathsf{c}, y \approx 1 \parallel \;\rightarrow\; f(h(y)) \approx x\,\}
\end{aligned}$$

■

**Constraints and Base Variables**

According to Corollary 4.4, any term $t$, occurring in a clause $C = \Gamma \rightarrow \Delta$ from the clause set $N'$ and whose top symbol is a base operator symbol, is a base term (meaning that $t$ contains no free symbol), therefore no free operator symbol may appear in $C$ below a base one, hence only base terms are abstracted out. Consequently, every base variable $x$ in the abstracted clause $C' = \Lambda' \parallel \Gamma' \rightarrow \Delta'$ is introduced during abstraction and assigned to some *ground* base term $t$ via an equation $x \approx t$ in $\Lambda'$. Thus, abstraction populates the constraint $\Lambda'$ of the clause $C'$ by literals of two types:

- assignments $x \approx t$ of a base variable $x$ to a ground base term $t$, arising as the result of abstracting out base subterms occurring in $C$ immediately below a free operator symbol; and
- base literals $s \mathrel{\dot{\approx}} t$, where $s$ and $t$ are ground base terms and $\dot{\approx} \in \{\approx, \not\approx\}$, which have been already present in the clause $C \in N'$ prior to abstraction and moved then to the constraint of $C'$ at the final step of abstraction[1]; the literal gets negated when moved to the constraint if it comes from the succedent of $C$.

Since no free term is abstracted out, the number of literals in each subset of the free part of the abstracted clause $C' = \Gamma' \rightarrow \Delta'$ may only decrease in comparison to that of the corresponding original clause $C = \Lambda \parallel \Gamma \rightarrow \Delta$, i.e. $|\Gamma'| \leq |\Gamma|$ and $|\Delta'| \leq |\Delta|$.

Next, we show that the above properties regarding base variables and the structure of the clauses' constraints are actually invariants for any SUP(T) derivation $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$, where $N_0$ is obtained by basification and abstraction from $N$. The discovered properties will become particularly important when proving termination of SUP(T), Section 4.2.4.

PROPOSITION 4.7 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

[1]Recall that predicates are encoded as functions; for instance, the literal $\mathsf{c}+\mathsf{a} < 0$ from the example above is an abbreviation for the equation $<(\mathsf{c}+\mathsf{a}, 0) \approx true_<$. Encoding of predicates as functions is needed to compactly define the calculus. According to the definition of the SUP(T) calculus, only free literals are superposed, therefore encoding base predicates as functions is not strictly necessary. Still, we do it as it helps to compress the theoretical discussion by saving one extra case (of considering predicative atoms), like in Proposition 4.11. Whenever the Constraint Refutation rule is performed and the background T-solver is invoked, every such base predicative (dis)equation $P(t_1, \ldots, t_n) \mathrel{\dot{\approx}} true_P$, where $\dot{\approx} \in \{\approx, \not\approx\}$, is passed to the solver as a corresponding predicate $P(t_1, \ldots, t_n)$ or $\neg P(t_1, \ldots, t_n)$, respectively.

*Let $N''$ be obtained by basification and abstraction from a set $N$ of $\Sigma'$-clauses in which all terms of a base sort are ground. Let $C = \Lambda \parallel \Gamma \to \Delta$ be an arbitrary clause in $N''$. For every base variable $x \in var(C) \cap \mathcal{X}$ there exists a ground base term $t \in T_\Omega$ such that*[1] *the constraint $\Lambda$ of $C$ contains an equation $x \approx t$.*

◄ PROOF

Without loss of generality, assume $N' = N_I \cup N_B$ is obtained from $N$ by basification, and $N'' = N'_I \cup N'_B$ obtained from $N'$ by abstraction, where $N'_I$ and $N'_B$ correspond to $N_I$ and $N_B$, respectively.

Consider an arbitrary clause $D \in N_I$ and an arbitrary literal $L = (t_1 \mathrel{\dot{\approx}} t_2) \in D$, where $t_1$ and $t_2$ are two terms and $\dot{\approx} \in \{\approx, \not\approx\}$. Let $C \in N'_I$ be the clause resulting from abstraction of $D$. By Corollary 4.4, any subterm of each $t_i$ with a base top operator symbol is a ground base term. Moreover, according to Proposition 4.3, $L$ contains no extension subterm. Hence, either $t_1$ and $t_2$ are ground base terms, or they are non-base terms of a free sort with no occurrence of a base variable. In the first case, the literal $L$ is moved by abstraction to the constraint of $C$ (where it gets negated if it appears positively in $D$). For the second case, assume $s^i_1, \ldots, s^i_{m_i}$ are all the occurrences of base terms appearing in each $t_i$ *immediately* below a free operator symbol, for each $i \in \{1,2\}$ and some $m_i \geq 0$; we denote this fact by $t_i[s^i_1, \ldots, s^i_{m_i}]$ (if a base term $s$ occurs in $t_i$ more than once, say $k > 1$ times, then it appears among $s^i_1, \ldots, s^i_{m_i}$ also $k$ times). Abstraction of $t_1 \approx t_2$

- replaces $s^i_1, \ldots, s^i_{m_i}$ with fresh base variables $x^i_1, \ldots, x^i_{m_i}$, respectively, yielding a literal $t'_1 \approx t'_2$, where $t'_i = t[x^i_1, \ldots, x^i_{m_i}]$, for each $i \in \{1,2\}$; and
- populates the constraint of $C$ with assignments $x^i_1 \approx s^i_1$, …, $x^i_{m_i} \approx s^i_{m_i}$, for each $i \in \{1,2\}$.

Thus, the set of all base variables occurring in the abstracted literal $L' = t'_1 \approx t'_2$ consists of the fresh variables $x^i_j$, each of which is assigned to a ground base term $s^i_j$ via an equation $x^i_j \approx s^i_j$ in the constraint of $C$, for all $i \in \{1,2\}$ and $j \in \{1, \ldots, m_i\}$. Applying the above argument to all free literals in $C$, the assertion follows for all base variables occurring in the free part of $C$. Since all base variables are introduced during abstraction and contained in free literals, the assertion follows for all base variables in $C$.

Consider an arbitrary clause $D \in N_B$. Let $C \in N'_B$ be the clause resulting from abstraction of $D$. According to Proposition 4.3, $D$ is a positive unit clause ($\to t \approx \mathsf{a}$), where $\mathsf{a}$ is a base parameter and $t$ a ground smooth extension term. According to Corollary 4.4, all $t$'s subterms with a base top operator symbol are base and ground. Assume $s_1, \ldots, s_m$ are all the occurrences of base terms appearing in $t$ immediately below a free operator symbol, for some $m \geq 0$; we denote this fact by $t[s_1, \ldots, s_m]$ (if a base term $s$ occurs in $t$ more than once, say $k > 1$ times, then it appears among $s_1, \ldots, s_m$ also $k$ times). Abstraction of $t \approx \mathsf{a}$

- replaces $\mathsf{a}$ and $s_1, \ldots, s_m$ with fresh base variables $x$, $y_1, \ldots, y_m$, respectively, yielding a literal $t' \approx x$, where $t' = t[x_1, \ldots, x_m]$; and
- populates the constraint with assignments $x \approx \mathsf{a}$, and $x_1 \approx s_1$, …, $x_m \approx s_m$.

[1]Recall that $T_\Omega$ stands for the set of all ground base terms. Notation for different types of term sets is introduced in Section 2.2, page 14, and Section 3.2 (Definition 3.16), page 39.

Thus, basification of $D$ results in a clause

$$C = x \approx \mathsf{a}, x_1 \approx s_1, \ldots, x_m \approx s_m \parallel \rightarrow t' \approx x,$$

in which every base variable is assigned to a ground base term via an equation in the constraint of $C$. ■

PROPOSITION 4.8 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$. *Let* $\succ$ *be a reduction ordering total on ground* $\Sigma'$*-terms.*

*Let* $N''$ *be obtained by basification and abstraction from a set* $N$ *of* $\Sigma'$*-clauses in which all terms of a base sort are ground. The clause set* $N''$ *is locally sufficiently complete*[1].

PROOF ► Without loss of generality, assume $N' = N_I \cup N_B$ is obtained from $N$ by basification, where $N_I$ is the set of the basified clauses from $N$, and $N_B$ is the set of clauses basifying those in $N_I$, and $N'' = N'_I \cup N'_B$ obtained from $N'$ by abstraction, where $N'_I$ and $N'_B$ correspond to $N_I$ and $N_B$, respectively.

Consider an arbitrary smooth ground instance $C \in smgi(N'')$ of a clause $C' \in N''$, such that $C = C'\sigma$, for some smooth ground substitution $\sigma$. The clause $C$ contains an extension term if and only if the clause $C'$ does, as any term in the image of any smooth ground substitution is extension-free. In the proof of Proposition 4.7 we have shown that all extension terms appearing in $N''$ occur in clauses $N'_B$. Hence, $C' \in N'_B$. In the proof of Proposition 4.7 we have also shown and any $C' \in N'_B$ has the following form:

$$C' = (x_0 \approx \mathsf{a}, x_1 \approx t_1, \ldots, x_n \approx t_n \parallel \rightarrow t[x_1, \ldots x_n] \approx x_0)$$

where:

- $\Lambda = (x_0 \approx \mathsf{a}, \ldots, x_n \approx t_n)$ is the constraint of the clause,
- $t \in T^E_{\Omega'}(\mathcal{X})$ a smooth extension $\Sigma'$-term, whose variables $var\, t = \{x_1, \ldots, x_n\}$ are all base, if any,
- $x_0$ a base variable,
- each $t_i$ a ground base term.

Let $s_0, \ldots, s_n \in T_\Omega$ be the base ground terms such that $x_i\sigma = s_i$, for each $i \in \{0, \ldots, n\}$. Put $t_0 = \mathsf{a}$. Then the clause $C = C'\sigma$ has the following form:

$$\begin{aligned} C &= (s_0 \approx t_0, \ldots, s_n \approx t_n \parallel \rightarrow t[s_1, \ldots, s_n] \approx s_0) \\ &= (s_0 \not\approx t_0 \vee \ldots \vee s_n \not\approx t_n \vee t[s_1, \ldots, s_n] \approx s_0) \end{aligned}$$

Let $\mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}}$ be an arbitrary weak algebra, and $\mathcal{A} \in \mathscr{C}$ an arbitrary base algebra monomorphic[2] to $\mathcal{A}'$. Assume $\mathcal{A}' \models sgi(N'')$. Consequently, $\mathcal{A}' \models C \in smgi(N'') \subseteq sgi(N'')$. Consider two possible cases:

1. if $\mathcal{A}' \models s_i \approx t_i$ for each $i \in \{0, \ldots, n\}$, then $\mathcal{A}' \models C$ if and only if $\mathcal{A}' \models t[s_1, \ldots, s_n] \approx s_0$.

[1]Please recall Definition 3.84 of a locally sufficiently complete clause set, page 102.

[2]The existence of such base algebra $\mathcal{A}$ is guaranteed, because, by definition, for any weak algebra $\mathcal{A}'$ there is a base algebra $\mathcal{A}$ which is monomorphic to $\mathcal{A}'$; see Definition 3.73 of a weak algebra, page 89.

2. otherwise, $\mathcal{A}' \not\models s_j \approx t_j$ for some $j \in \{0,\dots,n\}$. Put $D = (s_j \approx t_j \rightarrow)$. Evidently, $D$ is a ground base clause and $\mathcal{A}' \models D$. Consequently, $\mathcal{A} \models D$ (as, for otherwise, $\mathcal{A} \models s_j \approx t_j$; Proposition 3.51 and Theorem 3.76 imply $\mathcal{A}' \models s_j \approx t_j$, a contradiction). According to Lemma 3.51, $\mathcal{A} \models D$ if and only if there exist finitely many clauses $C_1,\dots,C_m \in \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$ such that $C_k \preceq D$, for every $k \in \{1,\dots,m\}$, and $C_1,\dots,C_m \models D$. Consequently, $C_k \prec C\sigma$, for every $k \in \{1,\dots,m\}$, and $C_1,\dots,C_m \models C\sigma$, hence $C \in \mathcal{R}^{\mathcal{F}}_F(\mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}) \subseteq \mathcal{R}^{\mathcal{F}}_F(smgi(N'') \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}})$, by Definition 3.40 of the flat redundancy criterion $\mathcal{R}^{\mathcal{F}}$.

Since $C \in smgi(N'')$, $\mathcal{A}' \in \mathscr{W}_{\mathsf{HSp}}$, and $\mathcal{A} \in \mathscr{C}$ have been picked arbitrarily, we conclude that $N''$ is locally sufficiently complete. ∎

◄ PROPOSITION 4.9

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

*Let* $N_0 \vdash N_1 \vdash N_2 \vdash \dots$ *be a SUP(T) derivation,* $N_0$ *is obtained by basification and abstraction from a set* $N$ *of* $\Sigma'$*-clauses in which all terms of a base sort are ground; let* $C = \Lambda \parallel \Gamma \rightarrow \Delta$ *be an arbitrary clause in* $N_i$, $i \geq 0$. *Then for every base variable* $x \in var(C) \cap \mathcal{X}$ *there exists a ground base term* $t \in T_\Omega$ *such that an equation* $x \approx t$ *is in the constraint* $\Lambda$ *of* $C$.

◄ PROOF

We give a proof by induction on the length of derivation of $N_i$.

*Induction Base.* For $N_0$ the assertion holds by Proposition 4.7.

*Induction Hypothesis.* Assume the assertion holds for all clause sets $N_i$ derived from $N_0$, where $i \leq n$, for some $n > 0$.

*Induction Step.* Let $N_{n+1} = N_n \cup C$, where $C$ is a conclusion of an inference rule application to some clauses in $N_n$. Consider the Hierarchic Equality Resolution rule, Definition 3.29,

$$\mathcal{I}\frac{\Lambda \parallel \Gamma, s \approx t \rightarrow \Delta}{(\Lambda \parallel \Gamma \rightarrow \Delta)\sigma}$$

with the premise $C_1 \in N_n$ and conclusion $C$, respectively. As $\sigma$ is simple and the premise is abstracted, the substitution $\sigma$ can unify a base variable only with another base variable. The variables $var(C_1) \setminus dom(\sigma)$ are not affected by $\sigma$, and $cdom(\sigma) \subseteq var(s,t) \subseteq var(C_1)$. Therefore, every base variable $x \in var(C) \cap \mathcal{X}$ comes from the clause $C_1$, and $x = y\sigma$, where $y$ is a base variable (not necessarily different from $x$) from $var(C_1) \cap \mathcal{X}$. By induction hypothesis, every base variable $y \in var(C_1) \cap \mathcal{X}$ in $C_1$ is assigned to some ground base term $t' \in T_\Omega$ via an equation $(y \approx t') \in \Lambda$ in the constraint $\Lambda$ of $C_1$, hence $(y \approx t')\sigma = (x \approx t'\sigma) \in \Lambda\sigma$, where $\Lambda\sigma$ is the constraint of $C$. Since $t'$ is ground, we obtain $(x \approx t') \in \Lambda\sigma$.

Consider the inference rule Hierarchic Superposition Right, Definition 3.28,

$$\mathcal{I}\frac{\Lambda_1 \parallel \Gamma_1 \rightarrow \Delta_1, l \approx r \qquad \Lambda_2 \parallel \Gamma_2 \rightarrow \Delta_2, s[l'] \approx t}{(\Lambda_1, \Lambda_2 \parallel \Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2, s[r] \approx t)\sigma}$$

with the premises $C_1, C_2 \in N_n$ and conclusion $C$, respectively. Analogously to the case of Hierarchic Equality Resolution, we learn that every base variable $x \in var(C) \cap \mathcal{X}$ comes from a clause $C_1$ or $C_2$, and $x = y\sigma$, where $y$ is a base variable (not necessarily different from $x$) from $var(C_1, C_2) \cap \mathcal{X}$. Likewise, we conclude that an equation $(x \approx t')$ is contained in the constraint $(\Lambda_1, \Lambda_2)\sigma$ of $C$, for some ground base term $t' \in T_\Omega$. Analysis of the Hierarchic Superposition Left rule is similar.

Consider the Hierarchic Splitting rule, Definition 3.26,

$$\mathcal{S}\frac{\Lambda_1, \Lambda_2 \parallel \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2}{\Lambda_1 \parallel \Gamma_1 \to \Delta_1 \quad \Big| \quad \Lambda_2 \parallel \Gamma_2 \to \Delta_2}$$

with the premise $C \in N_n$ and conclusions $C_1$ and $C_2$, respectively. By induction hypothesis, we know that for every base variable $x \in var(C) \cap \mathcal{X}$ the equation $(x \approx t) \in (\Lambda_1, \Lambda_2)$, for some $t \in T_\Omega$. As, by Definition 3.26 of the rule, $\Lambda_1, \Gamma_1, \Delta_1$ on one hand, and $\Lambda_2, \Gamma_2, \Delta_2$, on the other, are variable-disjoint, it holds that for every base variable $x \in var(C_i) \cap \mathcal{X}$, there exists a term $t \in T_\Omega$, such that the atom $(x \approx t) \in \Lambda_i$, for each $i \in \{1,2\}$.

The case of the Constraint Refutation rule is trivial as the rule's conclusion is an empty clause. An application of a reduction rule Hierarchic Tautology Deletion or Hierarchic Subsumption Deletion yields a clause set $N_{n+1}$ which is a subset of $N_n$, hence the induction step trivially holds for the reductions. ■

An application of the Hierarchic Superposition Left/Right rule can produce an assignment of two variables, say $(x \approx y)$. If the variables are base, we replace $(x \approx y)$ with $(s \approx t)$, where $s$ and $t$ are the ground base terms, which the variables are respectively assigned to (Proposition 4.9), and move the equation $(s \approx t)$ to the constraint of the conclusion, where it gets negated in the case of applying the Hierarchic Superposition Right rule. We call this transformation the *variables assignments grounding.*

DEFINITION 4.10 ► Variables Assignments Grounding

Assume HSp = (Sp, Sp′) is a hierarchic specification with the base specification Sp = $(\Sigma, \mathscr{C})$ and the body Sp′ = $(\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

Let $C = \Lambda \parallel \Gamma \to \Delta$ be an abstracted clause, $(x \approx y) \in \Gamma \cup \Delta$ an equation in the free part of the clause between two base variables $x, y \in var(\Gamma, \Delta) \cap \mathcal{X}$, that are assigned to some ground base terms $s, t \in T_\Omega$ via equations $(x \approx s)$, $(y \approx t) \in \Lambda$ in the constraint of the clause. Deletion of every such an equation $x \approx y$ from the free part $\Gamma \to \Delta$ of the clause $C$ and adding a ground equation $s \approx t$ if $(x \approx y) \in \Gamma$, or a ground disequation $s \not\approx t$ if $(x \approx y) \in \Delta$, to the constraint $\Lambda$ is called ***variables assignments grounding***. We write $VAG(C)$ to denote the result of variables assignments grounding applied to the clause $C$. ■

Obviously, $VAG(C)$ is equivalent to $C$. The variables assignments grounding is implicitly applied to every derived clause.

PROPOSITION 4.11 ► *Assume* HSp = (Sp, Sp′) *is a hierarchic specification with the base specification* Sp = $(\Sigma, \mathscr{C})$ *and the body* Sp′ = $(\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

*Let* $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ *be a SUP(T) derivation, where* $N_0$ *is obtained by basification and abstraction from a set* $N$ *of* $\Sigma'$*-clauses in which all terms of a base sort are ground; let* $C = \Lambda \parallel \Gamma \to \Delta$ *be an arbitrary clause in* $N_i$, $i \geq 0$*. If variables assignments grounding is applied to every clause derived, then every literal in* $\Lambda$ *is either*

– *an assignment $x \approx t$, or*

– *a (dis)equation $s_1 \mathrel{\dot{\approx}} s_2$,*

*where $x \in \mathcal{X}$ is a base variable, $t, s_1, s_2 \in T_\Omega$ ground base terms, $\dot{\approx} \in \{\approx, \not\approx\}$.*

We give a proof by induction on the length of derivation of $N_i$. ◄ PROOF

*Induction Base.* For $N_0$ a proof is the same as that of Proposition 4.7.

*Induction Hypothesis.* Assume the assertion holds for all clause sets $N_i$ derived from $N_0$, where $i \le n$, for some $n > 0$.

*Induction Step.* Let $N_{n+1} = N_n \cup C$, where $C$ is a conclusion of an inference rule application to some clauses in $N_n$. Consider a Hierarchic Superposition Left inference, Definition 3.27,

$$\mathcal{I}\frac{\Lambda_1 \parallel \Gamma_1 \to \Delta_1, l \approx r \qquad \Lambda_2 \parallel s[l'] \approx t, \Gamma_2 \to \Delta_2}{(\Lambda_1, \Lambda_2 \parallel s[r] \approx t, \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2)\sigma}$$

with the premises $C_1, C_2 \in N_n$ and conclusion $C$, respectively. The constraint $\Lambda = (\Lambda_1, \Lambda_2)\sigma$ of $C$ is the conjunction of the premises' constraints with the substitution $\sigma$ applied onto them, therefore every literal $L' \in \Lambda$ equals $L\sigma$, for some $L \in \Lambda_1 \cup \Lambda_2$. As $\sigma$ is simple and the premises are abstracted, the substitution $\sigma$ can unify a base variable only with another base variable. By induction hypothesis, every literal $L \in \Lambda_1 \cup \Lambda_2$ is either:

– an assignment $x \approx t$, then $L\sigma = (x \approx t)\sigma = (y \approx t)$, where $y = x\sigma \in \mathcal{X}$ and $t \in T_\Omega$; or

– a (dis)equation $s_1 \approx s_2$, then $L\sigma = (s_1 \approx s_2)\sigma = (s_1 \approx s_2)$, where $s_1, s_2 \in T_\Omega$.

If the rule produces an assignment $x \approx y$ of two base variables in the free part of the conclusion, it is moved to the constraint and grounded by the variables assignments grounding transformation, so that it becomes $s_1 \approx s_2$, for some ground base terms $s_1, s_2 \in T_\Omega$.

Analysis of the other inference rules (except Constraint Refutation and Hierarchic Splitting) is similar. The Hierarchic Splitting rule produces two clauses, whose constraints are subsets of the premise's constraint, which enjoys the stated property by induction hypothesis. The induction step trivially holds for an application of the Constraint Refutation inference rule. Also, the induction step trivially holds for an application of the reduction rules Hierarchic Tautology Deletion and Hierarchic Subsumption Deletion. ■

**Free Literals**

Next, we study the structure of free literals in clauses derived from a clause set obtained by basification and abstraction from a set clauses in which all terms of a base sort are ground. The main property of free literals in clauses in such a derivation is that an extension symbol $f \in \Omega''$, $sort(f) \in \mathcal{S}$ may occur only in positive literals of the form $t \approx x$, where $x \in \mathcal{X}$ is a base variable, and the top symbol of $t$ is the only occurrence of $f$ in $t$. This property will be required for showing model existence for a saturated clause set that does not contain an empty clause, Section 4.2.3.

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} =$ ◄ PROPOSITION 4.12

*$(\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively.*

*Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ be a SUP(T) derivation, where $N_0$ is obtained by basification and abstraction from a set $N$ of $\Sigma'$-clauses in which all terms of a base sort are ground; let $C$ be an arbitrary clause in the derivation. All extension terms occurring in $C$, if any, are smooth and appear only in literals of the form $t \approx x$, where $t \in T^E_{\Omega'}(\mathcal{X}')$ is an extension term and $x \in \mathcal{X}$ a base variable.*

PROOF ► We give a proof by induction on the length of the derivation.

*Induction Base.* Without loss of generality, assume $N' = N_I \cup N_B$ is obtained from $N$ by basification, and $N_0 = N'_I \cup N'_B$ obtained from $N'$ by abstraction, where $N'_I$ and $N'_B$ correspond to $N_I$ and $N_B$, respectively.

According to Proposition 4.3, no clause in $N_I$ contains an extension term, hence the statement trivially holds for clauses in $N'_i$.

Consider an arbitrary clause $D \in N_B$. Let $C \in N'_B$ be the clause resulting from abstraction of $D$. According to Proposition 4.3, $D$ is a positive ground unit clause $(\rightarrow t \approx \mathsf{a})$, where $\mathsf{a}$ is a base parameter and $t = f(t'_1, \ldots, t'_n)$ a ground smooth extension term. According to Corollary 4.4, all $t$'s subterms with a base top operator symbol are base. Assume $s_1, \ldots, s_m$ are all the occurrences of base terms appearing in $t$ immediately below a free operator symbol, for some $m \geq 0$; we denote this fact by $t[s_1, \ldots, s_m]$ (if a base term $s$ occurs in $t$ more than once, say $k > 1$ times, then it appears among $s_1, \ldots, s_m$ also $k$ times). Abstraction of $t \approx \mathsf{a}$

- replaces $\mathsf{a}$ and $s_1, \ldots, s_m$ with fresh base variables $x$, $y_1, \ldots, y_m$, respectively, yielding a literal $t' \approx x$, where $t' = t[x_1, \ldots, x_m] = f(t''_1, \ldots, t''_n)$; and
- populates the constraint with assignments $x \approx \mathsf{a}$, and $x_1 \approx s_1$, …, $x_m \approx s_m$.

Thus, basification of $D$ results in a clause

$$C = x \approx \mathsf{a}, x_1 \approx s_1, \ldots, x_m \approx s_m \parallel \rightarrow f(t''_1, \ldots, t''_n) \approx x,$$

where $f(t''_1, \ldots, t''_n) \in T^E_{\Omega'}(\mathcal{X})$ is a smooth extension term, and $x \in \mathcal{X}$ a base variable.

*Induction Hypothesis.* Assume the assertion holds for every clause in all sets $N_i$ derived from $N_0$, where $i \leq n$, for some $n > 0$.

*Induction Step.* Let $N_{n+1} = N_n \cup C$, where $C$ is a conclusion of an inference rule application to some clauses in $N_n$. We consider the inference rules Hierarchic Equality Resolution and Hierarchic Superposition Right in detail; analysis of the other rules is similar. But first, we show an important structural property of the image $im(\sigma)$ of a unifier $\sigma$ which takes part in an *arbitrary* inference on $N_n$.

According to definitions of the SUP(T) inference rules, the unifier $\sigma$ is simple, hence every base variable in the domain of $\sigma$ is mapped to a base term, actually, to a base variable as the clauses are abstracted:

$$\forall x \in dom(\sigma) \cap \mathcal{X} : x\sigma \in \mathcal{X}.$$

By induction hypothesis, an extension term may appear in the premises from $N_n$ only below the equality symbol $\approx$ (therefore, an extension term does not appear below a symbol ranging into a free sort), consequently no term a non-base variable is mapped to has an extension subterm. Thus, no term in the image of $\sigma$ contains an extension subterm:

$$\forall t \in im(\sigma). \forall p \in \rho(t) : t/p \notin T^E_{\Omega'}(\mathcal{X}').$$

Consequently, any term obtained by an application of $\sigma$ contains only those occurrences of extension symbols that are present in the term the substitution is applied to:

$$\forall t' \in T_{\Omega'}(\mathcal{X}'), \forall p \in \rho(t'\sigma) : t'\sigma/p \in T^E_{\Omega'}(\mathcal{X}') \Rightarrow top(t'\sigma/p) = top(t'/p).$$

Consider a Hierarchic Equality Resolution inference, Definition 3.29,

$$\mathcal{I}\frac{\Lambda \parallel \Gamma, s \approx t \rightarrow \Delta}{(\Lambda \parallel \Gamma \rightarrow \Delta)\sigma}$$

with the premise $C_1 \in N_n$ and conclusion $C$, respectively. Consider an arbitrary atom $A\sigma \in \Gamma\sigma \cup \Delta\sigma$. If the corresponding atom $A \in \Gamma \cup \Delta$ does not contain an occurrence of an extension term, then, by the observation made about $\sigma$, neither does $A\sigma$. If $A$ contains an extension term, then, by induction hypothesis, $A \in \Delta$ and $A = (t \approx x)$, where $t \in T^E_{\Omega'}(\mathcal{X}')$ is a smooth extension term and $x \in \mathcal{X}$ a base variable. By the observation made about $\sigma$, we conclude $t\sigma$ is again a smooth extension term, and $x\sigma$ a base variable.

Consider the inference rule Hierarchic Superposition Right, Definition 3.28,

$$\mathcal{I}\frac{\Lambda_1 \parallel \Gamma_1 \rightarrow \Delta_1, l \approx r \qquad \Lambda_2 \parallel \Gamma_2 \rightarrow \Delta_2, s[l'] \approx t}{(\Lambda_1, \Lambda_2 \parallel \Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2, s[r] \approx t)\sigma}$$

with the premises $C_1, C_2 \in N_n$ and conclusion $C$, respectively.

Consider an arbitrary atom $A\sigma \in \Gamma_1\sigma \cup \Gamma_2\sigma \cup \Delta_1\sigma \cup \Delta_2\sigma$. Similarly to the case of the Hierarchic Equality Resolution inference, we conclude that $A\sigma$ contains an extension term, if and only if $A\sigma$ is an equation from $\Delta_1\sigma \cup \Delta_2\sigma$ between a smooth extension term and a base variable.

Now, we are to determine the structure of the literal $(s[r] \approx t)\sigma$. Consider the term $l$ from the premise $C_1$. There are the following two possible cases regarding occurrences of extension symbols in $l \approx r$:

– In $l \approx r$ there is an occurrence of an extension term. By induction hypothesis, one of the term $l$ or $r$ is a smooth extension term, another a base variable. According to condition (iii) of the rule's definition, $l\sigma\psi \succ r\sigma\psi$, for a simple grounding substitution $\psi$. The combined substitution $\sigma\psi$ is simple and grounding, hence an application of $\sigma\psi$ onto a base variable in $dom(\sigma\psi)$ results in a ground base term. Since all ground base terms are smaller than any non-base term, we conclude that $r$ is a base variable, say $x \in \mathcal{X}$, and $l \in T^E_{\Omega'}(\mathcal{X}')$ is a smooth extension term. From induction hypothesis, it also follows that $s$ is a smooth extension term, and $t$ a base variable, say $y \in \mathcal{X}$. Thus, we have

$$\begin{aligned} &(s[r] \approx t)\sigma \\ &\quad (x \approx y)\sigma \\ &\quad x' \approx y', \end{aligned}$$

for some base variables $x', y' \in \mathcal{X}$. Since $x' \approx y'$ is a base literal it is moved negated to the constraint of the conclusion.

– Otherwise, from induction hypothesis it follows that $l$ and $r$ and all their subterms are non-extension terms. Consider two subcases:

  • if in $s \approx t$ there is an occurrence of an extension term, then we likewise conclude that $s \in T^E_{\Omega'}(\mathcal{X}')$ is a smooth extension term, and $t$ a base variable, say $t = x$, for some $x \in \mathcal{X}$. Since $l$, and hence $l'$, is of a free sort, $l'$ is a strict subterm of $s$. The term $s[r]$ obtained from $s$ by replacing $l'$ with $r$, is also a smooth extension term. From the observation made about $\sigma$, we conclude that $s[r] \in T^E_{\Omega'}$ is a smooth extension term, and $t\sigma = x\sigma \in \mathcal{X}$ a base variable.

- otherwise, from induction hypothesis it follows that $s$ and $t$ contain no extension subterm. Consequently, by the observation made about $\sigma$, the terms $s[r]\sigma$ and $t\sigma$ contain no extension subterm as well.

Thus, every free literal in the conclusion $C$ either has no occurrence of an extension term, or is an equation between a smooth extension term and a base variable.

Consider the Hierarchic Splitting rule, Definition 3.26,

$$\mathcal{S}\ \frac{\Lambda_1, \Lambda_2 \parallel \Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2}{\Lambda_1 \parallel \Gamma_1 \to \Delta_1 \quad \Big| \quad \Lambda_2 \parallel \Gamma_2 \to \Delta_2}$$

with the premise $C \in N_n$ and conclusions $C_1$ and $C_2$, respectively. The assertion evidently holds for the conclusions, because they are subsets of the premise, which satisfies the assertion by the induction hypothesis.

The case of the Constraint Refutation rule is trivial as the rule's conclusion is an empty clause. An application of a reduction rule Hierarchic Tautology Deletion or Hierarchic Subsumption Deletion yields a clause set $N_{n+1}$ which is a subset of $N_n$, hence the induction step trivially holds for the reduction rules. ■

### 4.2.3 Model Existence

In Sections 4.1 and 4.2.1 we have shown that ground clause sets are not sufficiently complete, in general. Basification resolves this problem by sufficiently defining terms which appear in the original set, turning thus the original clause set to a *locally sufficiently complete* one. According to Proposition 4.8, local sufficient completeness is preserved during abstraction. A locally sufficiently complete set $N$ of abstracted clauses has a hierarchic model if the limit $N_\infty$ of a fair SUP(T) derivation from $N$ does not contain an empty clause and the set of all base clauses in $N_\infty$ is theory-consistent, Theorem 3.93. In Section 4.3 right after the proof of Theorem 4.54, we will show that the base subset of the limit $N_\infty$ of a fair SUP(T)-derivation $N_0 \vdash N_1 \vdash \ldots$, where $N_0 = N$ is obtained by basification and abstraction from a set of $\Sigma'$-clauses in which all base sort terms are ground, is theory consistent whenever $\square \not\in N_\infty$, yielding by Theorem 3.93 the existence of a hierarchic model of $N$.

Here we present another technique of finding a hierarchic model. The below approach differs from the one demonstrated in Section 3.5 in various aspects, particularly in the structure of the constructed hierarchic (Herbrand) model and the requirements imposed on the underlying ordering $\succ$ (here $\succ$ is not obliged to orient all extension terms greater than any other term containing no extension subterm).

**Overview.** Let $M = N_\infty$ be the limit of a fair SUP(T)-derivation $N_0 \vdash N_1 \vdash \ldots$, where $N_0$ is obtained by basification and abstraction from a set of ground $\Sigma'$-clauses. Since $N_0 \vdash N_1 \vdash \ldots$ is fair, it follows that $M$ is saturated, by Lemma 3.9. Assume $M$ does not contain an empty clause $\square$. According to Theorem 3.72, the ground clause set $M_\mathcal{A} = sgi_\mathcal{A}(M) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$ is saturated with respect to the standard SUP calculus for ground clauses $(\mathcal{F}, \mathcal{R}^\mathcal{F})$, where $\mathcal{A} \in \mathscr{C}$ is a base algebra satisfying $sgi(M) \cap Cl_\Sigma$, the base subset of $sgi(M)$. By Lemma 3.41, $M_\mathcal{A}$ has a Herbrand model $\mathcal{I}_{M_\mathcal{A}} = T_{\Omega'}/R_{M_\mathcal{A}}$, where $R_{M_\mathcal{A}}$ is a rewrite system constructed from the maximal literals of productive clauses in $M_\mathcal{A}$, and $T_{\Omega'}/R_{M_\mathcal{A}}$ the quotient for ground $\Sigma'$-terms $T_{\Omega'}$ by the smallest congruence relation containing $R_{M_\mathcal{A}}$ (see Definition 2.29 for further details regarding construction of $\mathcal{I}_{M_\mathcal{A}}$ and $R_{M_\mathcal{A}}$). In general, the Herbrand interpretation $\mathcal{I}_{M_\mathcal{A}}$ is not a hierarchic algebra, as it may allow "junks" into base sorts (that is, elements of the base part of the universe, that no base terms are equal to). We tackle this issue by extending the rewrite system $R_{M_\mathcal{A}}$ with additional rewrite rules $R^{SD}_{M_\mathcal{A}}$ to a rewrite system $R'_{M_\mathcal{A}} = R_{M_\mathcal{A}} \cup R^{SD}_{M_\mathcal{A}}$. The main purpose of rewrite rules in $R^{SD}_{M_\mathcal{A}}$ is to sufficiently define those terms which are not sufficiently defined by $R_{M_\mathcal{A}}$ by equating such terms to (arbitrary) base terms. Our goal is to construct $R^{SD}_{M_\mathcal{A}}$ in a safe way such that the corresponding Herbrand interpretation $\mathcal{I}'_{M_\mathcal{A}} = T_{\Omega'}/R'_{M_\mathcal{A}}$ is a hierarchic model of $M_\mathcal{A}$, hence a hierarchic model of the original set of clauses.

The basic idea behind the construction of the rewrite system $R^{SD}_{M_\mathcal{A}}$ is to assign all smooth extension terms $t'$, which are not sufficiently defined under $R_{M_\mathcal{A}}$, to arbitrary base terms $t$ (for two different extension terms $t'_1$ and $t'_2$, the corresponding base terms $t_1$ and $t_2$ do not have to be necessarily different). Besides, we want the resulting rewrite system $R'_{M_\mathcal{A}}$ to be *consistent with the clause set* $M_\mathcal{A}$ and *convergent*.

◄ DEFINITION 4.13
Rewrite system $R^{SD}_N$

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively.

Let $N$ be a set of ground $\Sigma'$-clauses. Assume $\mathcal{I}_N = T_{\Omega'}/R_N$ is a candidate Herbrand interpretation[1] for $N$. We define a ground rewrite system $R^{SD}_N$ as the set of all rewrite rules with unique left-hand-sides, from every ground smooth extension term $l \in T^E_{\Omega'}$, that is not equal under $\mathcal{I}_N$ to any base term and reduced with respect to the rewrite system $R_N$, to an arbitrary ground base term $r \in T_\Omega$:

$$(l \to r) \in R^{SD}_N \overset{\text{def}}{\iff} l \in T^E_{\Omega'} \text{ and } r \in T_\Omega \text{ such that:}$$

- $\forall (l' \to r') \in R^{SD}_N : l = l' \Rightarrow r = r'$,
- $\forall t \in T_\Omega : \mathcal{I}_N \not\models l \approx t$,
- $l$ smooth, and
- $l = l{\downarrow}_{R_N}$

■

◄ PROPOSITION 4.14

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

[1]Construction of $R_N$ and $\mathcal{I}_N$ is given in Definition 2.29, page 26.

*Let $N$ be an arbitrary set of ground $\Sigma'$-clauses. In any model of $R'_N = R_N \cup R^{SD}_N$, any ground smooth extension term is equal to some base term:*

$$\forall\, t \in T^E_{\Omega'}.\exists\, s \in T_\Omega :\ t \text{ smooth } \Rightarrow R'_N \models t \approx s.$$

PROOF ► Follows from Definition 4.13 of $R^{SD}_N$ and Birkhoff's Theorem (Theorem 2.28). ■

From the Extension Terms Lemma (Lemma 3.19) it follows that all models of $R'_N = R_N \cup R^{SD}_N$ interpret any ground non-base term of a base sort the same as some base term:

$$\forall\, t \in T_{\Omega'}(\mathcal{S}) \setminus T_\Omega.\exists\, s \in T_\Omega : R'_N \models t \approx s.$$

Let $M = N_\infty$ be the limit of a fair SUP(T)-derivation $N_0 \vdash N_1 \vdash \ldots$, where $N_0$ is obtained by basification and abstraction from a set of ground $\Sigma'$-clauses, and $\mathcal{A} \in \mathscr{C}$ a base algebra. In order to study properties of the rewrite systems $R^{SD}_{M_\mathcal{A}}$ and $R'_{M_\mathcal{A}}$, we need to determine the structure of equations (rewrite rules) in $R_{M_\mathcal{A}}$, which are, by construction, the maximal literals of productive clauses in $M_\mathcal{A} = sgi_\mathcal{A}(M) \cup \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A}$.

PROPOSITION 4.15 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*Let $N_0 \vdash N_1 \vdash \ldots$ a SUP(T)-derivation, where $N_0$ is obtained by basification and abstraction from a set of ground $\Sigma'$-clauses, and $C$ an arbitrary clause in $sgi(N_i)$, for some $i \geq 0$. All extension terms occurring in $C$, if any, are smooth and appear only in literals of form $t \approx s$, where $t \in T^E_{\Omega'}$ is an extension term and $s \in T_\Omega$ a ground base term.*

PROOF ► Follows from Proposition 4.12 and the facts that (i) every $N_i$ in the derivation may contain only base variables, and (ii) simple substitutions map base variables only to base terms. ■

PROPOSITION 4.16 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $M = N_\infty$ be the limit of a fair SUP(T)-derivation $N_0 \vdash N_1 \vdash \ldots$, where $N_0$ is obtained from a set of ground $\Sigma'$-clauses by basification and abstraction. Let $\mathcal{A} \in \mathscr{C}$ an arbitrary base algebra. The rewrite system $R'_{M_\mathcal{A}} = R_{M_\mathcal{A}} \cup R^{SD}_{M_\mathcal{A}}$ is convergent.*

PROOF ► According to Theorem 2.21, an arbitrary rewrite system $R$ is convergent if it is terminating and left-reduced. According to Theorem 2.20, $R$ is terminating iff the left-hand-side of every rule in $R$ is greater than its right-hand-side with respect to some reduction ordering.

From Definition 2.29 we know, that $l \succ r$ for every rule $(l \to r) \in R_{M_\mathcal{A}}$. According to Definition 4.13, the left-hand-side $l$ of any rule $(l \to r) \in R^{SD}_{M_\mathcal{A}}$ is a ground extension term, hence non-base, and the left-hand-side $r$ a ground base term. In any reduction ordering admissible for SUP(T), any ground non-base term is greater than any ground base one, therefore $l \succ r$, for every $(l \to r) \in R^{SD}_{M_\mathcal{A}}$. Hence, $R'_{M_\mathcal{A}}$ is terminating.

The rewrite systems $R_{M_\mathcal{A}}$ and $R^{SD}_{M_\mathcal{A}}$ are left-reduced apart: $R_{M_\mathcal{A}}$ is left-reduced by construction, $R^{SD}_{M_\mathcal{A}}$ is left-reduced, because the left-hand-side of any rule in it is a unique *smooth* extension term. We are to show that the union of the rewrite systems is left-reduced as well. Consider two arbitrary rules $(l \to r) \in R_{M_\mathcal{A}}$ and $(l' \to r') \in R^{SD}_{M_\mathcal{A}}$:

- According to Definition 4.13 of $R^{SD}_{M_\mathcal{A}}$, the left-hand-side $l'$ is in its normal form with respect to $R_{M_\mathcal{A}}$: $l' = l'{\downarrow_{R_{M_\mathcal{A}}}}$. Therefore, no rewriting from $l'$ by $l \to r$ is possible.
- A rewriting from $l$ by $l' \to r'$ is impossible at the top position, because, otherwise, $l'$ could be rewritten, in turn, by $l \to r$.
- By Definition 2.29 of $R_{M_\mathcal{A}}$, the rule $l \to r$ is actually an equation (positive literal) from a clause from $\mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A} \cup sgi_\mathcal{A}(M) = M_\mathcal{A}$. Obviously, $l \to r$ does not come from $\mathsf{D}_\mathcal{A}$. If $l \to r$ comes from $\mathsf{E}_\mathcal{A}$, then $l$ and $r$ are ground base terms. If $l \to r$ comes from $sgi_\mathcal{A}(M) \subseteq sgi(M)$, then, according to Proposition 4.15, if $l$ contains an extension subterm, then $l$ is actually a smooth extension term. In both cases, no strict subterm of $l$ is an extension term, and, hence, no rewriting from $l$ by $l' \to r'$ is possible below the top position.

Since the rules $(l \to r)$ and $(l' \to r')$ have been selected arbitrarily, we conclude that the left-hand-sides of all rules in $R_{M_\mathcal{A}}$ are in their normal forms with respect $R^{SD}_{M_\mathcal{A}}$, and vice versa. Consequently, $R'_{M_\mathcal{A}}$ is left-reduced. ■

◄ THEOREM 4.17

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*Let* $M = N_\infty$ *equal the limit of a fair SUP(T)-derivation* $N_0 \vdash N_1 \vdash \ldots$, *where* $N_0$ *is obtained from a set of ground* $\Sigma'$*-clauses by basification and abstraction. Assume there exists a base algebra* $\mathcal{A} \in \mathscr{C}$ *satisfying the set* $M \cap Cl_\Sigma$ *of all base clauses within* $M$. *If* $M$ *does not contain an empty clause* $\square$, *then* $M$ *has a hierarchic model.*

◄ PROOF

Since $M$ is the limit of a fair derivation $N_0 \vdash N_1 \vdash \ldots$, the clause set $M$ is saturated, Lemma 3.9. By the Hierarchic Saturation Theorem (Theorem 3.72), the clause set $M_\mathcal{A} = \mathsf{E}_\mathcal{A} \cup \mathsf{D}_\mathcal{A} \cup sgi_\mathcal{A}(M)$ is saturated with respect to the flat superposition calculus $(\mathcal{F}, \mathcal{R}^\mathcal{F})$. From $\square \notin M$ it follows that $\square \notin M_\mathcal{A}$. According to Lemma 3.41, $M_\mathcal{A}$ has a Herbrand model $\mathcal{I}_{M_\mathcal{A}} = T_{\Omega'}/R_{M_\mathcal{A}}$. We are to show that a Herbrand interpretation $\mathcal{I}'_{M_\mathcal{A}} = T_{\Omega'}/R'_{M_\mathcal{A}}$ is a hierarchic model of $M$, where $R'_{M_\mathcal{A}} = R_{M_\mathcal{A}} \cup R^{SD}_{M_\mathcal{A}}$.

The rest of the proof is split into three parts, in which we show that: first, $\mathcal{I}'_{M_\mathcal{A}}$ is a model of the ground clause set $M_\mathcal{A}$; second, $\mathcal{I}'_{M_\mathcal{A}}$ is a model of $M$; and, third, $\mathcal{I}'_{M_\mathcal{A}}$ is hierarchic algebra.

**Subgoal $\mathcal{I}'_{M_\mathcal{A}} \models M_\mathcal{A}$.** Let us, first, make the following two observations regarding rewritings by $R_{M_\mathcal{A}}$. In any SUP(T)-admissible ordering $\succ$, any ground non-base term is greater than any ground base one, hence a ground base term can be rewritten in $R_{M_\mathcal{A}}$ only to another ground base term. Moreover, according to Proposition 4.15 all maximal literals in productive clauses in $sgi_\mathcal{A}(M) \subseteq sgi(M)$, which contain an extension subterm, are equations between an extension term and a ground base term. Clauses in $\mathsf{E}_\mathcal{A}$ may produce only equations between two ground base terms, and no clause in $\mathsf{D}_\mathcal{A}$ is productive, hence a non-extension term can be rewritten in $R_{M_\mathcal{A}}$ only to another non-extension term.

Consider an arbitrary clause $C \in M_{\mathcal{A}}$. The clause $C$ is entailed by $\mathcal{I}_{M_{\mathcal{A}}}$ if and only if $\mathcal{I}_{M_{\mathcal{A}}}$ satisfies some literal $L = (s \mathrel{\dot{\approx}} t) \in C$, where $\dot{\approx} \in \{\approx, \not\approx\}$. Consider two possible case regarding the sign of $L$:

– $L$ is positive, i.e. $L = s \approx t$, then

$$\begin{array}{cll} & \mathcal{I}_{M_{\mathcal{A}}} \models s \approx t & \\ \Leftrightarrow & s \leftrightarrow^*_{R_{M_{\mathcal{A}}}} t & \text{// by Birkhoff's Theorem (Thm. 2.28)} \\ \Rightarrow & s \leftrightarrow^*_{R'_{M_{\mathcal{A}}}} t & \text{// as } R_{M_{\mathcal{A}}} \subseteq R'_{M_{\mathcal{A}}} \\ \Leftrightarrow & \mathcal{I}'_{M_{\mathcal{A}}} \models s \approx t & \text{// by Birkhoff's Theorem,} \\ & & \text{// where } \mathcal{I}'_{M_{\mathcal{A}}} = T_{\Omega'}/R'_{M_{\mathcal{A}}} \end{array}$$

– $L$ is negative, i.e. $L = s \not\approx t$, then

$$\begin{array}{cll} & \mathcal{I}_{M_{\mathcal{A}}} \models s \not\approx t & \\ \Leftrightarrow & \mathcal{I}_{M_{\mathcal{A}}} \not\models s \approx t & \\ \Leftrightarrow & s \not\leftrightarrow^*_{R_{M_{\mathcal{A}}}} t & \text{// by Birkhoff's Theorem} \\ \Leftrightarrow & s{\downarrow}_{R_{M_{\mathcal{A}}}} \neq t{\downarrow}_{R_{M_{\mathcal{A}}}} & \text{// as } R_{M_{\mathcal{A}}} \text{ convergent} \end{array}$$

As $L$ is negative, the clause $C$, obviously, does not come from $\mathsf{E}_{\mathcal{A}}$, consequently $C \in \mathsf{D}_{\mathcal{A}} \cup sgi_{\mathcal{A}}(M)$. Put $s' = s{\downarrow}_{R_{M_{\mathcal{A}}}}$ and $t' = t{\downarrow}_{R_{M_{\mathcal{A}}}}$. If $C \in \mathsf{D}_{\mathcal{A}}$, then $s$ and $t$ are ground base terms, hence, by the observation made about rewritings by $R_{M_{\mathcal{A}}}$, so are $s'$ and $t'$. If $C \in sgi_{\mathcal{A}}(M) \setminus \mathsf{D}_{\mathcal{A}}$, then by Proposition 4.15 $s$ and $t$ are non-extension terms, hence, by the observation made about rewritings by $R_{M_{\mathcal{A}}}$, so are $s'$ and $t'$. In both cases, the terms $s'$ and $t'$ cannot be rewritten by $R^{SD}_{M_{\mathcal{A}}}$ any further, as, by Definition 4.13 of $R^{SD}_{M_{\mathcal{A}}}$, the left-hand-side of any rule in $R^{SD}_{M_{\mathcal{A}}}$ is an extension term. Therefore:

$$\begin{array}{cll} & s{\downarrow}_{R'_{M_{\mathcal{A}}}} = s' \neq t' = t{\downarrow}_{R'_{M_{\mathcal{A}}}} & \\ \Leftrightarrow & s \not\leftrightarrow^*_{R'_{M_{\mathcal{A}}}} t & \text{// as } R'_{M_{\mathcal{A}}} \text{ convergent, Prop. 4.16} \\ \Leftrightarrow & \mathcal{I}'_{M_{\mathcal{A}}} \models s \not\approx t & \text{// by Birkhoff's Theorem,} \\ & & \text{// where } \mathcal{I}'_{M_{\mathcal{A}}} = T_{\Omega'}/R'_{M_{\mathcal{A}}} \end{array}$$

Thus, $\mathcal{I}'_{M_{\mathcal{A}}} \models L$, consequently $\mathcal{I}'_{M_{\mathcal{A}}} \models C$. As $C$ has been picked arbitrarily, we conclude $\mathcal{I}'_{M_{\mathcal{A}}} \models M_{\mathcal{A}}$.

**Subgoal $\mathcal{I}'_{M_{\mathcal{A}}} \models M$.** By Lemma 3.61, $M_{\mathcal{A}} \models sgi(M)$, hence $\mathcal{I}'_{M_{\mathcal{A}}} \models sgi(M)$. Assume $\mathcal{A}'$ is an arbitrary model of $R'_{M_{\mathcal{A}}}$. According to Proposition 4.14, any ground extension term is interpreted under $\mathcal{A}'$ as some base term. Consequently, by the Extension Terms Lemma (Lemma 3.19), any ground non-base term of a base sort is interpreted under $\mathcal{A}'$ as some base term:

$$\forall\, t \in T_{\Omega'}(\mathcal{S}) \setminus T_{\Omega}. \exists\, s \in T_{\Omega} : \mathcal{A}'(t) = \mathcal{A}'(s).$$

From this we conclude that for any grounding substitution $\sigma'$ there exists an $\mathcal{A}'$-equivalent simple grounding substitution $\sigma$, in the sense that $dom(\sigma') = dom(\sigma)$, and $\forall x \in dom(\sigma') : \mathcal{A}'(x\sigma') = \mathcal{A}'(x\sigma)$. This implies that for any clause $D \in M$, each ground instance $D\sigma' \in gi(M)$ is equivalent under $\mathcal{A}'$ to some simple ground instance $D\sigma \in sgi(M)$ of the same clause $D$, that is, $\mathcal{A}' \models D\sigma'$ iff $\mathcal{A}' \models D\sigma$. Therefore, $\mathcal{A}' \models sgi(M)$ iff $\mathcal{A}' \models gi(M)$ iff $\mathcal{A}' \models M$. As $\mathcal{A}'$ has been picked arbitrarily, $\mathcal{I}'_{M_{\mathcal{A}}}$ is a model of $R'_{M_{\mathcal{A}}}$ by construction, and $\mathcal{I}'_{M_{\mathcal{A}}} \models sgi(M)$, we conclude $\mathcal{I}'_{M_{\mathcal{A}}} \models M$.

**Subgoal $\mathcal{I}'_{M_{\mathcal{A}}} \in \mathscr{H}_{\mathsf{HSp}}$.** Put $M$ equal to the set of all ground positive unit clauses constructed from the equations (rewrite rules) in $R'_{M_{\mathcal{A}}}$:

$$M \overset{\text{def}}{=} \{ \to s \approx t \mid (s \to t) \in R'_{M_{\mathcal{A}}} \}.$$

Evidently, $\mathcal{I}'_{M_{\mathcal{A}}} \models M$. From $\mathcal{I}'_{M_{\mathcal{A}}} \models M_{\mathcal{A}}$ we know $\mathcal{I}'_{M_{\mathcal{A}}} \models \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}}$. Hence, since $M$ is ground, we conclude $\mathcal{I}'_{M_{\mathcal{A}}} \models M \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} = sgi_{\mathcal{A}}(M) \cup \mathsf{E}_{\mathcal{A}} \cup \mathsf{D}_{\mathcal{A}} = M_{\mathcal{A}}$. From Proposition 4.14 and Definition 3.78, it follows that $M$ is sufficiently complete. According to the Hierarchic Model Lemma (Lemma 3.79), the Herbrand interpretation $\mathcal{I}'_{M_{\mathcal{A}}}$ is a hierarchic model of $M_{\mathcal{A}}$, hence a hierarchic algebra.

From $\mathcal{I}'_{M_{\mathcal{A}}} \models M$ and $\mathcal{I}'_{M_{\mathcal{A}}} \in \mathscr{H}_{\mathsf{HSp}}$, we conclude $\mathcal{I}'_{M_{\mathcal{A}}}$ is a hierarchic model of $M$. ■

### 4.2.4 Termination

Here we show that the SUP(T) calculus, if following a particular strategy, terminates on all clause sets obtained from ground clause sets by basification and abstraction[1]. We prove the termination by first showing the following three statements: (i) every non-Horn clause can be split into Horn clauses; (ii) every base variable occurring in a clause is assigned to some ground base term via the constraint of the clause, and the number of such base terms is limited by a certain finite number; since all variables are base, the number of different variables in a clause is bounded; and (iii) the maximum size of literals is also limited (with an appropriately chosen ordering). Then, we show that these statements and the fact that the enrichment signature is finite imply that after finitely many new clauses have been derived any further clause inferred is the same as some clause previously derived (up to variable renaming).

#### Limiting the Length of Derived Clauses by Exhaustive Splitting

A bounded length of clauses in any SUP(T) derivation can be gained by splitting all clauses into Horn clauses and using eager selection for SUP(T) inferences. In Proposition 4.19 we prove that all clauses derived from a set of Horn clauses by SUP(T) inferences with eager selection do not exceed a certain length.

First, we show that a clause set, basified and abstracted, can actually be split into Horn clauses. Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ be a SUP(T) derivation, where $N_0$ is obtained from a set $N$ of ground $\Sigma'$-clauses by basification and abstraction. In Propositions 4.9 and 4.11, we have shown that for any clause $C = \Lambda \parallel \Gamma \to \Delta$ in the derivation the following hold:

- every base variable $x \in var(C) \cap \mathcal{X}$ is assigned to some ground base term $t$ via an equation $(x \approx t) \in \Lambda$ in the constraint $\Lambda$ of $C$;
- the constraint $\Lambda$ consists of literals of the two types:
    - assignments $x \approx t$ of a base variable $x$ to a ground base term $t$, and

[1]Although all variables occurring in such clause sets are base, we formulate the properties for explicitly distinguished base variables. This enables us to directly extend the results obtained for the ground FOL(T) to non-ground fragments involving non-ground free terms containing free-sorted variables, Section 4.3.

- (dis)equations $s_1 \mathrel{\dot{\approx}} s_2$, where $s_1$ and $s_2$ are ground base terms and $\dot{\approx} \in \{\approx, \not\approx\}$.

From the structure of $\Lambda$, we see that no two distinct variables $x, y \in var(\Lambda)$ occur in the same literal, which means that the constraint $\Lambda$ can always be split into $n$ variable disjoint subsets, where $n = |var(\Lambda)|$, the number of distinct variables occurring in $\Lambda$. Now, assume $x$ is an arbitrary base variable occurring in the free part of $C$ more than once, say $k > 1$ times; let us denote this by $C = \Lambda \parallel (\Gamma \to \Delta)[x : k]$. The variable $x$ is assigned to some ground base term $t \in T_\Omega$ via an equation $(x \approx t) \in \Lambda$. If we replace all subsequent occurrences of the variable $x$ in $\Gamma$ and $\Delta$ with fresh unique base variables $x_2, \ldots, x_k$ and extend the constraint $\Lambda$ with literals $x_2 \approx t, \ldots, x_k \approx t$, we obtain an equivalent clause

$$\Lambda', x_2 \approx t, \ldots, x_k \approx t \parallel (\Gamma' \to \Delta')[x : 1, x_2 : 1, \ldots, x_k : 1],$$

where $x$ and each $x_i$ occur in $\Gamma'$ and $\Delta'$ only once. Applying this transformation to every base variable in $C$ that has multiple occurrences in the free part, gives us a clause $C' = \Lambda'' \parallel \Gamma'' \to \Delta''$ which is equivalent to $C$ and where every base variable $x \in var(C')$ has at most one occurrence in the free part $\Gamma'' \to \Delta''$ of the clause, implying that no two free literals share a base variable. We call the exhaustive application of the transformation to all variables in the clause *variables cloning*.

DEFINITION 4.18 ► Variables Cloning

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

Let $C = \Lambda \parallel \Gamma \to \Delta$ be an abstracted clause, $x \in var(\Lambda)$ a base variable occurring in the constraint of the clause $k > 1$ times and assigned to a ground base term $t \in T_\Omega$ via an equation $(x \approx t) \in \Lambda$ in the constraint of the clause. Replacement of every occurrence of every such a variable $x$ with fresh base variables $x_2, \ldots, x_k \in \mathcal{X}$ of the same sort $sort(x) = sort(x_2) = \ldots = sort(x_k)$ in all literals in $\Lambda$ except the literal $(x \approx t)$, and extension of $\Lambda$ with equations $x_2 \approx t, \ldots, x_k \approx t$ is called ***variables cloning***. We write $VC(C)$ to denote the result of variables cloning applied to the clause $C$. ■

Note that the obtained clause $C' = VC(C)$ enjoys the statements of Propositions 4.9 and 4.11. Since all variables in $C'$ are base, literals in the free part of $C'$ are all variable-disjoint. After applying variables cloning to a non-Horn clause, it can be split into Horn clauses by a sequence of the Hierarchic Splitting rule applications.

As any non-Horn clause in $N_0$ can be split into Horn clauses, we assume form now on that all clauses in $N_0$ are Horn, without loss of generality. Actually, it is sufficient to apply splitting only once exhaustively to clauses in $N_0$, and then the calculus generates only further Horn clauses.

PROPOSITION 4.19 ►

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let* $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ *be a SUP(T) derivation with eager selection. If* $N_0$ *is a Horn clause set, then every clause* $C \in N_i$, $i \geq 1$, *contains at most as many free literals as the longest premise of the inference, deriving* $C$, *does.*

◄ PROOF

An application of the Hierarchic Equality Resolution rule to a clause $C \in N_0$ obviously yields a Horn conclusion with a decremented number of free literals. Consider the inference rules Hierarchic Superposition Left/Right. With eager selection, all negative literals are selected in the antecedent of every clause in $N_0$, therefore an inference is possible only between either two positive unit clauses (Hierarchic Superposition Right), or between a positive unit clause and a Horn clause (Hierarchic Superposition Left), yielding in each case a Horn clause with the number of literals equal, at most, to the number of literals in the longest premise[1]. Applying this argument inductively, the assertion follows for every clause set $N_i$ in the derivation $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$. ■

◄ DEFINITION 4.20
Number $n_{\mathcal{L}}$

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

Given a set $M$ of abstracted clauses, we write $n_{\mathcal{L}}(M)$ to denote the maximum number of free literals in a clause from $M$:

$$n_{\mathcal{L}}(M) \overset{\text{def}}{=} max_{C \in M}\{|\Gamma| + |\Delta| \mid C = \Lambda \parallel \Gamma \to \Delta\}$$

■

◄ COROLLARY 4.21

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let* $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ *be a SUP(T) derivation with eager selection. If* $N_0$ *is a Horn clause set, then the number of free literals in any clause* $C \in N_i$*, for any* $i \geq 0$*, is at most* $n_{\mathcal{L}}(N_0)$.

Since basification, abstraction, and splitting, which produce $N_0$ out of a given set $N$ of ground $\Sigma'$-clauses, can only decrease the number of *free* literals in a clause $C' \in N_0$ in comparison to the number of *all* literals in the corresponding clause $C \in N$, we conclude that the maximum number of free literals in a clause in a SUP(T) derivation with eager selection $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ is smaller than (or equal to) the maximum length of a clause in $N$:

$$n_{\mathcal{L}}(N_i) \leq max_{C \in N}(|C|) \qquad (4.1)$$

**Limiting the Number of Variables in Derived Clauses**

◄ DEFINITION 4.22
Number $n_{\mathcal{V}}$

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively.

Given a set $M$ of abstracted clauses, we write $n_{\mathcal{V}}(M)$ to denote the overall number of ground base terms, to each of which there is an assignment of a variable via an equation in the constraint of a clause in $M$:

$$n_{\mathcal{V}}(M) \overset{\text{def}}{=} \Big| \bigcup_{C \in M} \{t \in T_{\Omega} \mid C = \Lambda \parallel \Gamma \to \Delta, (x \approx t) \in \Lambda\} \Big|$$

■

◄ EXAMPLE 4.23

For the clause set $N''$ from the previous example (page 131), we have $n_{\mathcal{V}}(N'') = 5$,

[1]Actually, the number of free literals is smaller than in the longest premise, only if the inference produces an assignment of two base variables which is then grounded by variables assignments grounding and moved to the constraint of the conclusion.

and the terms, to which base variables are assigned, are (in the order of appearance): $\mathsf{b}, \mathsf{a}, 1 + \mathsf{a}, \mathsf{c}, 1$. ■

PROPOSITION 4.24 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let* $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ *be a SUP(T) derivation, where* $N_0$ *is obtained by basification and abstraction from a set* $N$ *of* $\Sigma'$*-clauses in which all terms of a base sort are ground. Then* $n_{\mathcal{V}}(N_i) \leq n_{\mathcal{V}}(N_0)$*, for every* $i \geq 0$.

PROOF ► For the case $i = 0$ the assertion trivially holds. Let $N_{i+1} = N_i \cup C$, where $C$ is the conclusion of an inference with premises in $N_i$. Assume $n_{\mathcal{V}}(N_i) \leq n_{\mathcal{V}}(N_0)$, induction hypothesis. In the proofs of Propositions 4.9 and 4.11, we have shown that every ground base term $s$, appearing in the constraint of $C$, comes from one of the inference's premises. Therefore, every term $t$, for which there is an assignment $x \approx t$ in the constraint of $C$, is inherited from $N_i$, implying that $n_{\mathcal{V}}(N_{i+1}) \leq n_{\mathcal{V}}(N_i)$, hence, by induction hypothesis, $n_{\mathcal{V}}(N_{i+1}) \leq n_{\mathcal{V}}(N_0)$. ■

Note that Proposition 4.24 requires no eager selection for the derivation, or $N_0$ being a Horn clause set (hence, no splitting or variables cloning), or variables assignments grounding in clauses derived.

According to Proposition 4.11, for an arbitrary clause $C = \Lambda \parallel \Gamma \to \Delta$ in the derivation $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ with variables assignments grounding applied to all derived clauses, it holds that every base literal in $\Lambda$ containing a variable $x$ is of the form $x \approx t$, where $t \in T_\Omega$. Assume the variable $x$ occurs in $\Lambda$ more than once, say $k > 1$ times, then $L_1 = (x \approx t_1), \ldots, L_k = (x \approx t_k)$ are all literals in $\Lambda$ with an entry of $x$, where $t_1, \ldots, t_k \in T_\Omega$. By replacing all occurrences of $x$ in each $L_i$ except $L_1$ with the term $t_1$, we obtain literals $L'_2 = (t_1 \approx t_2), \ldots, L'_n = (t_1 \approx t_n)$; then the following clause

$$(\Lambda \setminus \{L_2, \ldots, L_n\} \cup \{L'_2, \ldots, L'_n\}) \parallel \Gamma \to \Delta,$$

whose constraint is obtained from $\Lambda$ by replacing each $L_i$ with $L'_i$, for all $i \in \{2, \ldots, k\}$, is equivalent to the original clause $C$. Applying this transformation to *every* variable that has multiple occurrences in $\Lambda$, we obtain an equivalent clause $C' = \Lambda' \parallel \Gamma \to \Delta$, in which every base variable $x \in var(C')$ is assigned to a *single* ground base term $t \in T_\Omega$ via an equation $(x \approx t) \in \Lambda'$. We call the exhaustive application of the transformation to all variables in the clause *variable assignments propagation.*

DEFINITION 4.25 ► Variable Assignments Propagation

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

Let $C = \Lambda \parallel \Gamma \to \Delta$ be an abstracted clause, $x \in var(\Lambda)$ a base variable occurring in the constraint of the clause $C$ and assigned to a ground base term $t \in T_\Omega$ via an equation (positive literal) $L = (x \approx t) \in \Lambda$ in the constraint. Replacement of every such a variable with such a term $t$ in all literals in $\Lambda$ except the literal $L$ is called ***variable assignments propagation***. We write $VAP(C)$ to denote the result of variable assignments propagation applied to the clause $C$. ■

Now, assume $x, y \in var(C') \cap \mathcal{X}$ are two arbitrary distinct base variables in the obtained clause $C' = VAP(C)$. The constraint $\Lambda'$ contains two literals $x \approx s$, $y \approx t$, for some ground base terms $s, t \in T_\Omega$, which are uniquely defined. If $s = t$, then the clause $C'[x/y]$ obtained from $C'$ by replacing every occurrence of $y$ with $x$ is equivalent to $C'$. Note that $\Lambda'[x/y]$ contains two identical literals $x \approx s$, one of which can be safely removed. After exhaustively merging all variables that are assigned to the same term, we obtain a clause $C'' = \Lambda'' \parallel \Gamma' \to \Delta'$, which is equivalent to $C'$, hence, equivalent to $C$, and in which any two distinct base variables are assigned to distinct ground base terms via respective equations in the constraint $\Lambda''$ of $C''$:

$$x \neq y \Rightarrow \{x \approx s, y \approx t\} \subseteq \Lambda'' \text{ and } s \neq t,$$

for any two base variables $x, y \in var(C'') \cap \mathcal{X}$ and some ground base terms $s, t \in T_\Omega$. We call the exhaustive application of the transformation to all variables in the clause *variables merging*.

◄ DEFINITION 4.26
Variables Merging

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

Let $C = \Lambda \parallel \Gamma \to \Delta$ be an abstracted clause, $x, y \in var(\Lambda)$ two distinct base variables occurring in the constraint of the clause $C$ and assigned to the same ground base term $t \in T_\Omega$ via equations $(x \approx t), (y \approx t) \in \Lambda$ in the constraint. Replacement of every such a variable $y$ with such a variable $x$ in all literals in $\Lambda$ is called ***variables merging***. We write $VM(C)$ to denote the result of variables merging applied to the clause $C$. ■

Note that variable assignments propagation and variables merging are compatible with Propositions 4.9 and 4.11. Although it is not mentioned in the definitions of the SUP(T) inference rules, we implicitly apply variables assignments grounding, variable assignments propagation, and variables merging to all clauses in the derivation. The cascade of the three transformations is called the *grounding-propagation-merging of variables*.

◄ DEFINITION 4.27
MPG Transformation

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

Let $C = \Lambda \parallel \Gamma \to \Delta$ be an abstracted clause. Application of the cascade of the three transformations[1] (in the given order): variables assignments grounding, variable assignments propagation, and variables merging is called the ***grounding-propagation-merging of variables***. We write $MPG(C)$ to denote the result of applying the combined transformation to a clause $C$:

$$MPG(C) \overset{\text{def}}{=} VM(VAP(VAG(C))).$$

For the sake of conciseness, we use a term ***MPG transformation***[2] for the combined operation. ■

[1]The transformations variables assignments grounding, variable assignments propagation, and variables merging are introduced in Definitions 4.10, 4.25, and 4.26, pages 136, 148, and 149, respectively.

[2]We dedicate this transformation, which plays a central role for establishing decidability results of SUP(T), to the *Max Planck Society for the Advancement of Science* (German: *Max-Planck-Gesellschaft zur Förderung der Wissenschaften e.V.*; abbreviated *MPG*).

It is worthwhile to notice, that variables merging is antagonistic to variables cloning[1], but the former is performed *after* application of an inference rule, whereas the latter is done *prior to* applying the Hierarchic Splitting rule, so that the two operations are not in conflict to each other, particularly because splitting is applied exhaustively before any other inference takes place.

Since every time a clause is derived all its base variables, which are assigned to the same term, are merged, the number of base variables in any clause derived is bounded by the number of different base terms, to which there are assignments in the constraint of the derived clause; and the number of all such terms, as we have shown in Proposition 4.24, is not increasing.

COROLLARY 4.28 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

*Let* $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ *be a SUP(T) derivation, where* $N_0$ *is obtained by basification and abstraction from a set* $N$ *of* $\Sigma'$*-clauses in which all terms of a base sort are ground. If the MPG transformation is applied to every clause in the derivation, then the number of all base variables in any clause in the derivation is at most*[2] $n_{\mathcal{V}}(N_0)$*:*

$$|\,var(C) \cap \mathcal{X}| \leq n_{\mathcal{V}}(N_0),$$

*for any clause* $C \in N_i$*,* $i \geq 0$*.*

Since all variables in a clause $C \in N_i$ in a SUP(T) derivation $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$, where $N_0$ is obtained from a set of ground $\Sigma'$-clauses by basification and abstraction, are base, we conclude that the number of *all* variables in $C$ is bounded by $n_{\mathcal{V}}(N_0)$:

$$var(C) \leq n_{\mathcal{V}}(N_0).$$

**Limiting the Size of Literals in Derived Clauses**

A key condition to ensure a finite saturation of a given clause set $N_0$ by the SUP(T) calculus is to keep the size of literals limited. Here we present a reduction ordering that limits the growth of the literals' size in any SUP(T) derivation underlied by the ordering.

DEFINITION 4.29 ► Hierarchic lexicographic path ordering HLPO($\gamma$)

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

Let $\gamma : T_{\Omega'}(\mathcal{X}') \to \mathbb{N}$ be a total function. The *hierarchic lexicographic path ordering* $\succ_{\mathrm{H}(\gamma)}$ augmenting the function $\gamma$ is defined by: $t \succ_{\mathrm{H}(\gamma)} s$ iff:

1. $\gamma(t) > \gamma(s)$, or

2. $\gamma(t) = \gamma(s)$ and $t \succ_{\mathrm{lpo}} s$,

[1]The variables cloning transformation is introduced in Definition 4.18, page 146.
[2]Please, recall Definition 4.22 of the number $n_{\mathcal{V}}(N_0)$, page 147.

where $\succ_{\text{lpo}}$ is the standard lexicographic path ordering[1]. ■

In general, the above HLPO($\gamma$) ordering is not stable under substitutions. However in the context of simple substitutions and an appropriately defined function $\gamma$, it actually becomes a reduction ordering which is well-founded and total on ground terms.

◄ DEFINITION 4.30
Function $\omega$

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

We define a function $\omega : T_{\Omega'}(\mathcal{X}') \to \mathbb{N}$, which given a $\Sigma'$-term $t \in T_{\Omega'}(\mathcal{X}')$ returns the number of free operator symbol occurrences in it, as follows:

$$\omega(t) \overset{\text{def}}{=} \left|\{p \in \rho(t) \mid top(t/p) \in \Omega''\}\right|.$$

The function $\omega$ extends onto atoms and literals as

$$\omega(s \mathrel{\dot{\approx}} t) \overset{\text{def}}{=} \omega(s) + \omega(t),$$

where $s, t \in T_{\Omega'}(\mathcal{X}')$ and $\dot{\approx} \in \{\approx, \not\approx\}$. ■

An instance of the Hierarchic lexicographic path ordering HLPO($\omega$) augmenting the function $\omega$ compares two terms by, first, comparing the number of free operator symbol occurrences in the terms, and, second, if the number of free operator symbol occurrences is the same, by comparing the terms with respect to the standard LPO.

In the context of the SUP(T) calculus for the ground FOL(T) fragment, terms involved in inferences may contain only base variables. Moreover, SUP(T) admits only simple substitutions, hence the variables may be substituted only with base terms. Next, we show that $\succ_{\text{H}(\omega)}$ is a reduction ordering under these conditions. Lemma 4.31 and Proposition 4.33 followed hold only in the context of the ground FOL(T) fragment – for the non-ground case we give corresponding statements in Section 4.3.

◄ LEMMA 4.31

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

*The hierarchic lexicographic path ordering* $\succ_{\text{H}(\omega)}$ *augmenting the function* $\omega$ *is a reduction ordering for all* $\Sigma'$*-terms over base variables* $T_{\Omega'}(\mathcal{X})$ *with respect to simple substitutions.*

◄ PROOF

We need to show that $\succ_{\text{H}(\omega)}$ is irreflexive, transitive, and well-founded binary relation that is compatible with contexts, stable under simple substitutions, and has the subterm property. Let $t, s, r \in T_{\Omega'}(\mathcal{X})$ be arbitrary $\Sigma'$-terms over base variables $\mathcal{X}$.

**Irreflexivity** of $\succ_{\text{H}(\omega)}$ follows from that of $>$ and $\succ_{\text{lpo}}$.

[1] Please, recall Definition 2.15 of the standard LPO.

**Transitivity**. Assume $t \succ_{\mathrm{H}(\omega)} s \succ_{\mathrm{H}(\omega)} r$. We are to show that $t \succ_{\mathrm{H}(\omega)} r$. Note that $\omega(t) \geq \omega(s) \geq \omega(r)$. Consider two possible cases:

(i) if the terms have the same number of free operator symbol occurrences:

$$\omega(t) \;=\; \omega(s) \;=\; \omega(r),$$

then

$$t \;\succ_{\mathrm{lpo}}\; s \;\succ_{\mathrm{lpo}}\; r,$$

and $t \succ_{\mathrm{lpo}} r$ follows by transitivity of $\succ_{\mathrm{lpo}}$, consequently $t \succ_{\mathrm{H}(\omega)} r$;

(ii) $\omega(t) > \omega(s)$ or $\omega(s) > \omega(r)$, then $\omega(t) > \omega(r)$, consequently $t \succ_{\mathrm{H}(\omega)} r$.

**Well-foundedness**. Let $t_1, t_2, \ldots$ be arbitrary terms from $T_{\Omega'}(\mathcal{X})$ such that

$$t_1 \;\succ_{\mathrm{H}(\omega)}\; t_2 \;\succ_{\mathrm{H}(\omega)} \ldots.$$

We are to show that any such a descending chain of terms is finite. We prove it by induction on the number $\omega(t_1)$ of free operator symbol occurrences in the largest term $t_1$ in the chain.

*Induction Base.* If $\omega(t_1) = 0$, then

$$\omega(t_1) \;=\; \omega(t_2) \;=\; \ldots \;=\; 0.$$

Therefore

$$t_1 \;\succ_{\mathrm{lpo}}\; t_2 \;\succ_{\mathrm{lpo}}\; \ldots,$$

and the chain $t_1 \succ_{\mathrm{H}(\omega)} t_2 \succ_{\mathrm{H}(\omega)} \ldots$ is finite by well-foundedness of $\succ_{\mathrm{lpo}}$.

*Induction Hypothesis.* Assume that any descending chain

$$t_1 \;\succ_{\mathrm{H}(\omega)}\; t_2 \;\succ_{\mathrm{H}(\omega)} \ldots,$$

where $\omega(t_1) \leq n$ for some $n \geq 0$, is finite.

*Induction Step.* Assume $\omega(t_1) = n + 1$. There are two possible cases:

(i) if the terms have the same number of free operator symbol occurrences:

$$\omega(t_1) \;=\; \omega(t_2) \;=\; \ldots \;=\; n+1,$$

then the case is analogous to the induction base;

(ii) otherwise, there exists an index $\ell > 1$ such that $\omega(t_1) > \omega(t_\ell)$. Let $k$ be the smallest index such that $\omega(t_1) \geq \omega(t_k) + 1$. Since

$$\omega(t_1) \;=\; \omega(t_2) \;=\; \ldots \;=\; \omega(t_{k-1}),$$

we know that

$$t_1 \;\succ_{\mathrm{lpo}}\; t_2 \;\succ_{\mathrm{lpo}} \ldots \succ_{\mathrm{lpo}}\; t_{k-1}.$$

By well-foundedness of $\succ_{\mathrm{lpo}}$, the left subchain

$$t_1 \;\succ_{\mathrm{H}(\omega)}\; t_2 \;\succ_{\mathrm{H}(\omega)} \ldots \succ_{\mathrm{H}(\omega)}\; t_{k-1}$$

is finite (and so is $k$). By induction hypothesis, the right subchain

$$t_k \;\succ_{\mathrm{H}(\omega)}\; t_{k+1} \succ_{\mathrm{H}(\omega)} \ldots$$

is finite as well.

Thus, any descending chain $t_1 \succ_{\mathrm{H}(\omega)} t_2 \succ_{\mathrm{H}(\omega)} \ldots$ is finite, and, hence, $\succ_{\mathrm{H}(\omega)}$ is well-founded.

**Compatibility with contexts**. Let $r = f(r_1,\dots,r_n)$, for some operator symbol $f \in \Omega'$ of arity $n \geq 1$, and some $\Sigma'$-terms $r_1,\dots,r_n \in T_{\Omega'}(\mathcal{X})$ over base variables $\mathcal{X}$. Assume $t \succ_{\mathrm{H}(\omega)} s$. Consider terms

$$\begin{aligned} r' &= r[t]_i, \\ r'' &= r[s]_i, \end{aligned}$$

obtained from the $r$ by replacing its $i$-th immediate subterm $r/i = r_i$ by $t$ and $s$, respectively, for some $1 \leq i \leq n$. Then

$$\begin{aligned} \omega(r') &= \omega(r) - \omega(r/i) + \omega(t), \\ \omega(r'') &= \omega(r) - \omega(r/i) + \omega(s). \end{aligned}$$

If $\omega(t) > \omega(s)$, then $\omega(r') > \omega(r'')$, thus $r' \succ_{\mathrm{H}(\omega)} r''$ by condition 1 of Definition 4.29. If $\omega(t) = \omega(s)$, then $t \succ_{\mathrm{lpo}} s$ and $\omega(r') = \omega(r'')$. Let us compare $r'$ and $r''$ with respect to $\succ_{\mathrm{lpo}}$. Since $r'/j = r''/j$, for every $1 \leq j \leq n$, $j \neq i$, and

$$r'/i = t \succ_{\mathrm{lpo}} s = r''/i,$$

we obtain, by condition 2a of Definition 2.15 of the standard LPO, that $r' \succ_{\mathrm{lpo}} r''/j$, for every $1 \leq j \leq n$. Moreover,

$$(r'/1,\dots,r'/n)\ (\succ_{\mathrm{lpo}})_{\mathrm{lex}}\ (r''/1,\dots,r''/n),$$

hence, by condition 2c of Definition 2.15, we obtain $r' \succ_{\mathrm{lpo}} r''$, yielding, by condition 2 of Definition 4.29, that $r' \succ_{\mathrm{H}(\omega)} r''$. Thus, $\succ_{\mathrm{H}(\omega)}$ is compatible with contexts.

**Stability under simple substitutions**. Assume $t \succ_{\mathrm{H}(\omega)} s$. Let $\sigma$ be an arbitrary simple substitution. Since all variables occurring in $t$ and $s$ are of a base sort, they can be mapped by $\sigma$ only to base terms, therefore $\omega(t\sigma) = \omega(t)$ and $\omega(s\sigma) = \omega(s)$. If $\omega(t) > \omega(s)$, then $\omega(t\sigma) > \omega(s\sigma)$ and $t\sigma \succ_{\mathrm{H}(\omega)} s\sigma$. If $\omega(t) = \omega(s)$, then $t \succ_{\mathrm{lpo}} s$. Since $\succ_{\mathrm{lpo}}$ is stable under substitutions, we know $t\sigma \succ_{\mathrm{lpo}} s\sigma$. Consequently, $t\sigma \succ_{\mathrm{H}(\omega)} s\sigma$, by condition 2 of Definition 4.29. Thus, $\succ_{\mathrm{H}(\omega)}$ is stable under simple substitutions.

**Subterm property**. Let $t = f(t_1,\dots,t_n)$, for some operator symbol $f \in \Omega'$ of arity $n \geq 1$, and some $\Sigma'$-terms $t_1,\dots,t_n \in T_{\Omega'}(\mathcal{X})$ over base variables $\mathcal{X}$; let $s = t_i$ be an immediate subterm of $t$, for some $i \in \{1,\dots,n\}$. If $f \in \Omega''$, then $\omega(t) \geq \omega(s) + 1$, consequently $\omega(t) > \omega(s)$ and $t \succ_{\mathrm{H}(\omega)} s$. Otherwise, $f \in \Omega$, and $\omega(t) \geq \omega(s)$. If $\omega(t) > \omega(s)$, then $t \succ_{\mathrm{H}(\omega)} s$. Otherwise, $\omega(t) = \omega(s)$, but since $\succ_{\mathrm{lpo}}$ possesses the subterm property, we know $t \succ_{\mathrm{lpo}} s$, consequently $t \succ_{\mathrm{H}(\omega)} s$. Thus, $\succ_{\mathrm{H}(\omega)}$ possesses the subterm property. ∎

◄ DEFINITION 4.32
Number $n_{\mathcal{F}}$

Assume HSp = (Sp, Sp′) is a hierarchic specification with the base specification Sp = $(\Sigma, \mathscr{C})$ and the body Sp′ = $(\Sigma', Ax')$.

Given a set $M$ of abstracted clauses, we write $n_{\mathcal{F}}(M)$ to denote the maximum number of free operator symbol occurrences in a free literal among all clauses in $M$:

$$n_{\mathcal{F}}(M) \stackrel{\mathrm{def}}{=} max_{C \in M}\{\omega(L) \mid L \in (\Gamma \to \Delta), C = \Lambda \parallel \Gamma \to \Delta\}.$$

∎

◄ PROPOSITION 4.33

*Assume* HSp = (Sp, Sp′) *is a hierarchic specification with the base specification* Sp = $(\Sigma, \mathscr{C})$ *and the body* Sp′ = $(\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

*Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ be a SUP(T) derivation, where $N_0$ is obtained from a set of ground $\Sigma'$-clauses by basification and abstraction. If $\succ_{\mathrm{H}(\omega)}$ is the underlying ordering, then for every free literal $L$ in any clause $C \in N_i$, $i \geq 0$, the number $\omega(L)$ of free operator symbol occurrences is at most $n_{\mathcal{F}}(N_0)$.*

PROOF ► We give a proof by induction on the length of derivation of $N_i$.

*Induction Base.* For $N_0$ the assertion trivially holds.

*Induction Hypothesis.* Assume the assertion holds for all clause sets $N_i$ derived from $N_0$, where $i \leq n$, for some $n > 0$.

*Induction Step.* Let $N_{n+1} = N_n \cup C$, where $C$ is a conclusion of an inference rule application to some clauses in $N_n$. Let us first make the following observation: for any simple substitution $\sigma$ and any term $t \in T_{\Omega'}(\mathcal{X})$ built over operators $\Omega'$ and base variables $\mathcal{X}$, the term $t\sigma$ contains as many free operator symbols as the term $t$ does, i.e. $\omega(t\sigma) = \omega(t)$, because (i) there is no non-base variable in $t$, and (ii) simple substitutions can map base variables only to base terms. Consider a Hierarchic Superposition Right inference, Definition 3.28,

$$\mathcal{I}\frac{\Lambda_1 \parallel \Gamma_1 \rightarrow \Delta_1, l \approx r \qquad \Lambda_2 \parallel \Gamma_2 \rightarrow \Delta_2, s[l'] \approx t}{(\Lambda_1, \Lambda_2 \parallel \Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2, s[r] \approx t)\sigma}$$

with the premises $C_1, C_2 \in N_n$ and conclusion $C$, respectively. Thus,

$$\begin{array}{rcl} & l\sigma = l'\sigma & \text{// by Cond. ((i)) of Def. 3.28} \\ \Rightarrow & \omega(l\sigma) = \omega(l'\sigma) & \\ \Rightarrow & \omega(l) = \omega(l') & \text{// as } l, l' \in T_{\Omega'}(\mathcal{X}) \text{ and } \sigma \text{ simple} \end{array}$$

Moreover,

$$\begin{array}{rcl} & r\sigma \not\succ_{\mathrm{H}(\omega)} l\sigma & \text{// by Cond. ((iii)) of Def. 3.28} \\ \Rightarrow & \omega(r\sigma) \leq \omega(l\sigma) & \text{// acc. to Def. 4.29 of } \succ_{\mathrm{H}(\omega)} \\ \Rightarrow & \omega(r) \leq \omega(l) & \text{// as } l, r \in T_{\Omega'}(\mathcal{X}) \text{ and } \sigma \text{ simple} \end{array}$$

Therefore,

$$\begin{array}{rrl} & \omega(s[r]\sigma) = \omega(s[r]) & \text{// as } s[r] \in T_{\Omega'}(\mathcal{X}) \text{ and } \sigma \text{ simple} \\ & \leq \omega(s[l]) & \text{// as } \omega(r) \leq \omega(l) \\ & = \omega(s[l']) & \text{// as } \omega(l) = \omega(l') \\ \Rightarrow & \omega\big((s[r] \approx t)\sigma\big) \leq \omega(s[l'] \approx t) & \\ & \leq n_{\mathcal{F}}(N_0) & \text{// by Induction Hypothesis} \end{array}$$

For any other literal $L\sigma$ in the inference's conclusion, where $L$ is the corresponding literal from a premise, we have $\omega(L\sigma) = \omega(L) \leq n_{\mathcal{F}}(N_0)$, by Induction Hypothesis. The analysis of the other rules is similar. ■

LEMMA 4.34 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ *and* $\Omega'' = \Omega' \setminus \Omega$ *be the enrichment sorts and operators, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

*Let $N_0$ be obtained from a finite set $N$ of ground $\Sigma'$-clauses by basification and abstraction. If $N_0$ is a Horn clause set, then any fair SUP(T) derivation $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ with eager selection, MPG transformation, and reduction ordering $\succ_{\mathrm{H}(\omega)}$ is terminating.*

◄ PROOF According to Proposition 4.11, any literal in the constraint of a clause in the derivation is either a (dis)equation $s_1 \mathrel{\dot{\approx}} s_2$ between some ground base terms $s_1, s_2 \in T_\Omega$, where $\dot{\approx} \in \{\approx, \not\approx\}$, or an assignment $x \approx t$ of a base variable $x \in \mathcal{X}$ to a ground base term $t \in T_\Omega$. In the proof of Proposition 4.11 we have shown that the terms $s_1, s_2, t$ are all inherited from $N_0$, meaning that no new base term is produced by an inference. In Proposition 4.28, we have shown that a clause in the derivation may contain at most $n_\mathcal{V}(N_0)$ different base variables. For finitely many different terms $s_1, s_2, t$ and a limited number of variables, there can be only finitely many different literals of the above form, therefore there can be only finitely many different constraints $\Lambda$, up to variable renaming and removal of duplicate literals.

According to Corollary 4.21, any clause in the derivation may contain at most $n_\mathcal{L}(N_0)$ free literals, each of which, according to Proposition 4.33, may have at most $n_\mathcal{F}(N_0)$ operator symbol occurrences, limiting thus the size of free literals. As all variables in the derivation are base, the number of different variables in the free part of a clause is also limited by $n_\mathcal{V}(N_0)$. For the finite set $\Omega''$ of free operator symbols occurring in $N_0$, a limited number and size of literals, and limited number of variables, there exist only finitely many different free parts $\Gamma \to \Delta$, up to variable renaming.

Combining the two observations above, we conclude that there can be only finitely many different clauses $\Lambda \parallel \Gamma \to \Delta$ in the derivation $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$, up to variable renaming. Thus, after sufficiently (and finitely) many clauses have been derived, any inference from $N_i$, for some $i \geq 0$, produces a clause that is already present in $N_i$, up to variable renaming, hence the inference is redundant. Since only irredundant inferences have to be performed, the derivation $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ is terminating. ■

Please, note that fact that only finitely many different constraints $\Lambda$ can be derived, up to variable renaming and removal of duplicate literals, implies that only finitely many different base clauses are derivable. To establish the existence of a finite number of different constraints, we have only used the properties stated in Propositions 4.11 and 4.28, which hold for all derivations $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ with the MPG transformation, where $N_0$ is obtained by basification and abstraction from a set $N$ of $\Sigma'$-clauses in which all terms of a base sort are ground. Hence, any SUP(T) derivation with the MPG transformation from any such clause set $N_0$ produces only finitely many diferent base clauses, up to a very trivial subsumption.

### 4.2.5 Decidability

According to the definition of SUP(T) inference rules, Section 3.3, an empty clause $\square$ can be produced only by an application of the Constraint Refutation rule, which is to apply to a set of base clauses (which are abstracted clauses with the free part empty). As every base variable $x$ is assigned to a ground base term $t$, Proposition 4.9, any non-ground base clause can be grounded by propagation of all assignments $x \approx t$. The term $t$ may contain base parameters introduced at the basification step. The parameters are essentially base variables existentially quantified on the top of the overall clause set. Thus, an application of the Constraint Refutation rule[1]

$$\mathcal{I}\frac{C_1[\mathsf{a}^1_1,\ldots,\mathsf{a}^1_{m_1}] \quad \ldots \quad C_n[\mathsf{a}^n_1,\ldots,\mathsf{a}^n_{m_n}]}{\square}$$

where $\mathsf{a}^1_1,\ldots,\mathsf{a}^n_{m_n}$ are all base parameters occurring in the premises $C_1,\ldots,C_n$ (any two parameters $\mathsf{a}^i_j$ and $\mathsf{a}^{i'}_{j'}$ do not have to be necessarily different), reduces to checking satisfiability of a base formula

$$\exists y^1_1,\ldots,y^n_{m_n} : C_1[y^1_1,\ldots,y^1_{m_1}] \wedge \ldots \wedge C_n[y^n_1,\ldots,y^n_{m_n}],$$

in which the base parameters are replaced with respective fresh base variables $y^1_1,\ldots,y^n_{m_n} \in \mathcal{X}$, which are all existentially quantified over the theory domain. For this reason, the only requirement we impose on the theory T (base specification Sp) is the decidability of its existential fragment.

THEOREM 4.35 ► Ground FOL(T) Decidability

*Assume* $\mathsf{HSp} = (\mathsf{Sp},\mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma,\mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*If the* $\exists^*$ *fragment is decidable in* $\mathsf{Sp}$, *then the SUP(T) calculus with eager selection, MPG transformation, reduction ordering* $\succ_{\mathrm{H}(\omega)}$, *and variables cloning and splitting preferred to other inferences, is a decision procedure for the ground clause FOL(T) fragment*[2].

PROOF ► Basification, presented in Sections 4.2.1, transforms a finite set $N$ of ground $\Sigma'$-clauses into a finite equisatisfiable clause set $N'$, such that every model of $N'$ is a model of $N$, Lemma 4.5.

Let $N''$ be a clause set obtained from $N'$ by abstraction. In Section 4.2.4 we have shown that every clause in $N''$ can be split into Horn clauses, if variables cloning is first applied to it. Let $N^1_0,\ldots,N^n_0$ be the sets of Horn clauses obtained by exhaustively applying splitting to $N''$. The clause set $N''$ has a hierarchic model if and only if some $N^i_0$, $i \in \{1,\ldots,n\}$, does so. According to Lemma 4.34, any fair SUP(T) derivation $N^i_0 \vdash N^i_1 \vdash N^i_2 \vdash \ldots$ with eager selection, MPG transformation, and the reduction ordering $\succ_{\mathrm{H}(\omega)}$ is terminating. Let $N^i_k$ be the last clause set in the derivation $N^i_0 \vdash N^i_1 \vdash \ldots \vdash N^i_k$, for some $k \geq 0$. The set $N^i_k$ is the limit of the derivation. Moreover, $N^i_k$ is finite. By Lemma 3.9, the clause set $N^i_k$ is saturated. If $\square \notin N^i_k$, then the set $N^i_k \cap Cl_\Sigma$ of all base clauses within $N^i_k$ is theory-consistent. Indeed, suppose for the sake of contradiction that $N^i_k \cap Cl_\Sigma$ is theory-inconsistent. As $N^i_k$ is finite, the set $N^i_k \cap Cl_\Sigma$ is also finite. Therefore, a constraint refutation inference derives $\square$ from $N^i_k \cap Cl_\Sigma$. Since $N^i_k$ is saturated, the inference must be redundant with respect to $\mathcal{R}^{\mathcal{H}}$, which holds, according to Definition 3.64 of $\mathcal{R}^{\mathcal{H}}$, if and only if an empty clause $\square$ is already in $N^i_k$, a contradiction. Let $\mathcal{A} \in \mathscr{C}$ be an arbitrary base algebra satisfying the set $N^i_k \cap Cl_\Sigma$. By Lemma 4.17, the clause set $N^i_k$ has a hierarchic model. This model is also a model of $N^i_0$. If $\square \in N^i_k$, then $N^i_k$ is unsatisfiable due to soundness of SUP(T).

As the base theory supports a decision procedure for the $\exists^*$ fragment, Constraint Refutation can be decided, therefore satisfiability of every $N^i_0$ can be determined, hence satisfiability of $N''$ and $N$ can be determined as well. Thus, the

[1]The Constraint Refutation rule is introduced in Definition 3.31, page 48.

[2]The ground clause FOL(T) fragment is represented by the class $\mathcal{GC}_{\Sigma'}$ of all sets of ground $\Sigma'$-clauses.

SUP(T) calculus with eager selection, MPG transformation, reduction ordering $\succ_{\mathrm{H}(\omega)}$, and variables cloning and splitting preferred to other inferences, is a decision procedure for the ground clause FOL(T) fragment. ■

Note that the usage of the hierarchic LPO $\succ_{\mathrm{H}(\omega)}$ yields an upper bound on the maximum size of free literals and, hence, guarantees the existence of only finitely many different free literals over a finite set $\Omega''$ of free operator symbols and a limited number of variables. Thus, even if the length of the free part of a clause not limited, there can be only finitely many different free parts $\Gamma \to \Delta$, up to variable renaming and duplication of literals. This observation supports an alternative strategy involving neither splitting nor eager selection: indeed, the two ingredients are not really needed for termination, but what is required in return is an additional Hierarchic Factoring rule, which is necessary to guarantee hierarchic model existence in the absence of the splitting rule.

Note that the Hierarchic Completeness Theorem (Theorem 3.82) requires the base specification to be compact, which is needed for ensuring the existence of a base model of the base subset set of the saturated clause set (see the proof of the $\mathcal{H}$-$\mathcal{F}$ Approximation Theorem on page 98 for further details). Basification may lead losing compactness of the base specification. Nevertheless, the compactness requirement is obsolete for the fragments considered here as all possible SUP(T) derivations from clause sets over these fragments are finite.

## 4.3 Deciding Non-Ground FOL(T)

In this section we consider an application of SUP(T) to a non-ground FOL(T) fragment, for which SUP(T) is a decision procedure as well. Due to the hierarchic design of SUP(T), the decidability result for the ground FOL(T) fragment can be easily extended to some non-ground fragments. Here we present an instance of such an extension.

### 4.3.1 BSHE(GBST) Class

DEFINITION 4.36 ► BSHE(GBST) class Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively.

We call the class consisting of all Horn clause sets, where

(i) every non-predicate operator symbol of non-zero arity ranges into a base sort, and

(ii) all base sort terms are ground,

the ***Bernays-Schönfinkel Horn class with equality and ground base sort terms***, for short ***BSHE(GBST)***. ■

If $N$ is a clause set from the BSHE(GBST) class, then any literal occurring in a clause from $N$ may contain a base or free predicate symbol, equality symbol, variables of a free sort which all occur immediately below a free predicate or equality symbol, base function symbols, extension symbols, and constant symbols either of a base or a free sort. Thus, any non-predicative compound term occurring in $N$ is ground and of a base sort.

EXAMPLE 4.37 ► Let $N$ be a clause set containing just one clause:

$$N = \{\, f(a) + 2 > b,\ P(u, 3f(a), c) \ \rightarrow\ Q(u, b)\,\},$$

where the background theory T is the rational linear arithmetic; $+, >, 2, 3 \in \Omega$, and $P, Q, f, a, b, c \in \Omega''$; the function symbols $f, b$ ranging into the base sort $\mathsf{S}$, and $a, c$ ranging into a free sort $\mathsf{S}'' \in \mathcal{S}''$; $u \in \mathcal{X}''$ is a non-base variable. The set $N$ agrees with Definition 4.36 of the BSHE(GBST) class.

As usual, the input clause set $N$ is first basified resulting in a clause set $N' = N_I \cup N_B$:

$$\begin{aligned} N_I &= \{\, \mathsf{a} + 2 > \mathsf{b},\ P(u, 3\mathsf{a}, c) \ \rightarrow\ Q(u, \mathsf{b})\,\},\\ N_B &= \{ \qquad\qquad\qquad\qquad\quad \rightarrow\ f(a) \approx \mathsf{a},\\ &\qquad\qquad\qquad\qquad\qquad\quad\ \ \rightarrow\ b \approx \mathsf{b}\,\} \end{aligned}$$

and then abstracted into a clause set $N'' = N'_I \cup N'_B$:

$$\begin{aligned} N'_I &= \{\, \mathsf{a} + 2 > \mathsf{b},\ x \approx 3\mathsf{a},\ y \approx \mathsf{b} \parallel P(u, x, c) \ \rightarrow\ Q(u, y)\,\},\\ N'_B &= \{ \qquad\qquad\qquad x \approx \mathsf{a} \parallel \qquad\qquad \rightarrow\ f(a) \approx x,\\ &\qquad\qquad\qquad\qquad\ x \approx \mathsf{b} \parallel \qquad\qquad \rightarrow\ b \approx x\,\}. \end{aligned}$$

■

Let $N$ be an arbitrary clause set from the BSHE(GBST) class. From now on we write $N' = N_I \cup N_B$ to denote the clause set obtained from $N$ by basification, where $N_I$ is the set of the basified clauses from $N$, and $N_B$ is the set of clauses basifying those in $N_I$; and $N''$ to denote the clause set $N'$ abstracted. Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ be a SUP(T) derivation with MPG transformation, where $N_0 = N''$. Most results for the ground fragment stated in propositions and corollaries of the previous section hold also for the BSHE(GBST) class, and their proofs are valid in the current context as well, therefore here we only reformulate the related properties to fit the BSHE(GBST) class without providing proofs.

## 4.3.2 Derivation Invariants

The presence of non-base variables interferes neither basification nor abstraction because they may occur only immediately below equality or free predicate symbols. Therefore, terms occurring in basified BSHE(GBST) clause sets have the same structural properties as in the case of basified ground FOL(T) clause sets.

◄ PROPOSITION 4.38

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*Let $N$ be a clause set from the BSHE(GBST) class. Let $N' = N_I \cup N_B$ be a clause set obtained from $N$ by basification, where $N_I$ is the set of the basified clauses from $N$, and $N_B$ is the set of clauses basifying those in $N_I$. No extension term appears in a clause from $N_I$. The only literal $L = t \approx \mathsf{a}$ in any clause from $N_B$ is an equation between a base parameter* $\mathsf{a}$ *and a ground smooth extension term $t \in T^E_{\Omega'}$.*

◄ COROLLARY 4.39

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*Let $N$ be a clause set from the BSHE(GBST) class, $N'$ a clause set obtained from $N$ by basification, and $L = t_1 \mathrel{\dot{\approx}} t_2$ a literal in a clause from $N'$, where $\dot{\approx} \in \{\approx, \not\approx\}$. Every subterm of each $t_i$ with a base top operator symbol is a ground base term:*

$$\forall\, p \in \rho(t_i) : top(t_i/p) \in \Omega \Rightarrow t_i/p \in T_\Omega,$$

*for every $i \in \{1,2\}$.*

Likewise, basification of a BSHE(GBST) clause set $N$ produces an equisatisfiable clause set $N'$ such that every model of $N'$ is a model of $N$.

◄ LEMMA 4.40

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $N$ be a BSHE(GBST) clause set.If $N'$ is obtained from $N$ by basification, then:*

*(i) $N'$ and $N$ equisatisfiable;*

*(ii) $N' \models N$.*

The derivation invariants regarding the structure of constraints hold in the context of the BSHE(GBST) class too.

PROPOSITION 4.41 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

*Let* $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ *be a SUP(T) derivation, where* $N_0$ *is obtained by basification and abstraction from a BSHE(GBST) clause set; let* $C = \Lambda \parallel \Gamma \to \Delta$ *be an arbitrary clause in* $N_i$*,* $i \geq 0$*. Then for every base variable* $x \in var(C) \cap \mathcal{X}$ *there exists a ground base term* $t \in T_\Omega$ *such that an equation* $x \approx t$ *is in the constraint* $\Lambda$ *of* $C$*.*

PROPOSITION 4.42 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

*Let* $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ *be a SUP(T) derivation, where* $N_0$ *is obtained by basification and abstraction from a BSHE(GBST) clause set; let* $C = \Lambda \parallel \Gamma \to \Delta$ *be an arbitrary clause in* $N_i$*,* $i \geq 0$*. If variables assignments grounding*[1] *is applied to every clause derived, then every literal in* $\Lambda$ *is either*

- *an assignment* $x \approx t$*, or*
- *a (dis)equation* $s_1 \mathrel{\dot{\approx}} s_2$*,*

*where* $x \in \mathcal{X}$ *is a base variable,* $t, s_1, s_2 \in T_\Omega$ *ground base terms,* $\dot{\approx} \in \{\approx, \not\approx\}$*.*

In the BSHE(GBST) fragment the only function symbols ranging into a free sort are free constants. Therefore a non-base variable can be substitute either with another non-base variable or a free constant. For this reason, the structural properties of free literals stated in Propositions 4.12 and 4.15 extend also onto the BSHE(GBST) class.

PROPOSITION 4.43 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively. Let* $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ *be the underlying variable set consisting of base and non-base variables, respectively.*

*Let* $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ *be a SUP(T) derivation, where* $N_0$ *is obtained by basification and abstraction from a BSHE(GBST) clause set, and C an arbitrary clause in the derivation. All extension terms occurring in C, if any, are smooth and appear only in literals of form* $t \approx x$*, where* $t \in T^E_{\Omega'}(\mathcal{X}')$ *is an extension term and* $x \in \mathcal{X}$ *a base variable.*

PROPOSITION 4.44 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base and body signatures, respectively.*

*Let* $N_0 \vdash N_1 \vdash \ldots$ *a SUP(T)-derivation, where* $N_0$ *is obtained by basification and abstraction from a BSHE(GBST) clause set, and C an arbitrary clause in* $sgi(N_i)$*, for some* $i \geq 0$*. All extension terms occurring in C, if any, are smooth and appear only in literals of form* $t \approx s$*, where* $t \in T^E_{\Omega'}$ *is an extension term and* $s \in T_\Omega$ *a ground base term.*

[1]Please, recall Definition 4.10 of variables assignments grounding, page 136.

### 4.3.3 Model Existence

Let $N_0 \vdash N_1 \vdash \ldots$ a SUP(T)-derivation, where $N_0$ is obtained by basification and abstraction from a BSHE(GBST) clause set. Proposition 4.44 allows us to exploit the mechanism used to ensure the hierarchic model existence for the ground fragment here as well. Indeed, the only requirement imposed on a clause set $M_{\mathcal{A}}$, where $M = N_\infty$ is the limit of the derivation $N_0 \vdash N_1 \vdash \ldots$, is the appearance of extension terms only in positive literals of form $t \approx s$, where $t$ is a smooth extension term and $s$ a ground base term. If the requirement is satisfied, the rewrite system $R'_{M_{\mathcal{A}}} = R_{M_{\mathcal{A}}} \cup R^{SD}_{M_{\mathcal{A}}}$ is convergent, where $R_{M_{\mathcal{A}}}$ is a rewrite system constructed from the maximal literals of productive clauses in $M_{\mathcal{A}}$, Definition 2.29, and the rewrite system $R^{SD}_{M_{\mathcal{A}}}$ sufficiently defines terms which are not sufficiently defined by $R_{M_{\mathcal{A}}}$, Definition 4.13. Convergency of $R'_{M_{\mathcal{A}}}$ is needed to show that the Herbrand interpretation $\mathcal{I}'_{M_{\mathcal{A}}} = T_{\Omega'}/R'_{M_{\mathcal{A}}}$ is a hierarchic model of $N$.

PROPOSITION 4.45 *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $M = N_\infty$ be the limit of a fair SUP(T)-derivation $N_0 \vdash N_1 \vdash \ldots$, where $N_0$ is obtained by basification and abstraction from a BSHE(GBST) clause set. Let $\mathcal{A} \in \mathscr{C}$ an arbitrary base algebra. The rewrite system $R'_{M_{\mathcal{A}}} = R_{M_{\mathcal{A}}} \cup R^{SD}_{M_{\mathcal{A}}}$ is convergent.*

THEOREM 4.46 *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $M = N_\infty$ equal the limit of a fair SUP(T)-derivation $N_0 \vdash N_1 \vdash \ldots$, where $N_0$ is obtained by basification and abstraction from a BSHE(GBST) clause set. Assume there exists a base algebra $\mathcal{A} \in \mathscr{C}$ satisfying the set $M \cap Cl_\Sigma$ of all base clauses within $M$. If $M$ does not contain an empty clause $\square$, then $M$ has a hierarchic model.*

### 4.3.4 Termination

Since we consider only Horn clauses, the set $N_0$ is already Horn (as neither basification nor abstraction add atoms to the succedent of a clause), hence no splitting (and variable cloning) needs to be applied in any derivation from $N_0$. Thus, the properties regarding the maximal number of free literals stated in Proposition 4.19 and subsequent Corollary 4.21 are true for the BSHE(GBST) fragment as well. Also, the assertions on the number of base variables in clauses derived stand in this context too.

PROPOSITION 4.47 *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ be a SUP(T) derivation, where $N_0$ is obtained by basification and abstraction from a BSHE(GBST) clause set. Then*[1] *$n_{\mathcal{V}}(N_i) \leq n_{\mathcal{V}}(N_0)$, for every $i \geq 0$.*

COROLLARY 4.48 *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$, *where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base*

[1]We write $n_{\mathcal{V}}(M)$ to denote the overall number of ground base terms, to each of which there is an assignment of a variable via an equation in the constraint of a clause in the set of abstracted clauses $M$ (see Definition 4.22, page 147).

*and body signatures, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.*

*Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ be a SUP(T) derivation, where $N_0$ is obtained by basification and abstraction from a BSHE(GBST) clause set. If the MPG transformation*[1] *is applied to every clause in the derivation, then the number of all base variables in any clause in the derivation is at most*[2] *$n_{\mathcal{V}}(N_0)$:*

$$|\,var(C) \cap \mathcal{X}| \leq n_{\mathcal{V}}(N_0),$$

*for any clause $C \in N_i$, $i \geq 0$.*

However, we have to take a special care of *free* variables. For this reason, we introduce another instance of the hierarchic lexicographic path ordering, Definition 4.29, which limits the number of non-base variables in a derived clause and ensures a bounded literal growth. Since all free function symbols of non-zero arity range into a base sort $\mathsf{S} \in \mathcal{S}$, any term of a free sort $\mathsf{S}'' \in \mathcal{S}''$ occurring in the input clause set $N$ is either a non-base variable $x \in \mathcal{X}''$, or a free constant $a \in \Omega''$. Clearly, this property is preserved by basification, abstraction, and any inference rule application, and holds thus for all $N_i$ in $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$. Therefore, during the derivation a non-base variable $x \in \mathcal{X}''$ can be unified either with another non-base variable $y \in \mathcal{X}''$, or with a free constant $a \in \Omega''$. If a non-base variable is unified with a free constant, the number of free operator symbol occurrences in the superposed term can actually grow. To cope with this, we need to count in addition the number of non-base variable occurrences. Here we exploit an instance of HLPO($\gamma$), where for $\gamma$ we use the function $\phi$ defined as follows.

DEFINITION 4.49 ► Function $\phi$

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.

We define a function $\phi : T_{\Omega'}(\mathcal{X}') \to \mathbb{N}$, which given a $\Sigma'$-term $t \in T_{\Omega'}(\mathcal{X}')$ returns the number of non-base symbol's occurrences in $t$, as follows:

$$\phi(t) \overset{\text{def}}{=} \big|\{p \in \rho(t) \mid top(t/p) \in \Omega'' \cup \mathcal{X}''\}\big|.$$

The function $\phi$ extends onto atoms and literals as

$$\phi(s \mathrel{\dot{\approx}} t) \overset{\text{def}}{=} \phi(s) + \phi(t),$$

where $s, t \in T_{\Omega'}(\mathcal{X}')$ and $\dot{\approx} \in \{\approx, \not\approx\}$. ■

The function $\phi$ counts also occurrences of non-base variables, in contrast to the function[3] $\omega$, which counts only free operator symbol occurrences. The resulting instance $\succ_{\mathrm{H}(\phi)}$ of HLPO is a reduction ordering for the BSHE(GBST) fragment, with respect to simple substitutions.

PROPOSITION 4.50 ► *Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where* $\Sigma = (\mathcal{S}, \Omega)$ *and* $\Sigma' = (\mathcal{S}', \Omega')$ *are the base*

[1]Please, recall Definition 4.27 of the MPG transformation, page 149.
[2]Please, recall Definition 4.22 of the number $n_{\mathcal{V}}(N_0)$, page 147.
[3]The function $\omega$ is introduced in Definition 4.30, page 151.

*and body signatures, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.*

*If all non-constant function symbols in $\Omega''$ range into base sorts, then the hierarchic lexicographic path ordering $\succ_{\mathrm{H}(\phi)}$ augmenting the function $\phi$ is a reduction ordering for $\Sigma'$-terms $T_{\Omega'}(\mathcal{X}')$ with respect to simple substitutions.*

◄ PROOF

Let $t, s, r \in T_{\Omega'}(\mathcal{X}')$ be arbitrary $\Sigma'$-terms, $r = f(r_1, \ldots, r_n)$, where $n \geq 1$. Argumentation for the properties irreflexivity, transitivity, well-foundedness, context compatibility, and the subterm property is exactly the same as in Proposition 4.31 with the only difference being in using the function $\phi$ in place of $\omega$ (the presence of non-base variables does not harm the arguments of Proposition 4.31 regarding the listed properties).

The proof of stability under simple substitutions is slightly different—due to the presence of non-base variables. Assume $t \succ_{\mathrm{H}(\phi)} s$. Let $\sigma$ be a simple substitution. Any base variable $x \in var(t) \cap \mathcal{X}$—analogously for $s$—can be mapped by $\sigma$ only to a base term, and since base terms may contain only base symbols, we learn $\phi(x\sigma) = 0 = \phi(x)$. As the only non-variable terms of a free sort are constants, any non-base variable $y \in var(t) \cap \mathcal{X}''$ can be mapped by $\sigma$ either to a non-base variable or a free constant, hence $\phi(y\sigma) = 1 = \phi(y)$. Thus, an application of $\sigma$ to variables of $t$ does not change the number of occurrences of free symbols, and, therefore, $\phi(t) = \phi(t\sigma)$; analogously $\phi(s) = \phi(s\sigma)$. If $\phi(t) > \phi(s)$, then $\phi(t\sigma) > \phi(s\sigma)$ as well, hence $t\sigma \succ_{\mathrm{H}(\phi)} s\sigma$. If $\phi(t) = \phi(s)$, then $\phi(t\sigma) = \phi(s\sigma)$, and it must also hold that $t \succ_{\mathrm{lpo}} s$. Since $\succ_{\mathrm{lpo}}$ is stable under substitutions, we know $t\sigma \succ_{\mathrm{lpo}} s\sigma$, consequently, $t\sigma \succ_{\mathrm{H}(\phi)} s\sigma$. So, $\succ_{\mathrm{H}(\phi)}$ augmenting $\phi$ is stable under simple substitutions. ■

We redefine the function $n_{\mathcal{F}}$, Definition 4.32, in the following way.

◄ DEFINITION 4.51
Number $n_{\mathcal{F}}$

Assume $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ is a hierarchic specification with the base specification $\mathsf{Sp} = (\Sigma, \mathscr{C})$ and the body $\mathsf{Sp}' = (\Sigma', Ax')$.

Given a set $M$ of abstracted clauses, we write $n_{\mathcal{F}}(M)$ to denote the maximum number of occurrences of free operator symbols and non-base variables in a free literal among all clauses in $M$:

$$n_{\mathcal{F}}(M) \overset{\text{def}}{=} max_{C \in M}\{\phi(L) \mid L \in (\Gamma \to \Delta), C = \Lambda \parallel \Gamma \to \Delta\}.$$

■

◄ PROPOSITION 4.52

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ be a SUP(T) derivation, where $N_0$ is obtained by basification and abstraction from a BSHE(GBST) clause set. If $\succ_{\mathrm{H}(\phi)}$ is the underlying ordering, then for every free literal $L$ in any clause $C \in N_i$, $i \geq 0$, the number $\phi(L)$ of occurrences of free operator symbols and non-base variables is at most $n_{\mathcal{F}}(N_0)$.*

◄ PROOF

Analogous to Proposition 4.33. ■

◄ LEMMA 4.53

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$*, where $\Sigma = (\mathcal{S}, \Omega)$ and $\Sigma' = (\mathcal{S}', \Omega')$ are the base and body signatures, respectively. Let $\mathcal{S}'' = \mathcal{S}' \setminus \mathcal{S}$ and $\Omega'' = \Omega' \setminus \Omega$ be the enrichment* *sorts and operators, respectively. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}''$ be the underlying variable set consisting of base and non-base variables, respectively.*

*Let $N_0$ be obtained by basification and abstraction from a BSHE(GBST) finite clause set. Then any SUP(T) derivation $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ with eager selection, MPG transformation, and reduction ordering $\succ_{\mathrm{H}(\phi)}$ is terminating.*

PROOF ► According to Proposition 4.42, any literal in the constraint of a clause in the derivation is either a (dis)equation $s_1 \mathrel{\dot{\approx}} s_2$ between some ground base terms $s_1, s_2 \in T_\Omega$, where $\dot{\approx} \in \{\approx, \not\approx\}$, or an assignment $x \approx t$ of a base variable $x \in \mathcal{X}$ to a ground base term $t \in T_\Omega$. The terms $s_1, s_2, t$ are all inherited from $N_0$, meaning that no new base term is produced by an inference. According to Proposition 4.48, a clause in the derivation may contain at most $n_{\mathcal{V}}(N_0)$ different base variables. For finitely many different terms $s_1, s_2, t$ and a limited number of variables, there can be only finitely many different literals of the above form, therefore there can be only finitely many different constraints $\Lambda$, up to variable renaming.

According to Corollary 4.21, any clause in the derivation may contain at most $n_{\mathcal{L}}(N_0)$ free literals, each of which, according to Proposition 4.52, may have at most $n_{\mathcal{F}}(N_0)$ occurrences of free operator symbols and non-base variables, limiting thus the size of free literals. According to Corollary 4.48, the number of base variables is also limited by the number $n_{\mathcal{V}}(N_0)$. For a finite set $\Omega''$ of free operator symbols occurring in $N_0$, a limited number and size of literals, and limited number of variables, there exist only finitely many different free parts $\Gamma \to \Delta$, up to variable renaming.

Combining the two observations above, we conclude that there can be only finitely many different clauses $\Lambda \parallel \Gamma \to \Delta$, up to variable renaming, in the derivation $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$. Thus, any inference from $N_i$, for some $i \geq 0$, with a conclusion $C$ that can be matched to a clause $D \in N_i$ by variable renaming, is redundant. After sufficiently (and finitely) many clauses have been derived, any further inference is thus redundant. Since only irredundant inferences have to be performed, $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ is terminating. ■

### 4.3.5 Decidability

According to Proposition 4.42, every base variable $x \in \mathcal{X}$, occurring in a clause in a SUP(T) derivation $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ from a clause set $N_0$ obtained by basification and abstraction from an input BSHE(GBST) clause set, is assigned to is assigned to a ground base term $t \in T_\Omega$, if variables assignments grounding is applied to every clause in the derivation. Any non-ground base clause in the derivation can be thus grounded by propagation of all assignments $x \approx t$. The term $t$ may contain base parameters introduced at the basification step. The parameters are essentially base variables existentially quantified on the top of the overall clause set. Thus, an application of the Constraint Refutation rule[1]

$$\mathcal{I}\,\frac{C_1[\mathsf{a}^1_1,\ldots,\mathsf{a}^1_{m_1}] \quad \ldots \quad C_n[\mathsf{a}^n_1,\ldots,\mathsf{a}^n_{m_n}]}{\square}$$

where $\mathsf{a}^1_1,\ldots,\mathsf{a}^n_{m_n}$ are all base parameters occurring in the premises $C_1,\ldots,C_n$ (any two parameters $\mathsf{a}^i_j$ and $\mathsf{a}^{i'}_{j'}$ do not have to be necessarily different), reduces to

[1]The Constraint Refutation rule is introduced in Definition 3.31, page 48.

checking satisfiability of a base formula

$$\exists\, y_1^1,\ldots,y_{m_n}^n : C_1[y_1^1,\ldots,y_{m_1}^1] \wedge \ldots \wedge C_n[y_1^n,\ldots,y_{m_n}^n],$$

in which the base parameters are replaced with respective fresh base variables $y_1^1,\ldots,y_{m_n}^n \in \mathcal{X}$, which are all existentially quantified the theory domain. Similarly to the ground FOL(T) case, the only requirement we impose on the theory T for the BSHE(GBST) class is the decidability of its existential fragment.

◄ THEOREM 4.54
BSHE(GBST) Decidability

*Assume* $\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}')$ *is a hierarchic specification with the base specification* $\mathsf{Sp} = (\Sigma, \mathscr{C})$ *and the body* $\mathsf{Sp}' = (\Sigma', Ax')$.

*If the* $\exists^*$ *fragment is decidable in* $\mathsf{Sp}$*, then the SUP(T) calculus with eager selection, MPG transformation, and reduction ordering* $\succ_{\mathrm{H}(\phi)}$ *augmenting the function* $\phi$ *is a decision procedure for the BSHE(GBST) class.*

◄ PROOF

Basification, presented in Section 4.2.1, transforms a finite set $N$ of BSHE(GBST) clauses into a finite equisatisfiable clause set $N'$, such that every model of $N'$ is a model of $N$, Lemma 4.40.

Assume $N_0$ is a clause set obtained from $N'$ by abstraction. As $N$ is a Horn clause set, $N_0$ is so as well. Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ be a SUP(T) derivation with eager selection, MPG transformation, and the underlying ordering $\succ_{\mathrm{H}(\phi)}$ augmenting the function $\phi$. According to Lemma 4.53, the derivation is terminating. Let $N_k$ be the last clause set in the derivation, for some $k \geq 0$. The set $N_k$ is the limit of the derivation. Moreover, $N_k$ is finite. By Lemma 3.9, the clause set $N_k$ is saturated. If $\square \notin N_k$, then the set $N_k \cap Cl_\Sigma$ of all base clauses within $N_k$ is theory-consistent. Indeed, suppose for the sake of contradiction that $N_k \cap Cl_\Sigma$ is theory-inconsistent. As $N_k$ is finite, the set $N_k \cap Cl_\Sigma$ is also finite. Therefore, a constraint refutation inference derives $\square$ from $N_k \cap Cl_\Sigma$. Since $N_k$ is saturated, the inference must be redundant with respect to $\mathcal{R}^{\mathcal{H}}$, which holds, according to Definition 3.64 of $\mathcal{R}^{\mathcal{H}}$, if and only if an empty clause $\square$ is already in $N_k$, a contradiction. Let $\mathcal{A} \in \mathscr{C}$ be an arbitrary base algebra satisfying the set $N_k \cap Cl_\Sigma$. By Lemma 4.46, the clause set $N_k$ has a hierarchic model. This model is also a model of $N_0$. If $\square \in N_k$, then $N_k$ is unsatisfiable due to soundness of SUP(T).

As the base theory supports a decision procedure for the $\exists^*$ fragment, Constraint Refutation can be decided, therefore satisfiability $N_0$ can be determined, hence satisfiability of $N'$ and $N$ can be determined as well. Thus, the SUP(T) calculus with eager selection, MPG transformation, and reduction ordering $\succ_{\mathrm{H}(\phi)}$ is a decision procedure for the ground clause FOL(T) fragment. ■

Similarly to the ground FOL(T) case, where we use the reduction ordering $\succ_{\mathrm{H}(\omega)}$ augmenting the function $\omega$, in the BSHE(GBST) case the usage of the hierarchic LPO $\succ_{\mathrm{H}(\phi)}$ augmenting the function $\phi$ also yields an upper bound on the maximum size of free literals and, hence, guarantees the existence of only finitely many different free literals over a finite set $\Omega''$ of free operator symbols and a limited number of variables. Thus, even if the length of the free part of a clause not limited, there can be only finitely many different free parts $\Gamma \to \Delta$, up to variable renaming and duplication of literals. This observation supports an alternative strategy which does not require the input clause set to consist only of Horn clauses, but what is required in return is an additional Hierarchic Factoring rule, which is necessary to guarantee hierarchic model existence if non-Horn clauses admitted.

For the sake of simplicity, we have restricted our attention to a *Horn* clause fragment of FOL(T), where every non-constant function symbol from the underlying FOL signature ranges into the sort of the theory T and all base sort terms are ground. Actually, SUP(T) can decide the general clause fragment as well, because since in this case the only function symbols ranging into the free sort are constants, every non-Horn clause can be split into Horn clauses by instantiating every subsequent occurrence of the same variable by the free constants, reducing the original non-Horn problem to a Horn one.

### 4.3.6 Relation to Weak Abstraction. Completeness on GBT Class

We finish the discussion of SUP(T) decidability by addressing SUP(T) completeness for the ground base sort terms fragment GBT [BW13]. The GBT class comprises all $\Sigma'$-clauses, in which all terms of a base sort are ground. The GBT class is thus larger than both the ground and the BSHE(GBST) fragments. Although SUP(T) is, evidently, not a decision procedure for the GBT class[1], the hierarchic calculus is still strong enough to guarantee completeness for the GBT fragment.

Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ be fair SUP(T) derivation with exhaustive application of the MPG transformation, where $N_0 = N$ is obtained by basification and "full" abstraction[2] from a set of $\Sigma'$-clauses in which all base sort terms are ground. Assume the limit $N_\infty$ of the derivation does not contain an empty clause $\square$. By Lemma 3.9, the limit $N_\infty$ is saturated.

According to Proposition 4.11, any literal in the constraint of a clause in the derivation is either a (dis)equation $s_1 \mathrel{\dot{\approx}} s_2$ between some ground base terms $s_1, s_2 \in T_\Omega$, where $\dot{\approx} \in \{\approx, \not\approx\}$, or an assignment $x \approx t$ of a base variable $x \in \mathcal{X}$ to a ground base term $t \in T_\Omega$. In the proof of Proposition 4.11 we have shown that the terms $s_1, s_2, t$ are all inherited from $N_0$, meaning that no new base term is produced by an inference. In Proposition 4.28, we have shown that a clause in the derivation may contain at most $n_{\mathcal{V}}(N_0)$ different base variables. For finitely many different terms $s_1, s_2, t$ and a limited number of variables, there can be only finitely many different literals of the above form, up to variable renaming, therefore there can be only finitely many different constraints $\Lambda$, up to variable renaming and removal of duplicate literals. This implies that only *finitely many* different *base clauses* can be produced in $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$, up to very straightforward redundancy deletion. Consequently, the limit $N_\infty$ of the derivation may contain only finitely many different base clauses, say $C_1, \ldots, C_n$, for some $n \geq 0$, where $C_i = (\Lambda_i \parallel \rightarrow)$, for each $i \in \{1, \ldots, n\}$. Therefore, there exists a base algebra $\mathcal{A} \in \mathscr{C}$ that satisfies every clause $C_i$. Indeed, suppose for the sake of contradiction, that the set $\{C_1, \ldots, C_n\}$ is satisfiable by no base algebra in $\mathscr{C}$, i.e. $C_1, \ldots, C_n \models_{\mathscr{C}} \bot$. Hence, there is a Constraint

[1] If the background theory T is empty, SUP(T) reduces to the standard superposition calculus SUP for general FOL clauses and GBT reduces to the general FOL fragment, which is known to be semi-decidable.

[2] We refer to the abstraction algorithm introduced in Section 3.2 as *"full" abstraction*, in contrast to *weak abstraction* of Baumgartner and Waldmann [BW13].

Refutation inference, Definition 3.31:

$$\mathcal{I}\,\frac{\Lambda_1 \parallel \rightarrow \qquad \ldots \qquad \Lambda_n \parallel \rightarrow}{\square}$$

As $N_\infty$ is saturated, the inference is redundant. According to Definition 3.64 of the hierarchic redundancy criterion $\mathcal{R}^{\mathcal{H}}$, a Constraint Refutation inference is redundant if and only if an empty clause is present in the clause set, a contradiction.

According to Proposition 4.8, the clause set $N$ is locally sufficiently complete (see Definition 3.84. By Theorem 3.93, the set $N$ has a hierarchic model. This implies refutational completeness of the SUP(T) calculus combined with basification for all GBT clause sets.

## 4.4 Application: Reasoning in Ontologies with Arithmetical Facts

One of possible applications of the above decidability result is reasoning (saturation and querying) in ontologies with arithmetical facts, such as time stamps, size/amount information, etc. Suda, Weidenbach, and Wischnewski have shown in [SWW10, Wis12] that the ontology YAGO[1] can be expressed in terms of the Bernays-Schönfinkel Horn class with equality, abbreviated BSHE, and presented a variant of superposition that decides the class. Typical examples of clauses in the representation of the ontology are (the examples are taken from [SWW10]):

| | |
|---|---|
| $\rightarrow \mathit{bornIn}(\mathit{AlbertEinstein}, \mathit{Ulm})$ | // for "Albert Einstein was born in Ulm" |
| $\rightarrow \mathit{human}(\mathit{AngelaMerkel})$ | // for "Angela Merkel is a human" |
| $\mathit{human}(x) \rightarrow \mathit{mammal}(x)$ | // for "Every human is a mammal" |

The ontology can be queried with questions like "who are the people who died in New York at the same place where their children were born?", which is formally encoded as the following conjecture

$$\exists\, x, y, z. \mathit{diedIn}(x, y) \wedge \mathit{hasChild}(x, z) \wedge \mathit{bornIn}(z, y) \wedge \mathit{locatedIn}(y, \mathit{NewYork})$$

which after negating becomes the clause

$$\mathit{diedIn}(x, y), \mathit{hasChild}(x, z), \mathit{bornIn}(z, y), \mathit{locatedIn}(y, \mathit{NewYork}) \rightarrow$$

with all variables universally quantified.

If the ontology is further extended with arithmetical information, such as, for instance:

| | |
|---|---|
| $\rightarrow \mathit{overInY}(\mathit{WWII}, 1945)$ | // for "The World War II ended in 1945" |
| $\rightarrow \mathit{diedInY}(\mathit{KarlTheGreat}, 814)$ | // for "Karl The Great died in 814" |
| $\rightarrow \mathit{gdp}(\mathit{Russia}, 2011, 2.4 \cdot 10^{12})$ | // for "The GDP[1] of Russia was \$2.4 trillion in 2011" |
| $\rightarrow \mathit{population}(\mathit{USA}, 2012, 313.8 \cdot 10^{6})$ | // for "The population of USA was 313.8 million in 2012" |

then it can be addressed with queries like:

- "Was the GDP of Germany steadily growing every year by at least 3% in 2008-2010?", which is encoded as the conjecture:

$$\forall\, x_1, x_2, y_1, y_2. \mathit{gdp}(\mathit{Germany}, y_1, x_1), \mathit{gdp}(\mathit{Germany}, y_2, x_2),\\ 2008 \le y_2, y_2 \le 2010, y_2 = y_1 + 1 \;\rightarrow\; x_2 \ge 1.03 \cdot x_1$$

- "Was the GDP of Germany at least 3% higher than that of any other country in Europe in the years 2008-2010?":

$$\forall\, u. \forall\, x_1, x_2, y. \mathit{gdp}(\mathit{Germany}, y, x_1), \mathit{gdp}(u, y, x_2), \mathit{locatedIn}(u, \mathit{Europe}),\\ u \not\approx \mathit{Germany}, 2008 \le y, y \le 2010 \;\rightarrow\; x_1 \ge 1.03 \cdot x_2$$

[1]YAGO (Yet Another Great Ontology) is the first automatically retrieved ontology out of Wikipedia and WordNet with accuracy of about 97% [SKW07]; a well-known ontology in the information retrieval community.

[1]GDP – Gross domestic product.

If $\mathcal{O}$ denotes the clause representation of the ontology, and $F$ a conjecture above, then the set $\mathcal{O} \cup \neg F$ is in the BSHE(GBST) class. If the property $F$ holds in the ontology, i.e. $F$ follows from $\mathcal{O}$, then the clause set $\mathcal{O} \cup \neg F$ is unsatisfiable, and it is satisfiable otherwise. The background theory T is the theory of linear arithmetic, for which satisfiability of the existential closure of a quantifier-free formula is decidable. According to Theorem 4.54, satisfiability of $\mathcal{O} \cup \neg F$ is decidable by SUP(T) underlain by eager selection and reduction ordering $\succ_{\mathrm{H}(\phi)}$ augmenting $\phi$.

Now the query answering mechanisms presented in [WW12] can actually be extended according to the above decidability result to yield also a decision procedure for the extended language.

CHAPTER 5

# SUP(LA): Superposition Modulo Linear Arithmetic

In this chapter we instantiate and refine the hierarchic superposition calculus SUP(T) for the theory of linear arithmetic LA. In Section 5.1 we define the hierarchic specification of the combination of FOL and LA, and exhibit the essential specialties of SUP(LA). In Section 5.2, we present a solution to the issues regarding application of SUP(LA) to the hierarchic FOL(LA) combination that are turned there into effective procedures via a mapping to Linear Programming tasks. Key aspects of the overall implementation of SPASS(LA) are provided in Section 5.3. In Section 5.4 we present experimental results of applying SPASS(LA) to reachability problems of transition systems and discuss application of SUP(T) to container data structure axiomatizations using the example of lists.
The essential parts of this chapter have been presented in [AKW09a, AKW09b]

## 5.1 Introduction

In this chapter we instantiate and refine the hierarchic superposition calculus introduced in Chapter 3 for the theory of linear arithmetic. In particular, we refine the reduction rules definitions and develop new effective algorithms for redundancy elimination taking the linear arithmetic theory into account.

### 5.1.1 Hierarchic Specification of FOL(LA)

First, we define a hierarchic specification and its ingredients, base specification and body, for the hierarchic combination FOL(LA) of first-order theory FOL with the theory of linear arithmetic LA. The notions (re)defined here are valid from now on till the end of Chapter 5.

We write $\Sigma$ to denote the signature of LA defined as follows:

$$\Sigma = (\mathsf{Q}, \Omega)$$

where:

- $\Omega$ is the set of arithmetic operators given as:

$$\begin{aligned} \Omega = \{&+, -, \times, && \text{// arithmetic functions}\\ &\leq, <, \approx, >, \geq, && \text{// arithmetic relations}\\ &\mathbb{Q}\,\} && \text{// rational numbers (numeric constants)} \end{aligned}$$

- $\mathsf{Q}$ the only arithmetic sort,

with the underlying set $\mathcal{X}$ of all variables of the arithmetic sort $\mathsf{Q}$.

We set the class $\mathscr{C}$ of base algebras to consist of the standard model $\mathcal{M}$ of linear arithmetic (and all algebras isomorphic to it). For the sake of simplicity, we simply write $\mathcal{M}$ in place of $\mathscr{C}$. The model $\mathcal{M}$ maps the arithmetic sort $\mathsf{Q}$ to the set $\mathbb{Q}$ of all rational numbers[1], i.e. $\mathcal{M}(\mathsf{Q}) = \mathbb{Q}$, and interprets the functions and relations in $\Omega$ as intended. Clearly, the algebra $\mathcal{M}$ is term-generated (as there is a one-to-one correspondence of $\mathcal{M}$ universe's elements to numeric constants in $\mathbb{Q} \subset T_\Omega$). We write $\mathsf{Sp}$ to denote the base specification of linear arithmetic, defined as

$$\mathsf{Sp} = (\Sigma, \mathcal{M})$$

The operators and the sorts of $\Sigma$ are further extended with free (uninterpreted) operators to $\Omega' \supseteq \Omega$ and $\mathcal{S}' \ni \mathsf{Q}$, respectively; the set of arithmetic variables is extended with non-base variables to $\mathcal{X}' \supseteq \mathcal{X}$. Recall, that every non-base variable in $\mathcal{X}'' = \mathcal{X}' \setminus \mathcal{X}$, if any, is of a free sort $\mathsf{S}'' \in \mathcal{S}' \setminus \mathsf{Q}$, whereas free operator symbols may have arguments of the arithmetic sort or range into it. These constitute a FOL(LA) signature $\Sigma'$:

$$\Sigma' = (\mathcal{S}', \Omega')$$

underlied by the set of variables $\mathcal{X}'$.

Let $Ax'$ be a set of formulae built over $\Sigma'$. The pair

$$\mathsf{Sp}' = (\Sigma', Ax')$$

[1]We ambiguously write $\mathbb{Q}$ to denote the subset of $\Omega$ consisting of all arithmetic constants (syntactical objects) and the set of rationals (the universe of $\mathcal{M}$).

defines the body of a hierarchic FOL(LA) specification

$$\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}').$$

As usual, we call operators in $\Omega'' = \Omega' \setminus \Omega$, sorts in $\mathcal{S}'' = \mathcal{S}' \setminus \mathsf{Q}$, and axioms $Ax'$ the enrichment. In contrast to the LA specification $\mathsf{Sp}$, the enrichment

$$(\mathcal{S}'', \Omega'', Ax')$$

is not fixed beforehand and is defined by an input clause set $N$: the symbols in $N$ that do not otherwise occur in $\Sigma$ define $\mathcal{S}''$ and $\Omega''$. For the sake of simplicity, in the rest we set $Ax' = \emptyset$, which, according to the discussion of alternative formulations of refutational theorem proving for hierarchic theories (see Section 3.2), is not a limitation, but rather a more convenient way of arguing in the context of the combination of FOL with NLA.

Thus, refutational theorem proving for the hierarchic combination FOL(LA) is aimed at answering a question, whether a given clause set $N$ over the FOL(LA) signature $\Sigma'$ is inconsistent relative to $\mathcal{M}$, i.e. is there no model of $N$ whose restriction to the LA signature $\Sigma$ is (isomorphic to) $\mathcal{M}$. Since $\mathcal{M}$ is the only model of LA, we write $\models_{\mathcal{M}}$ to denote the entailment relative to the theory of LA (please, recall Definition 3.25 of the entailment relative to a base theory).

◄ EXAMPLE 5.1

The following is a typical abstracted clause over the hierarchic specification $\mathsf{HSp}$:

$$x \approx z_1 + 40, y \approx z_1 - 3.5, z_2 \leq 60 \parallel T(f(y), y') \to S(x, x'),$$

where $x, y, z_1, z_2 \in \mathcal{X}$ are LA variables; $x', y' \in \mathcal{X}''$ non-base variables; $T, S \in \Omega''$ free predicate operators; $f \in \Omega''$ a free function symbol ranging from rationals into a free sort. ■

## 5.1.2 SUP(LA) Application Issues

Next, we consider the most important peculiarities of the application of the hierarchic calculus to FOL(LA) clauses. Application of the hierarchic versions of the standard rules (Hierarchic Equality Resolution, Hierarchic Superposition, etc..) to FOL(LA) clauses does not require LA reasoning, because:

– the rules apply independently from premises' constraints (as the rules' conditions are stipulated exclusively by the free parts of premises), and

– the conclusion's constraint is simply an instance of the conjunction of the premises' constraints, which can be easily computed by concatenating the constraints and applying the free part unifier; since the unifier is simple, its application reduces to identifying a subset of the constraints' variables.

In contrast, the Constraint Refutation rule and hierarchic reduction rules are further subjected to restrictions on the premises' constraints, that require performing calculations with respect to the background theory. Recall the Constraint Refutation rule, Definition 3.31:

$$\mathcal{I}\frac{\Lambda_1 \parallel \to \quad \ldots \quad \Lambda_n \parallel \to}{\square}$$

where $(\Lambda_1 \parallel \to), \ldots, (\Lambda_n \parallel \to) \models_{\mathscr{C}} \bot$. In Section 3.3 we have shown that the condition of the rules holds if and only if the existential closure $\exists \vec{x}^i . \bigwedge \Lambda_i$ of the constraint of *at least one base clause* among $C_1 = (\Lambda_1 \parallel \to)$, ..., $C_n = (\Lambda_n \parallel \to)$, where

$\vec{x}^i = var(\Lambda_i)$, is valid in the base theory (satisfiable by every base algebra). For the LA theory considered here, which is compact and where the clauses, e.g., do not share parameters, it is sufficient to consider the case $n = 1$, thereby reducing the condition of the rule to simply testing whether the existential closure $\exists \vec{x}^1 . \bigwedge \Lambda_1$ is satisfiable in the single model $\mathcal{M}$ of LA.

Consider the Hierarchic Tautology Deletion rule, Definition 3.96:

$$\mathcal{R}\frac{\Lambda \parallel \Gamma \to \Delta}{}$$

if

(i) $\models \Gamma \to \Delta$, or

(ii) $\exists \vec{x} . \bigwedge \Lambda \models_{\mathscr{C}} \bot$, for $\vec{x} = var(\Lambda)$.

According to the second condition the existential closure of the base part of the premise has to be unsatisfiable in the base theory, which for the LA theory case reduces to checking unsatisfiability/validity of $\exists \vec{x} . \bigwedge \Lambda$ in the single model $\mathcal{M}$ of LA.

Consider the Hierarchic Subsumption Deletion rule, Definition 3.97:

$$\mathcal{R}\frac{\Lambda_1 \parallel \Gamma_1 \to \Delta_1 \qquad \Lambda_2 \parallel \Gamma_2 \to \Delta_2}{\Lambda_1 \parallel \Gamma_1 \to \Delta_1}$$

where, for a simple matcher $\sigma$,

(i) $\Gamma_1\sigma \subseteq \Gamma_2$, $\Delta_1\sigma \subseteq \Delta_2$,

(ii) $\models_{\mathscr{C}} \forall \vec{x} . \exists \vec{y} . (\bigwedge \Lambda_2 \to \bigwedge \Lambda_1\sigma)$, for $\vec{x} = var(C_2) \cap \mathcal{X}$ and $\vec{y} = var(\Lambda_1\sigma) \setminus var(C_2)$,

(iii) $(\Lambda_2 \parallel \Gamma_2 \to \Delta_2) \neq \square$.

The third condition of the Hierarchic Subsumption Deletion rule is stipulated by the fact, that the rule missing the condition would allow subsuming an empty clause by an unsatisfiable base clause. The condition is intended rather for *theoretical purposes* and does not make much sense *in practice* as all/most theorem provers stop executing whenever an empty clause is derived. Note that $y \in var(\Lambda_1\sigma) \setminus var(C_2)$ implies $y \notin (var(\Gamma_1) \cup var(\Delta_1))$. If the base specification enables quantifier elimination, then every such $y$ could be eliminated in $\Lambda_1 \parallel \Gamma_1 \to \Delta_1$ and the encompassment condition (Condition (ii)) of the rule would become

$$\models_{\mathscr{C}} \forall \vec{x} . (\bigwedge \Lambda_2 \to \bigwedge \Lambda_1\sigma),$$

for $\vec{x} = var(C_2) \cap \mathcal{X} \supseteq var(\Lambda_1\sigma)$. However, for the theory of linear arithmetic elimination of variables $\vec{y} = var(\Lambda_1\sigma) \setminus var(C_2)$ is in the worst case *exponential.* In Section 5.2 we propose a polynomial transformation to linear programming for finding a simple theory matcher $\tau$ with $dom(\tau) = var(\Lambda_1\delta) \setminus var(C_2)$ and $cdom(\tau) \subseteq var(\Lambda_2) \subseteq var(C_2)$, that maps the variables $\vec{y} = var(\Lambda_1\sigma) \setminus var(C_2)$ to base terms, which are linear arithmetic combinations over variables from $\Lambda_2$, such that

(5.1) ► $$\models_{\mathscr{C}} \forall \vec{x} . (\bigwedge \Lambda_2 \to \bigwedge \Lambda_1\sigma\tau),$$

for $\vec{x} = var(C_2) \cap \mathcal{X} \supseteq var(\Lambda_1\sigma\tau)$. Such matcher $\tau$ is called an ***encompassment matcher***. Existence of an encompassment matcher $\tau$ is *sufficient* to satisfy the encompassment condition of the rule and can be determined in case of LA in (*weakly*) *polynomial* time in the size of the two constraints $\Lambda_1$, $\Lambda_2$.

## 5.2 Constraint Solving

As discussed in Section 5.1, we have to provide procedures for (un)satisfiability and for an implication test for linear arithmetic, potentially modulo an encompassment matcher getting rid of extra constraint variables. In this section, we solve the intended problems leveraging Linear Programming techniques.

### 5.2.1 Basic Notions

In the following, we use the standard notation from linear algebra and linear programming theory and assume that the reader is familiar with these topics. A standard reference is [Sch89].

***Vectors*** denoted with $\vec{a}$, $\vec{b}$, $\vec{c}$ or $\vec{d}$ refer to vectors of rational numbers, vectors $\vec{x}$, $\vec{y}$ or $\vec{z}$ denote vectors of variables; $\vec{p}$ and $\vec{\beta}$ may stand for either a rational or a variable vector. $\vec{a}^T$ denotes the ***transposed vector*** of $\vec{a}$, $a_i$ stands for the $i$-th element of $\vec{a}$. ***Matrices*** are denoted with capital letters like $A$, $B$, $G$, $H$, $S$, $T$ and $P$, where $A$, $B$, $H$ and $G$ denote matrices of rational values and $P$, $S$, $T$ may stand for matrices of either rational values or variables; $A_i$ is used to denote the $i$-th row of a matrix $A$. Given an arbitrary variable vector $\vec{x}$, we assume that all entries $x_i$ of $\vec{x}$ are pairwise distinct; the same concerns matrices of variables.

We use the following notation to denote *combined* vectors and matrices:

– $\vec{a} = \binom{\vec{a}'}{\vec{a}''}$ for a combined vector $\vec{a}$ consisting of elements of the vector $\vec{a}'$ succeeded by elements of the vector $\vec{a}''$;

– $A = \binom{A'}{A''}$ for a combined matrix $A$ consisting of rows of the matrix $A'$ succeeded by rows of the matrix $A''$.

A ground substitution $\nu : \mathcal{X} \to \mathbb{Q}$ mapping arithmetic variables $\mathcal{X}$ to rationals $\mathbb{Q}$ is called a ***base assignment***. Given a vector $\vec{t} = (t_1, \ldots, t_n)^T$, the result of applying a base assignment $\nu$ to $\vec{t}$ is the vector $\vec{t}\nu = (t_1\nu, \ldots, t_n\nu)^T$. We write $\vec{a} \circ b$ to show that $a_i \circ b$ for every entry $a_i$ of $\vec{a}$, where $\circ \in \{\leq, <, \approx, >, \geq\}$.

◄ DEFINITION 5.2
Affine Substitution

A simple substitution $\tau$ is called an ***affine substitution***, if every term in its image is a linear combination of some variables and a rational constant:

$$y\tau \overset{\text{def}}{=} \underbrace{\alpha_1 x_1 + \cdots + \alpha_n x_n}_{\text{lin. combination}} + \underbrace{\beta}_{\text{const.}}$$

for every $y \in dom(\tau)$ and some $\alpha_i, \beta \in \mathbb{Q}$ and $x_i \in \mathcal{X}$, for all $i \in \{1, \ldots, n\}$, $n \geq 1$. ■

**Systems of Linear Inequations**

A ***linear (in)equation*** is of the form $\vec{a}^T\vec{x} \circ c$ with ***coefficients*** $\vec{a} \in \mathbb{Q}^n$, ***right-hand-side*** $c \in \mathbb{Q}$, and *sense* $\circ \in \{\leq, <, \approx, >, \geq\}$. We can equivalently rewrite $\vec{a}^T\vec{x} \approx c$ to $\vec{a}^T\vec{x} \leq c \wedge \vec{a}^T\vec{x} \geq c$ and $\vec{a}^T\vec{x} \geq c$ to $-\vec{a}^T\vec{x} \leq -c$.

A ***system of linear (in)equations*** (***SLI*** – for short) is the conjunction of a set of linear (in)equations, denoted by

$$A\vec{x} \bullet \vec{c},$$

where $A \in \mathbb{Q}^{m \times n}$ is a rational matrix with $m$ rows $A_i \in \mathbb{Q}^n$, $\vec{c} \in \mathbb{Q}^m$ a vector of right-hand-sides, and $\bullet$ a vector of $m$ senses $\bullet_i \in \{\leq, <, \approx, >, \geq\}$. The ***feasible region*** of an SLI $\Lambda = A\vec{x} \bullet \vec{c}$ is the set of all points satisfying every (in)equation in the system, denoted by

$$Feas(\Lambda) \overset{\text{def}}{=} \{\vec{x}v \mid v : \mathcal{X} \to \mathbb{Q}, \mathcal{M} \models A\vec{x}v \bullet \vec{c}\},$$

where $\mathcal{M}$ is the model of LA. An SLI $\Lambda$ is called ***satisfiable***, if its feasible region $Feas(\Lambda)$ is non-empty. We say that an (in)equation $\vec{a}^T\vec{x} \le c$ or an SLI $\Lambda$ ***holds*** (or ***is valid***) for a base assignment $v$, meaning that the respective object is consistent when the assignment $v$ is applied to it, for instance: $1x_1 - 2x_2 \le 3$ holds for $v = [x_1 \mapsto 1, x_2 \mapsto 2]$. For short, the feasible region can be written as

$$Feas(\Lambda) = \{\vec{x}v \mid A\vec{x}v \bullet \vec{c}\}.$$

Any system of linear (in)equations can be transformed into the ***standard form***

$$\Lambda = \begin{pmatrix} A'\vec{x} & \le & \vec{c}' \\ A''\vec{x} & < & \vec{c}'' \end{pmatrix}$$

with the corresponding feasible region

$$Feas(\Lambda) = \{\vec{x}v \mid A'\vec{x}v \le \vec{c}', A''\vec{x}v < \vec{c}''\}.$$

**LP Problems**

A ***linear programming*** (***LP***) ***problem*** is one of maximizing or minimizing a linear function subject to a system of linear (in)equations. A general form of an LP problem is

$$\begin{array}{rl} \text{maximize:} & b_1x_1 + \cdots + b_nx_n \\ \text{subject to:} & a_{11}x_1 + \cdots + a_{1n}x_n \bullet_1 c_1, \\ & \vdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n \bullet_m c_m, \end{array}$$

where:

- $x_1, \ldots, x_n$ are variables,
- $\bullet_1, \ldots, \bullet_m \in \{\le, \approx, \ge\}$ senses of the (in)equations, and
- $b_i$, $a_{ij}$, $c_j \in \mathbb{Q}$ rational coefficients, for all $1 \le i \le n$, $1 \le j \le m$.

The "maximize" can alternatively be "minimize".

The linear function $b_1x_1 + \cdots + b_nx_n$ that we seek to maximize (minimize) is called the ***objective function***, and $b_1, \ldots, b_n$ are called the ***objective coefficients***. The (in)equations $a_{i1}x_1 + \cdots + a_{in}x_n \bullet_i c_i$ (for $i = 1, \ldots, m$) are referred to as the ***elementary constraints***, the values $c_1, \ldots, c_m$ are called the ***right-hand side values***.

It is convenient to express an LP problem in the ***matrix form***:

$$\begin{array}{rl} \text{maximize:} & \vec{b}^T\vec{x} \\ \text{subject to:} & A\vec{x} \bullet \vec{c}. \end{array}$$

where:

- matrix $A$ consists of rows $(a_{i1}, \ldots, a_{in})$, for all $1 \le i \le m$,
- $\vec{c} = (c_1, \ldots, c_m)^T$, and
- $\bullet = (\bullet_1, \ldots, \bullet_m)^T$.

The matrix $A$ is called the ***constraint matrix***, the vector $\vec{c}$ the ***right-hand-side vector***.

An LP problem is called ***feasible***, if the SLI $A\vec{x} \bullet \vec{c}$ is satisfiable, i.e. if there exists a base assignment $v$ of variables $\vec{x}$ to rationals satisfying every (in)equation in $A\vec{x} \bullet \vec{c}$; such an assignment $v$ is called ***satisfying***.

An ***optimal solution*** of an LP problem is determined by a satisfying assignment $v$, under which the value of the objective function $b_1 x_1 v + \cdots + b_n x_n v$ attains the largest value. An LP problem might have no optimal solution at all, if (i) it is infeasible or (ii) unbounded in the direction of gradient[1] of the objective function. In the second case, the LP problem is said to have an ***unbounded solution***.

It is well known that feasibility of a linear program can be tested in weakly polynomial time using the Ellipsoid method. For further details we refer to [Sch89].

**Constraint Representation**

Let $C = \Lambda \parallel \Gamma \rightarrow \Delta$ be an abstracted clause over FOL(LA). We can identify the constraint $\Lambda$ of the clause, given as a SLI

$$\Lambda = A\vec{x} \bullet \vec{c},$$

for a rational matrix $A \in \mathbb{Q}^{m \times n}$, variable vector $\vec{x}$, rational vector $\vec{c} \in \mathbb{Q}^m$, sense vector $\bullet \in \{\leq, <, \approx, >, \geq\}^m$, and $n$ and $m$ the numbers of variables and (in)equations in $\Lambda$, respectively, with the subset of $\mathbb{Q}^n$ of the feasible region of $\Lambda$, i.e. the set

$$Feas(\Lambda) = \{\vec{x}v \mid A\vec{x}v \bullet \vec{c}\}.$$

For the sake of conciseness, we assume that $\Lambda$ is in the standard form, without loss of generality.

## 5.2.2 Satisfiability Test

Let $C = \Lambda \parallel \Gamma \rightarrow \Delta$ be an abstracted clause over FOL(LA). The constraint $\Lambda$ of the clause is satisfiable, iff the feasible region $Feas(\Lambda)$ of $\Lambda$ is non-empty. Hence, the satisfiability of a constraint of a clause can be related to testing feasibility of a linear program. We present an algorithm to check the satisfiability of an SLI, called ***Satisfiability Test***.

If the constraint $\Lambda$ contains no strict inequation:

$$\Lambda = \left(A\vec{x} \leq \vec{c}\right)$$

the corresponding linear program is defined on the set of inequations in $\Lambda$ with an arbitrary objective function:

$$\begin{array}{lrcl} \text{maximize:} & & 0 & \\ \text{subject to:} & A\vec{x} & \leq & \vec{c} \end{array} \qquad \blacktriangleleft \ (5.2)$$

◀ THEOREM 5.3

*Let $\Lambda = \left(A\vec{x} \leq \vec{c}\right)$ be a system of linear inequations. The SLI is satisfiable if and only if the LP problem defined by (5.2) is feasible.*

If a constraint $\Lambda$ contains strict inequalities:

$$\Lambda = \begin{pmatrix} A'\,\vec{x} \leq \vec{c}\,' \\ A''\vec{x} < \vec{c}\,'' \end{pmatrix}$$

[1]The gradient of an objective function $b_1 x_1 + \cdots + b_n x_n$ is the vector $(b_1, \ldots, b_n)^T$ of the function's coefficients.

the corresponding linear program is constructed in the following way: we add one extra variable $\delta \geq 0$ to every strict inequation, turn the strict inequations to non-strict, and maximize $\delta$ over the obtained system of inequations:

(5.3) ►
$$\begin{array}{rrcl} \text{maximize:} & & \delta & \\ \text{subject to:} & A'\vec{x} & \leq & \vec{c}', \\ & A''\vec{x}+\delta & \leq & \vec{c}'', \\ & \delta & \geq & 0, \end{array}$$

If the LP problem has a strictly positive optimal solution or an unbounded solution, then $\Lambda$ is satisfiable. In order to increase the numerical stability of the test, it can also be executed for every strict inequation separately.

THEOREM 5.4 ► *Let $\Lambda = \begin{pmatrix} A'\vec{x} \leq \vec{c}' \\ A''\vec{x} < \vec{c}'' \end{pmatrix}$ be a system of linear inequations. The SLI is satisfiable if and only if the LP problem defined by (5.3) has a strictly positive optimal solution or an unbounded solution.*

### 5.2.3 Implication Test

Let $\Lambda_1$ and $\Lambda_2$ be two systems of linear inequations. Assume the set of variables of $\Lambda_1$ is contained in that of $\Lambda_2$, i.e. $var(\Lambda_1) \subseteq var(\Lambda_2)$. We say $\Lambda_2$ ***implies*** $\Lambda_1$ if the feasible region of $\Lambda_2$ is contained in the feasible region of $\Lambda_1$. Next we show how this containment problem can be reduced to testing feasibility of a linear program. We present an algorithm used to establish SLIs containment, called ***Implication Test***. More formally, we discuss our approach to solve the following problem: given two SLIs

$$\Lambda_1 = \begin{pmatrix} A'\,\vec{x} \leq \vec{c}' \\ A''\vec{x} < \vec{c}'' \end{pmatrix} \quad \text{and} \quad \Lambda_2 = \begin{pmatrix} B'\,\vec{x} \leq \vec{d}' \\ B''\vec{x} < \vec{d}'' \end{pmatrix}$$

determine whether $\Lambda_2$ implies $\Lambda_1$.

First, we discuss the case when all inequations are non-strict, which is based on the well known Farkas' Lemma (see e.g. [Sch89]), whereupon we extend the Farkas' Lemma to the case of mixed strict and non-strict inequations, which is then used to solve the general case.

LEMMA 5.5 ► Farkas' Lemma [Sch89] (affine variant)

*Let $\Lambda = (B\vec{x} \leq \vec{d})$ be a satisfiable system of $m$ linear inequations, $\vec{a}^T\vec{x} \leq c$ an inequation. Every point in the feasible region $Feas(\Lambda)$ satisfies the inequation, if and only if there exists a rational vector $\vec{p} \in \mathbb{Q}^m$ such that:*

*(i)* $\vec{p} \geq 0$,

*(ii)* $\vec{p}^T B = \vec{a}^T$, *and*

*(iii)* $\vec{p}^T \vec{d} \leq c$.

Informally speaking, the Farkas' Lemma states that (the feasible region of) $\Lambda$ is contained in (the feasible region of) $\vec{a}^T\vec{x} \leq c$ if the vector $\vec{a}$ is a sum of non-negative (condition (i)) portions of rows $B_i$ (condition (ii)) such that the sum of the respective portions of the right-hand-sides $d_i$ is at most $c$ (condition (iii)), or, equivalently saying, if a non-negative linear combination of inequations in $\Lambda$

equals an inequation $\vec{a}^T\vec{x} \le c'$ with the rhs $c'$ smaller than or equal to $c$:

$$\begin{aligned} \vec{p}^T(\Lambda) &= \left(\, \vec{p}^T(B\vec{x}) \le \vec{p}^T\vec{d}\,\right) \\ &= \left(\quad \vec{a}^T\vec{x} \quad \le \underbrace{c'}_{\le c}\,\right) \end{aligned}$$

An SLI $\Lambda_2$ implies an SLI $\Lambda_1$ if and only if every point in the feasible region of $\Lambda_1$ satisfies all inequations in $\Lambda_2$. Based on this observation and the Farkas' Lemma we conclude in the following corollary necessary and sufficient conditions under which an SLI implies another SLI, for the case when the both SLIs do not contain strict inequations.

◄ COROLLARY 5.6

*Let $\Lambda_1 = \left(A\vec{x} \le \vec{c}\right)$ and $\Lambda_2 = \left(B\vec{x} \le \vec{d}\right)$ be two systems of $m_1$ and $m_2$ linear inequations, respectively. If $\Lambda_2$ is satisfiable, then $\Lambda_2$ implies $\Lambda_1$, if and only if for every inequation $A_i\vec{x} \le c_i$ in $\Lambda_1$ there exists a rational vector $\vec{p} \in \mathbb{Q}^{m_2}$ such that:*

*(i)* $\vec{p} \ge 0$,

*(ii)* $\vec{p}^T B = A_i$, *and*

*(iii)* $\vec{p}^T\vec{d} \le c_i$.

Let $\Lambda_1$ and $\Lambda_2$ be defined as in Corollary 5.6. Rewriting the rows $\vec{p}^T$, defined in the corollary for each inequation $A_i\vec{x} \le c_i$, as a matrix of *unknowns* $P$, we can determine whether $\Lambda_2$ implies $\Lambda_1$ by testing feasibility of the following LP problem:

$$\begin{array}{rrcl} \text{maximize:} & & 0 & \\ \text{subject to:} & PB & \approx & A \\ & P\vec{d} & \le & \vec{c} \\ & P & \ge & 0 \end{array} \qquad ◄\ (5.4)$$

Putting $n_x$ equal the number of variables in $\vec{x}$, the above LP problem contains

$$\begin{aligned} m &= m_1 n_x + m_1 + m_2 m_1 \\ &= m_1(n_x + m_2 + 1) \end{aligned}$$

(in)equations over

$$n = m_1 m_2$$

variables (which constitute the matrix $P$). Note, that we are only interested in the existence of a solution to the above LP problem and do not need to compute concrete values of the unknown multipliers in the matrix $P$.

◄ THEOREM 5.7
SLIs Implication (non-strict case)

*Let $\Lambda_1 = \left(A\vec{x} \le \vec{c}\right)$ and $\Lambda_2 = \left(B\vec{x} \le \vec{d}\right)$ be two systems of linear inequations. If $\Lambda_2$ is satisfiable, then $\Lambda_2$ implies $\Lambda_1$ if and only if the LP problem defined by (5.4) is feasible.*

Also, alternatively we could solve the following $m_1$ smaller LP problems, where $m_1$ is the number of inequations in $\Lambda_1$:

$$\begin{array}{rrcl} \text{maximize:} & & 0 & \\ \text{subject to:} & \vec{p}^T B & \approx & A_i \\ & \vec{p}^T\vec{d} & \le & c_i \\ & \vec{p} & \ge & 0 \end{array} \qquad ◄\ (5.5)$$

for each inequation $A_i\vec{x} \le c_i$ in $\Lambda_1$ *independently* from each other: $\Lambda_2$ implies $\Lambda_1$ iff each of the LP problems is feasible. Each of the LP problems contains $n_x + 1 + m_2$ (in)equations over $m_2$ variables (which constitute the vector $\vec{p}$).

In the following lemma we provide an extension of the Farkas' Lemma to the case of strict inequations.

LEMMA 5.8 ► Farkas' Lemma for strict inequations

*Let $\Lambda = \left(\begin{smallmatrix} B'\vec{x} & \le & \vec{d}' \\ B''\vec{x} & < & \vec{d}'' \end{smallmatrix}\right)$ be a satisfiable system of $m'$ non-strict and $m''$ strict linear inequations, respectively, containing the trivial inequation $0 < 1$. Let $\vec{a}^T\vec{x} \circ c$ be an inequation, where $\circ \in \{\le, <\}$. Every point in the feasible region Feas($\Lambda$) of $\Lambda$ satisfies the inequation, if and only if there exists two rational vectors $\vec{p}' \in \mathbb{Q}^{m'}$ and $\vec{p}'' \in \mathbb{Q}^{m''}$ such that:*

*(i)* $\vec{p}', \vec{p}'' \ge 0$,

*(ii)* $\vec{p}'^T B' + \vec{p}''^T B'' = \vec{a}^T$,

*(iii)* $\vec{p}'^T d' + \vec{p}''^T d'' \le c$, *and*

*(iv)* $\vec{p}'' \ne 0$, *if* $\circ$ *is* $<$.

PROOF ► Two alternative proofs have been suggested by us in [Kru08] (Theorems 5.4 and 5.7) and [AKW09a, EKK+11] (Lemma 2)[1]. ■

Assume $\Lambda = \left(\begin{smallmatrix} B'\vec{x} & \le & \vec{d}' \\ B''\vec{x} & < & \vec{d}'' \end{smallmatrix}\right)$, then set a matrix $B$ and vectors $\vec{d}$ and $\bullet$ as follows:

$$B = \begin{pmatrix} B' \\ B'' \end{pmatrix} \qquad \vec{d} = \begin{pmatrix} \vec{d}' \\ \vec{d}'' \end{pmatrix} \qquad \bullet = \begin{pmatrix} \le \\ \vdots \\ < \\ \vdots \end{pmatrix} \begin{matrix} \}\, m' \\ \}\, m'' \end{matrix}$$

such that $\Lambda = (B\vec{x} \bullet \vec{d})$, where $m'$ and $m''$ are the numbers of non-strict and strict inequations in $\Lambda$, respectively. Put $\vec{p} = \left(\begin{smallmatrix} \vec{p}' \\ \vec{p}'' \end{smallmatrix}\right)$. Informally speaking, the above extension of the Farkas' Lemma differs from the affine variant (Lemma 5.5) in condition (iv), according to which at least one strict inequation in $\Lambda$ (including $0 < 1$) has to contribute into the non-negative linear combination $\vec{p}^T(\Lambda)$ of $\Lambda$'s inequations, such that

$$\begin{aligned} \vec{p}^T(\Lambda) &= \vec{p}^T\big(B\vec{x} \bullet \vec{d}\big) \\ &= \big(\vec{p}^T(B\vec{x}) \circ \vec{p}^T\vec{d}\big) \\ &= \big(\ \vec{a}^T\vec{x} \ \circ \underbrace{c'}_{\le c}\ \big) \end{aligned}$$

where $\circ \in \{\le, <\}$. In this case, it is guaranteed that if $c' = c$, then the sense of the combination is strict.

Based on the Farkas' Lemma for strict inequations we conclude in the following corollary necessary and sufficient conditions of implying an SLI by another for the general case when the both SLIs may contain strict inequations.

COROLLARY 5.9 ► *Let $\Lambda_1 = \left(\begin{smallmatrix} A'\vec{x} & \le & \vec{c}' \\ A''\vec{x} & < & \vec{c}'' \end{smallmatrix}\right)$ and $\Lambda_2 = \left(\begin{smallmatrix} B'\vec{x} & \le & \vec{d}' \\ B''\vec{x} & < & \vec{d}'' \end{smallmatrix}\right)$ be two systems of $m_1'$ and $m_2'$ non-strict and $m_1''$ and $m_2''$ strict linear inequations, respectively. Let $A = \left(\begin{smallmatrix} A' \\ A'' \end{smallmatrix}\right)$ and $\vec{c} = \left(\begin{smallmatrix} \vec{c}' \\ \vec{c}'' \end{smallmatrix}\right)$. If $\Lambda_2$ is satisfiable and contains the trivial inequation $0 < 1$, then $\Lambda_2$ implies $\Lambda_1$, if and*

[1]The proof appearing in [Kru08] exploits the very basic notions of Linear Algebra, whereas the one from [AKW09a, EKK+11] takes advances of the theory of Optimization. Thus, the latter is shorter in comparison to the former, but, on the other hand, is more sophisticated and involving.

*only if for every inequation $A_i\vec{x} \circ c_i$ in $\Lambda_1$, where $\circ \in \{\le, <\}$, there exist two rational vectors $\vec{p}' \in \mathbb{Q}^{m_2'}$ and $\vec{p}'' \in \mathbb{Q}^{m_2''}$ such that:*

*(i)* $\vec{p}', \vec{p}'' \ge 0$,

*(ii)* $\vec{p}'^T B' + \vec{p}''^T B'' = A_i$,

*(iii)* $\vec{p}'^T d' + \vec{p}''^T d'' \le c_i$, *and*

*(iv)* $\vec{p}'' \ne 0$, *if* $\circ$ *is* $<$.

Let $\Lambda_1$ and $\Lambda_2$ be as defined in Corollary 5.9. Rewriting the rows $\vec{p}'^T$ and $\vec{p}''^T$ defined in the corollary for each *non-strict* inequation $A_i'\vec{x} \le c_i'$ as matrices of *unknowns* $P^1$ and $P^2$, and rewriting the rows $\vec{p}'^T$ and $\vec{p}''^T$ defined in the corollary for each *strict* inequation $A_i''\vec{x} < c_i''$ as matrices of *unknowns* $P^3$ and $P^4$, respectively, we can determine whether $\Lambda_2$ implies $\Lambda_1$ by testing the following linear program:

$$\begin{array}{lrl} \text{maximize:} & \delta & \\ \text{subject to:} & P^1\, B' + P^2\, B'' & \approx A' \\ & P^3\, B' + P^4\, B'' & \approx A'' \\ & P^1\, \vec{d}' + P^2\, \vec{d}'' & \le \vec{c}' \\ & P^3\, \vec{d}' + P^4\, \vec{d}'' & \le \vec{c}'' \\ & P^1, P^2, P^3, P^4 & \ge 0 \\ & P^4\mathbb{1} - \delta & \ge 0 \\ & \delta & \ge 0 \end{array} \tag{5.6}$$

where $\mathbb{1} = (1, \ldots, 1)^T$ is the all ones vector of dimension $m_2''$ (the number of strict inequations in $\Lambda_2$). If the LP problem has an optimal solution different from zero or an unbounded solution, then the implication holds. Note again, that we are only interested in the existence of a solution to the above LP problem and do not need to compute concrete values of the multipliers forming the matrices $P^1, P^2, P^3, P^4$ of unknowns. Putting $n_x$ equal the number of variables in $\vec{x}$, the above LP problem contains

$$\begin{aligned} m &= (m_1' + m_1'')n_x + m_1' + m_1'' + (m_1' + m_1'')(m_2' + m_2'') + 2 \\ &= m_1 n_x + m_1 + m_1 m_2 + 2 \qquad \text{// where } m_i = m_i' + m_i'', \text{ for } \forall i \in \{1,2\} \\ &= m_1(n_x + m_2 + 1) + 2 \end{aligned}$$

(in)equations over

$$\begin{aligned} n &= (m_1' + m_1'')(m_2' + m_2'') + 1 \\ &= m_1 m_2 + 1 \end{aligned}$$

variables (which constitute the matrices $P^1, \ldots, P^4$ and the variable $\delta$).

THEOREM 5.10
SLIs Implication

*Let $\Lambda_1 = \begin{pmatrix} A'\vec{x} \le \vec{c}' \\ A''\vec{x} < \vec{c}'' \end{pmatrix}$ and $\Lambda_2 = \begin{pmatrix} B'\vec{x} \le \vec{d}' \\ B''\vec{x} < \vec{d}'' \end{pmatrix}$ be two systems of linear inequations, where $\vec{y}$ and $\vec{z}$ share no variable. If $\Lambda_2$ is satisfiable and contains the trivial inequation $0 < 1$, then $\Lambda_2$ implies $\Lambda_1$ if and only if the LP problem defined by (5.6) has an optimal solution different from zero or an unbounded solution.*

Also, alternatively we could solve the following $m_1'$ LP problems for each non-strict inequation $A_i'\vec{x} \le c_i'$ and $m_1''$ LP problems for each strict inequation $A_j''\vec{x} < c_j''$ in $\Lambda_1$ *independently* from each other; $\Lambda_2$ implies $\Lambda_1$ iff each of the LP problems has a non-zero solution:

(5.7) ►
$$\begin{array}{ll} \text{maximize:} & 1 \\ \text{subject to:} & \vec{p}'^T B' + \vec{p}''^T B'' \approx A'_i \\ & \vec{p}'^T \vec{d}' + \vec{p}''^T \vec{d}'' \leq c'_i \\ & \vec{p}', \vec{p}'' \geq 0 \end{array} \qquad \begin{array}{ll} \text{maximize:} & \delta \\ \text{subject to:} & \vec{p}'^T B' + \vec{p}''^T B'' \approx A''_j \\ & \vec{p}'^T \vec{d}' + \vec{p}''^T \vec{d}'' \leq c''_j \\ & \vec{p}', \vec{p}'' \geq 0 \\ & \vec{p}''^T \mathbb{1} - \delta \geq 0 \\ & \delta \geq 0 \end{array}$$

Each of the above LP problems for non-strict inequations contains

$$\begin{aligned} m' &= m'_1 n_x + m'_1 + (m'_2 + m''_2) \\ &= m'_1 (n_x + 1) + m_2 \qquad \text{// where } m_2 = m'_2 + m''_2 \end{aligned}$$

(in)equations over $m_2$ variables (which constitute the vectors $\vec{p}'$ and $\vec{p}''$). Each of the above LP problems for strict inequations contains

$$\begin{aligned} m'' &= m''_1 n_x + m''_1 + (m'_2 + m''_2) + 2 \\ &= m''_1 (n_x + 1) + m_2 + 2 \end{aligned}$$

(in)equations over $m_2 + 1$ variables (which constitute the vectors $\vec{p}'$, $\vec{p}''$ and the variable $\delta$).

EXAMPLE 5.11 ► To illustrate Corollary 5.9, let us consider the following SLIs:

$$\Lambda_1 = \begin{pmatrix} x_1 & + & 2x_2 & + & 3x_3 & \leq & 5 \\ 2x_1 & + & 4x_2 & + & x_3 & < & 4 \end{pmatrix}$$

$$\Lambda_2 = \begin{pmatrix} x_1 & + & 2x_2 & + & 3x_3 & \leq & 4 \\ 0.5x_1 & + & x_2 & - & x_3 & < & 0 \\ & & & & 0 & < & 1 \end{pmatrix}$$

Putting

$$P^1 = (1) \qquad P^2 = (0 \quad 0)$$
$$P^3 = (1) \qquad P^4 = (2 \quad 0)$$

we obtain:

$$P^1 \underbrace{\begin{pmatrix}1 & 2 & 3\end{pmatrix}}_{B'} + P^2 \underbrace{\begin{pmatrix}0.5 & 1 & -1 \\ 0 & 0 & 0\end{pmatrix}}_{B''} = \underbrace{\begin{pmatrix}1 & 2 & 3\end{pmatrix}}_{A'} \qquad P^1 \underbrace{\begin{pmatrix}4\end{pmatrix}}_{\vec{d}'} + P^2 \underbrace{\begin{pmatrix}0 \\ 1\end{pmatrix}}_{\vec{d}''} \leq \underbrace{\begin{pmatrix}5\end{pmatrix}}_{\vec{c}'}$$

$$P^3 \underbrace{\begin{pmatrix}1 & 2 & 3\end{pmatrix}}_{B'} + P^4 \underbrace{\begin{pmatrix}0.5 & 1 & -1 \\ 0 & 0 & 0\end{pmatrix}}_{B''} = \underbrace{\begin{pmatrix}2 & 4 & 1\end{pmatrix}}_{A''} \qquad P^3 \underbrace{\begin{pmatrix}4\end{pmatrix}}_{\vec{d}'} + P^4 \underbrace{\begin{pmatrix}0 \\ 1\end{pmatrix}}_{\vec{d}''} \leq \underbrace{\begin{pmatrix}4\end{pmatrix}}_{\vec{c}''}$$

Therefore, $\Lambda_2$ implies $\Lambda_1$. ■

### 5.2.4 Encompassment Matcher Existence

In Section 5.1 we have shown that the encompassment condition of the Hierarchic Subsumption Deletion rule (Definition 3.97, condition (ii)) can be approximated by assuring existence of a simple matcher $\tau$, called ***encompassment matcher***, which makes the condition

$$\models_{\mathscr{C}} \forall \vec{x}.(\bigwedge \Lambda_2 \rightarrow \bigwedge \Lambda_1 \tau),$$

to hold, where $dom(\tau) = var(\Lambda_1) \setminus var(C_2)$ and $cdom(\tau) \subseteq var(\Lambda_2) \subseteq var(C_2)$, for the constraints $\Lambda_1$ and $\Lambda_2$ of the reduction's premises $C_1$ and $C_2$. Here we present a procedure that decides if there exists an encompassment matcher $\tau$ which is an *affine substitution* (Definition 5.2), that maps the variables that solely occur in $\Lambda_1$ (and do not appear in $\Lambda_2$) to a sum of a linear combination of variables of $\Lambda_2$ and a rational constant. We stick to affine encompassment matchers because in this case we can use the effective algorithms from Section 5.2.3 to determine whether $\Lambda_2$ implies $\Lambda_1\tau$ for some $\tau$.

First, we consider the case when given SLIs $\Lambda_1$ and $\Lambda_2$ contain only non-strict inequations:

$$\Lambda_1 = \big(A\vec{x} + G\vec{y} \le \vec{c}\big) \quad \text{and} \quad \Lambda_2 = \big(B\vec{x} + H\vec{z} \le \vec{d}\big) \qquad ◀ \text{ (5.8)}$$

where the vectors $\vec{y}$ and $\vec{z}$ do not share variables. Let $m_1$ and $m_2$ be the numbers of rows in $\Lambda_1$ and $\Lambda_2$, respectively; and $n_x$, $n_y$, and $n_z$ equal the sizes of the variable vectors $\vec{x}$, $\vec{y}$, and $\vec{z}$, respectively. Consider an arbitrary affine matcher $\tau$ such that

$$\vec{y}\tau = S\vec{x} + T\vec{z} + \vec{\beta} \qquad ◀ \text{ (5.9)}$$

where $S \in \mathbb{Q}^{n_y \times n_x}$ and $T \in \mathbb{Q}^{n_y \times n_z}$ are matrices of coefficients of linear combinations over variables $\vec{x}$ and $\vec{y}$ respectively, and $\vec{\beta} \in \mathbb{Q}^{n_y}$ a vector of rational constants. Thus, application of $\tau$ to each every variable $y_i \in \vec{y}$ yields

$$y_i\tau = S_i\vec{x} + T_i\vec{z} + \beta_i.$$

Let us determine the result of applying $\tau$ onto the SLI $\Lambda_1$:

$$\begin{aligned}\Lambda_1\tau &= \big(A\vec{x} + G\vec{y} \le \vec{c}\big)\tau \\ &= \big(A\vec{x} + G\vec{y}\tau \le \vec{c}\big) \\ &= \big(A\vec{x} + G(S\vec{x} + T\vec{z} + \vec{\beta}) \le \vec{c}\big) \\ &= \big((A+GS)\vec{x} + GT\vec{z} \le \vec{c} - G\vec{\beta}\big)\end{aligned}$$

Assume $\Lambda_2$ is satisfiable. According to Corollary 5.6, $\Lambda_2$ implies $\Lambda_1\tau$ if and only if for every inequation

$$(A+GS)_i\vec{x} + (GT)_i\vec{z} \le c_i - (G\vec{\beta})_i$$

in $\Lambda_1\tau$, where $(A+GS)_i$ and $(GT)_i$ denote the $i$-th row in the matrices $A+GS$ and $GT$, respectively, and $(G\vec{\beta})_i$ denotes the $i$-th entry in the vector $G\vec{\beta}$, there exists a rational vector $\vec{p} \in \mathbb{Q}^{m_2}$ such that:

(i) $\vec{p} \ge 0$,

(ii) $\vec{p}^T B = (A+GS)_i$ and $\vec{p}^T H = (GT)_i$, and

(iii) $\vec{p}^T \vec{d} \le c_i - (G\vec{\beta})_i$.

Put $P$, $S$, $T$ be matrices of unknowns of size $m_1 \times m_2$, $n_y \times n_x$, and $n_y \times n_z$, respectively, and $\beta$ a vector of unknowns of size $n_y$. We can determine whether there exists an encompassment matcher $\tau$ such that $\Lambda_2$ implies $\Lambda_1\tau$ by testing feasibility of the following LP problem:

$$\begin{array}{ll}\text{maximize:} & 0 \\ \text{subject to:} & P\,B \approx A + G\,S \\ & P\,H \approx GT \\ & P\,\vec{d} \le \vec{c} - G\,\vec{\beta} \\ & \quad P \ge 0\end{array} \qquad ◀ \text{ (5.10)}$$

This LP problem contains

$$\begin{aligned} m &= m_1 n_x + m_1 n_z + m_1 + m_1 m_2 \\ &= m_1(n_x + n_z + m_2 + 1) \end{aligned}$$

(in)equations over

$$\begin{aligned} n &= m_1 m_2 + n_x n_y + n_z n_y + n_y \\ &= m_1 m_2 + n_y(n_x + n_z + 1) \end{aligned}$$

variables (which constitute the matrices $P$, $S$, $T$ and vector $\beta$). Note, that we are only interested in the existence of a solution to the above LP problem and do not need to compute concrete values of the unknowns in $P$, $S$, $T$, $\beta$. In contrast to the case when $var(\Lambda_1) \subseteq var(\Lambda_2)$ (i.e. when no encompassment matcher needs to be established), here we cannot reduce the LP problem to solving several smaller LP problems, because, in general, the unknown coefficients in $S$, $T$ and $\beta$ of $\tau$ are common for all inequations in $\Lambda_1\tau$ and therefore have to be subjected jointly.

THEOREM 5.12 ► Affine Encompassment Matcher Existence (non-strict case)
Let $\Lambda_1 = \left(A\vec{x} + G\vec{y} \le \vec{c}\right)$ and $\Lambda_2 = \left(B\vec{x} + H\vec{z} \le \vec{d}\right)$ be two systems of linear inequations, where $\vec{y}$ and $\vec{z}$ share no variable. If $\Lambda_2$ is satisfiable, then there exists an affine encompassment matcher $\tau$ such that $\Lambda_2$ implies $\Lambda_1\tau$ if and only if the LP problem defined by (5.10) is feasible. ■

Now, assume the given clause constraints $\Lambda_1$ and $\Lambda_2$ may contain strict inequations; without loss of generality, let they be defined as follows:

(5.11) ►
$$\Lambda_1 = \begin{pmatrix} A'\ \vec{x} + G'\ \vec{y} \le \vec{c}\,' \\ A''\ \vec{x} + G''\ \vec{y} < \vec{c}\,'' \end{pmatrix} \quad \text{and} \quad \Lambda_2 = \begin{pmatrix} B'\ \vec{x} + H'\ \vec{z} \le \vec{d}\,' \\ B''\ \vec{x} + H''\ \vec{z} < \vec{d}\,'' \end{pmatrix}$$

where the vectors $\vec{y}$ and $\vec{z}$ do not share variables, and $\Lambda_2$ contains the trivial strict inequation $0 < 1$. Let $m_1'$, $m_1''$ and $m_2'$, $m_2''$ be the numbers of non-strict and strict inequations in $\Lambda_1$ and $\Lambda_2$, respectively. As before, $n_x$, $n_y$, and $n_z$ equal the sizes of the variable vectors $\vec{x}$, $\vec{y}$, and $\vec{z}$, respectively. Let $\tau$ be an arbitrary affine matcher as defined by (5.9):

$$\vec{y}\tau = S\vec{x} + T\vec{z} + \vec{\beta},$$

for some $S \in \mathbb{Q}^{n_y \times n_x}$, $T \in \mathbb{Q}^{n_y \times n_z}$, and $\vec{\beta} \in \mathbb{Q}^{n_y}$. The result of applying $\tau$ onto the SLI $\Lambda_1$ gives

(5.12) ►
$$\Lambda_1\tau = \begin{pmatrix} (A' + G'S)\ \vec{x} + G'T\ \vec{z} \le \vec{c}\,' - G'\ \vec{\beta} \\ (A'' + G''S)\ \vec{x} + G''T\ \vec{z} < \vec{c}\,'' - G''\ \vec{\beta} \end{pmatrix}$$

Let matrices $A$ and $G$ and vectors $\vec{c}$ and $\bullet$ be defined as follows:

$$A = \begin{pmatrix} A' \\ A'' \end{pmatrix} \qquad G = \begin{pmatrix} G' \\ G'' \end{pmatrix} \qquad \vec{c} = \begin{pmatrix} \vec{c}\,' \\ \vec{c}\,'' \end{pmatrix} \qquad \bullet = \begin{pmatrix} \le \\ \vdots \\ < \\ \vdots \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\b\end{matrix}}\right\} m_1' \\ \left.\vphantom{\begin{matrix}a\\b\end{matrix}}\right\} m_2'' \end{matrix}$$

Then (5.12) reduces to

(5.13) ►
$$\Lambda_1\tau = \left((A + GS)\vec{x} + GT\vec{z} \bullet \vec{c} - G\vec{\beta}\right)$$

Assume $\Lambda_2$ is satisfiable and includes the trivial inequation $0 < 1$. According to Corollary 5.9, $\Lambda_2$ implies $\Lambda_1\tau$, if and only if for every inequation in $\Lambda_1\tau$, there exist two rational vectors $\vec{p}\,' \in \mathbb{Q}^{m_2'}$ and $\vec{p}\,'' \in \mathbb{Q}^{m_2''}$ such that:

(i) $\vec{p}\,', \vec{p}\,'' \ge 0$,

(ii) $\vec{p}'^T B' + \vec{p}''^T B'' = (A+GS)_i$ and $\vec{p}'^T H' + \vec{p}''^T H'' = (GT)_i$,

(iii) $\vec{p}'^T d' + \vec{p}''^T d'' \leq c_i - (G\vec{\beta})_i$, and

(iv) $\vec{p}'' \neq 0$, if $\bullet_i$ is $<$,

where $(A'+GS)_i$ and $(GT)_i$ denote the $i$-th row in the matrices $A'+GS$ and $GT$, respectively, and $(G\vec{\beta})_i$ denotes the $i$-th entry in the vector $G\vec{\beta}$.

Put $P^1$, $P^2$, $P^3$, and $P^4$ be matrices of unknowns of size $m_1' \times m_2'$, $m_1' \times m_2''$, $m_1'' \times m_2'$, and $m_1'' \times m_2''$ respectively; $S$, $T$ matrices of unknowns of size $n_y \times n_x$, and $n_y \times n_z$, respectively; and $\beta$ a vector of unknowns of size $n_y$. We can determine whether there exists an encompassment matcher $\tau$ such that $\Lambda_2$ implies $\Lambda_1\tau$ by testing feasibility of the following LP problem:

$$\begin{array}{ll} \text{maximize:} & \delta \\ \text{subject to:} & P^1 B' + P^2 B'' \approx A' + G'S \\ & P^3 B' + P^4 B'' \approx A'' + G''S \\ & P^1 H' + P^2 H'' \approx G'T \\ & P^3 H' + P^4 H'' \approx G''T \\ & P^1 \vec{d}' + P^2 \vec{d}'' \leq \vec{c}' - G'\vec{\beta} \\ & P^3 \vec{d}' + P^3 \vec{d}'' \leq \vec{c}'' - G''\vec{\beta} \\ & P^1, P^2, P^3, P^4 \geq 0 \\ & P^4\mathbb{1} - \delta \geq 0 \\ & \delta \geq 0 \end{array} \tag{5.14}$$

where $\mathbb{1} = (1,\dots,1)^T$ is the all-ones vector of size $m_2''$ (the number of strict inequations in $\Lambda_2$). This LP problem contains

$$\begin{aligned} m &= m_1'n_x + m_1''n_x + m_1'n_z + m_1''n_z + m_1' + m_1'' + (m_1' + m_1'')(m_2' + m_2'') + 2 \\ &= (m_1' + m_1'')(n_x + n_z) + m_1' + m_1'' + (m_1' + m_1'')(m_2' + m_2'') + 2 \\ &= m_1(n_x + n_z + m_2) + 2 \qquad \text{// where } m_i = m_i' + m_i'', \text{ for } \forall i \in \{1,2\} \end{aligned}$$

(in)equations over

$$\begin{aligned} n &= m_1'm_2' + m_1'm_2'' + m_1''m_2' + m_1''m_2'' + n_xn_y + n_zn_y + n_y \\ &= (m_1' + m_1'')(m_2' + m_2'') + n_xn_y + n_zn_y + n_y \\ &= m_1m_2 + n_xn_y + n_zn_y + n_y \end{aligned}$$

variables (which constitute the matrices $P^1,\dots,P^4,S,T$, vector $\beta$, and variable $\delta$). If the LP problem has an optimal solution different from zero or an unbounded solution, then then there exists a required encompassment matcher $\tau$. Again, we are only interested in the *existence* of a solution to the above LP problem and do not need to compute concrete values of unknowns in $P^1,\dots,P^4$, $S$, $T$, $\beta$. Like in the non-strict case the above LP problem cannot be reduced to several smaller LP problems, because the unknown coefficients $S$, $T$ and $\beta$ of $\tau$ are common for all inequations of $\Lambda_1\tau$ and therefore have to be subjected jointly.

THEOREM 5.13 Affine Encompassment Matcher Existence

*Let* $\Lambda_1 = \begin{pmatrix} A'\vec{x} + G'\vec{y} \leq \vec{c}' \\ A''\vec{x} + G''\vec{y} < \vec{c}'' \end{pmatrix}$ *and* $\Lambda_2 = \begin{pmatrix} B'\vec{x} + H'\vec{z} \leq \vec{d}' \\ B''\vec{x} + H''\vec{z} < \vec{d}'' \end{pmatrix}$ *be two systems of linear inequations, where* $\vec{y}$ *and* $\vec{z}$ *share no variable. If* $\Lambda_2$ *is satisfiable and contains the trivial inequation* $0 < 1$*, then there exists an affine encompassment matcher* $\tau$ *such that* $\Lambda_2$ *implies* $\Lambda_1\tau$ *if and only if the LP problem defined by (5.14) has an optimal solution different from zero or an unbounded solution.*

EXAMPLE 5.14

Consider the example depicted in Figure 5.1. In the left figure, we show the feasible

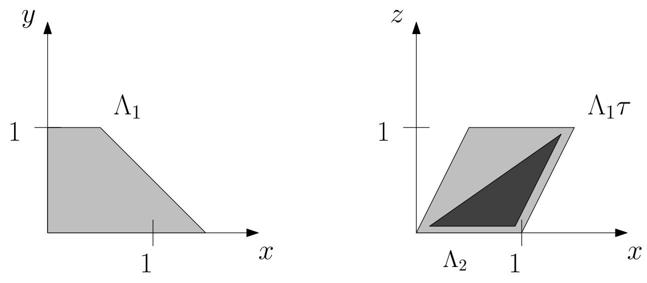


Figure 5.1: Graphical Depicture of SLIs Implication Regarding Encompassment Matcher. (Source of figure: [AKW09a])

region of the SLI $\Lambda_1$.

$$\Lambda_1 = \begin{pmatrix} x & & \geq 0 \\ & y & \geq 0 \\ & y & \leq 1 \\ 2x & +\ 2y & \leq 3 \end{pmatrix} \qquad \Lambda_1\tau = \begin{pmatrix} x & & \geq & 0 \\ -x & +\ 0.5z & \geq & -1 \\ -x & +\ 0.5z & \leq & 0 \\ & z & \leq & 1 \end{pmatrix} \qquad \Lambda_2 = \begin{pmatrix} & z & > & 0 \\ -x\ + & z & \leq & 0 \\ -x & +\ 0.5z & > & -1 \\ & z & < & 1 \end{pmatrix}$$

Choosing $\tau$ such that

$$y\tau = -1x + 0.5z + 1$$

gives the SLI $\Lambda_1\tau$ whose feasible region is shown in light gray in the right figure. The feasible region of $\Lambda_1\tau$ contains the feasible region of $\Lambda_2$ shown in dark gray in the right figure, therefore $\Lambda_2$ implies $\Lambda_1\tau$. ■

## 5.3 Implementation

The free part of the inference rules of the hierarchic superposition calculus described in Section 3.3 is identical to the standard calculus, except that only simple substitutions are considered. For the theory part of clauses an implementation needs to provide instantiation and union of two clause constraints. The operations resulting from subsumption or tautology deletion are much more involved because here the clause constraints need to be mapped to linear programming problems. The existence of solutions to the linear programs eventually decides on the applicability of the reduction rules. As reductions are more often checked than inferences computed, it is essential for an efficient implementation to support the operations needed for reductions. Therefore, we decided to actually store the clause constraint not in a symbolic tree like representation, as it is done for first-order terms, but directly in the input format data structure of LP solvers, where for the published SPASS(LA) binary[1] we rely on QSopt[2]. QSopt uses the simplex-method to solve systems of linear constraints which is not polynomial time but very efficient in practice. Alternatively, we could use an interior-point method to become polynomial while staying efficient in practice.

A key aspect in implementing SPASS(LA) is the representation of the clause constraints. We decided to use a representation that is close to standard LP solver interfaces such that performing the satisfiability test can be done by calling the LP solver directly with our constraint representation. The LP format is a "column-oriented" sparse format, meaning that the problems are represented column by column (variable) rather than row by row ((in)equation) and only non-zero coefficients are stored. Furthermore, the SPASS variable normalization in clauses, done for sharing on the free theory part, fits perfectly to this format. All variables occurring in clauses are subsequently named starting with the "smallest" variable. In addition, the other operations on the free part eventually ranging into the constraints like the application of substitutions and the union of constraints are efficiently mapped to the column-oriented representation.

Besides QSopt, we have also tried combinations of SPASS with other solvers capable of reasoning in linear arithmetic, namely: Z3, LIRA, Redlog.

Z3 is a very powerful SMT solver [dMB08c] integrating a variety of theories: linear and nonlinear arithmetic, bitvectors, arrays, datatypes, uninterpreted functions; and also supporting quantifiers based on such techniques as E-matching (for arrays, partial orders, etc.), heuristic and model-based quantifier instantiation (for the essentially uninterpreted fragment), quantifier elimination (for linear real/integer arithmetic, recursive datatypes, and partially non-linear arithmetic). Though extensive support for quantifiers, Z3 is not refutationally complete for combinations with uninterpreted functions.

LIRA [EK06, BDEK07] decides the first-order logic over mixed integer/real addition using automata: finite deterministic automata (FDA) for integer arithmetic, and weak deterministic Büchi automata (WDBA) for mixed real-integer arithmetic.

Redlog [DS97] implements symbolic algorithms on first-order formulas with respect to user-chosen first-order languages and theories including real numbers, integers, complex numbers, p-adic numbers, quantified propositional calculus,

[1] See `http://spass-prover.org/prototypes`.

[2] See `http://www2.isye.gatech.edu/~wcook/qsopt/`.

term algebras. The core of the system combines the quantifier elimination algorithms of Weispfenning [LW93, Wei94, Wei97] and simplification of quantifier-free formulae, operating on both Boolean structure and algebraic relationships of the formulae [DS95].

In our experiments, each of the tools has had to decide whether an LA formula of one the the following two kinds is valid:

– the existential closure of a conjunction of LA literals

$$\exists\, \vec{x}. \bigwedge \Lambda,$$

arising as a prerequisite of applying the Constraint Refutation rule (Definition 3.31) and Hierarchic Tautology Deletion rule (Definition 3.97), where $\vec{x} = var(\Lambda)$; and

– the $\forall^*\exists^*$-closure of implication between two conjunctions of LA literals

$$\forall\, \vec{x}. \exists \vec{y}. (\bigwedge \Lambda_2 \rightarrow \bigwedge \Lambda_1),$$

required to check if the encompassment condition of the Hierarchic Subsumption Deletion rule (Definition 3.97) is fulfilled, where $\vec{x} = var(C_2) \cap \mathcal{X}$ and $\vec{y} = var(\Lambda_1) \setminus var(C_2)$.

LIRA has shown the slowest performance and executed absolutely unacceptable on any implication formula containing more or less big numbers (for instance, 1000) or rationals, simply running out of memory. Redlog has demonstrated a very good performance on both kinds of formulae. Z3 has proven to be the fastest solver. Nevertheless, in contrast to the results presented in [FNORC08], the LP solver QSopt turns out to be even faster than Z3 for our satisfiability and implication tests. Although we did not do enough experiments to arrive at a final conclusion, the usage of solvers in SPASS(LA) differs from the SMT scenario, because we do not incrementally/decrementally change the investigated LA theory as done by SMT solvers but ask for solving different “small” LA problems containing typically not more that 10–30 (dis)equations. Remarkably, Z3 and Redlog have answered every implication query exactly the same as QSopt has done for the corresponding encompassment matcher existence LP problem, although the latter is only an approximation of the original encompassment condition of the Hierarchic Subsumption Deletion rule.

Recently, Bromberger [Bro12] has developed a specialized LA solver mainly focused on the implication test. This solver relies on the Simplex-based algorithm for SMT [DdM06], adjusted to employ the logical structure of implications. Besides, the new solver uses a sharing method which reduces memory consumption for storing clause constraints, and includes two simple, but quite powerful criteria for detecting unsatisfiable implications. The experimental results given in [Bro12] show that this approach drastically increases the overall efficiency of constraint reasoning within the SUP(LA) concept and outperforms any other solver used for the intended LA task.

## 5.4 Application: Reasoning about Transition Systems

In the following we present two examples showing that our notion on tautology and subsumption deletion is strong enough to decide formal safety properties of transition systems and our implementation SPASS(LA) is able to perform saturations in a reasonable amount of time. Furthermore, the proofs for the two presented properties of the examples are both satisfiability proofs and hence rely on the completeness of the calculus which is an important feature distinguishing our combination approach from others. Both examples are contained in the experimental SPASS(LA) version[1].

The transition system of the water tank controller depicted in Figure 5.2 is meant to keep the level of a water tank below 240 units, provided the initial level at state $S_0$ is less or equal 240 units. There is a non-controllable inflow from the outside of the system that adds at most 40 units per cycle to the water tank and a controllable outflow valve that can reduce the content of the tank by 40 units per cycle.

We model reachability of the transition system by introducing two place predicates in the variables of the water tank level $x_L$ and the inflow $x_i$ for each state and translate the transitions into implications between states. So, the transitions between the states of the automaton are formalized as follows:

- $S_0$ to $S_1$: $\forall u, v. \big(S_0(u,v) \wedge u \geq 200 \rightarrow S_1(u,v)\big)$,
- $S_0$ to $S_3$: $\forall u, v. \big(S_0(u,v) \wedge u < 200 \rightarrow S_3(u,v)\big)$,
- $S_1$ to $S_2$: $\forall u, v, w. \big(S_1(u,v) \wedge w \geq 0 \wedge w \leq 40 \rightarrow S_2(u,w)\big)$,
- $S_2$ to $S_0$: $\forall u, v. \big(S_2(u,v) \rightarrow S_0(u+v-40, v)\big)$,
- $S_3$ to $S_4$: $\forall u, v, w. \big(S_3(u,v) \wedge w \geq 0 \wedge w \leq 40) \rightarrow S_4(u,w)\big)$,
- $S_4$ to $S_0$: $\forall u, v. \big(S_4(u,v) \rightarrow S_0(u+v, v)\big)$,

Normal form translation and abstraction yields the following clauses:

$$\begin{array}{rcl}
u \geq 200 \;\|\; S_0(u,v) & \rightarrow & S_1(u,v) \\
u < 200 \;\|\; S_0(u,v) & \rightarrow & S_3(u,v) \\
w \leq 40,\, w \geq 0 \;\|\; S_1(u,v) & \rightarrow & S_2(u,w) \\
u' \approx u+v-40 \;\|\; S_2(u,v) & \rightarrow & S_0(u',v) \\
w \leq 40,\, w \geq 0 \;\|\; S_3(u,v) & \rightarrow & S_4(u,w) \\
u' \approx u+v \;\|\; S_4(u,v) & \rightarrow & S_0(u',v)
\end{array}$$

The conjecture is translated into the formula

$$\forall u, v. \big(u \leq 240 \rightarrow S_0(u,v)\big) \;\rightarrow\; \exists u', v'. \big(S_0(u',v') \wedge u' > 240\big)$$

meaning that starting with an initial state $S_0$ with a level below 240 units we can reach a state $S_0$ with a level strictly above 240 units. SPASS(LA) finitely saturates this conjecture together with the theory of the transition system without finding the empty clause in less than one second on any reasonable PC hardware. Due to completeness and the minimal model property of superposition with respect to existentially quantified conjunctions of atoms [HW10] this shows that the level of

[1]See `http://spass-prover.org/prototypes/`

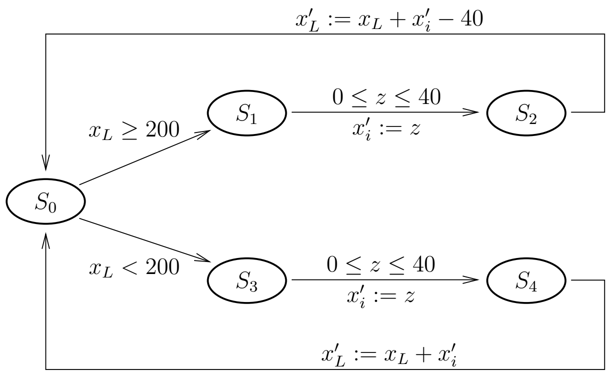


Figure 5.2: Water Tank Controller. (Source of figure: [AKW09a])

the water tank is always below 240 units. Obviously, the hierarchic superposition calculus is complete for this example, because there are no free function symbols at all.

SPASS(LA) may also prove reachability conjectures. For instance, the conjecture

$$\forall u, v\,\big(u \leq 240 \wedge S_0(u, v)\big) \Rightarrow \exists u', v'.\big(S_4(u', v') \wedge u' \leq 160\big)$$

means that starting with an initial state $S_0$ with a level below 240 units we can reach a state $S_4$ with a level below or equal 160 units. SPASS(LA) finitely saturates this conjecture together with the theory of the transition system and finds a proof in less than one second. Due to completeness this shows that the level of the water tank can get 160 units or below.

For a number of classes of transition systems, it can actually be shown that the translation used here always results in a fragment where the hierarchic superposition calculus is complete [Non00, Dim09].

Note that the above clause set is out of reach for current SMT based approaches as their underlying calculus is typically not complete with respect to non-ground clauses and their instantiation based techniques are not able to show satisfiability. For example, Z3 “gives up” on the above (and the below) clause set. Both Z3 and any other Nelson-Oppen style solver are not complete for the two clause sets. The enhanced prover Z3(SP) [dMB08a] is also not complete on both clause sets.

The bakery protocol is aimed at assuring mutual exclusion between two or more processes. We analyze the protocol for two processes; the corresponding automaton is depicted in Figure 5.3. The protocol works as follows. If a process $P_i$ gets into the critical section *crit*$_i$, it takes a number that is greater then the number of the other process and moves from the state *sleep*$_i$ into *wait*$_i$. If the other process does not try to get into the critical section, i.e. if the number of the other process is 0, the process may move on into *crit*$_i$. Otherwise the process that has taken a number first may move into the critical section. The other process waits until the critical section gets free and moves on afterwards. Exiting the critical section signs that the critical section has gotten free by setting the number of the exiting process to 0.

We build the transition system for the two processes as the Cartesian prod-

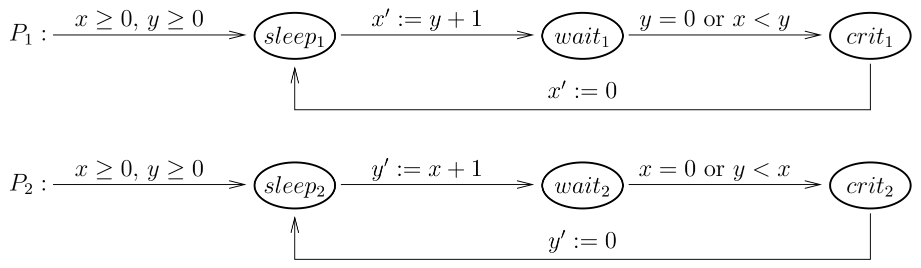


Figure 5.3: Bakery Protocol. (Source of figure: [AKW09a])

uct of the two subautomata representing the processes shown in Figure 5.3 and then model the transitions and states analogous to the water tank example. (For a thorough exposition of the constructing a multiple automata product we refer the reader to [Hen96]). The conjecture for the combined transition system

$$\forall u, v.\big(u \geq 0 \land v \geq 0 \land \mathit{Sl1Sl2}(u, v)\big) \rightarrow \exists u', v'.\mathit{Cr1Cr2}(u', v')$$

states that if $u$ and $v$ are both greater than 0 and the processes in their respective sleep states, then the critical section of both processes can be reached simultaneously. Again, SPASS(LA) saturates this conjecture together with the theory of the two processes without finding the empty clause in less than one second. This shows that the protocol is safe, i.e. the critical sections cannot be reached simultaneously by the both processes.

In [AKW09b] we have also discussed that the SUP(LA) calculus can be applied to analysis of container data structures. The main challenge to make SUP(LA) work on such kind of problems is to ensure sufficient completeness. We used there the well-known example of axiomatization of lists [ABRS09] to show that sufficient completeness can be gained by adding extra clauses, which define extension terms occurring in the axiomatization, and adding extra literals to the axiom clauses to preserve the intended semantics. The obtained complete axiom set is subject to the same superposition based techniques for decidability results as they have been suggested, e.g., in [ABRS09] for the standard superposition calculus. Sufficient completeness can also be obtained using the described encodings for other container data structures, such as arrays, making SUP(LA) a complete calculus for the hierarchic combination of LA over the rationals and arrays.

We have presented an instance of the hierarchic superposition calculus with LA and provided effective algorithms for subsumption and tautology deletion. Compared to other approaches combining LA with first-order logic, the hierarchic superposition calculus is complete, if the actual clause sets enjoys the (local) sufficient completeness criterion. We showed that for the theories of transition systems over LA and container data structures the (local) sufficient completeness criterion can be fulfilled. The calculus is implemented in a first prototype version called SPASS(LA). By two examples we show that it can already be effectively used to decide satisfiability of clause sets that are out of scope for other approaches, in particular SMT-based procedures.

CHAPTER 6

# SUP(NLA): Superposition Modulo Non-Linear Arithmetic

In this chapter we instantiate and refine the hierarchic superposition SUP(T) calculus to hierarchic combination of first-order theory FOL and non-linear arithmetic NLA over the reals including transcendental functions. In Section 6.1 we define the hierarchic specification of the combination of FOL and NLA, and exhibit the essential specialties of SUP(NLA). In Section 6.2, we present a solution to the issues regarding application of SUP(NLA) to the hierarchic FOL(NLA) combination and discuss our implementation of the resulting approach in a system combination of automated theorem prover SPASS and SMT-solver iSAT. In Section 6.3 we present experimental results of applying SPASS(iSAT) to (dis)prove safety properties of various traffic collision avoidance protocols.
The essential parts of this chapter have been presented in [EKS$^+$11].

## 6.1 Introduction

In this chapter we instantiate the hierarchic superposition SUP(T) calculus to hierarchic combination of first-order theory FOL and non-linear arithmetic NLA over the reals including transcendental functions. The combination of FOL and NLA is undecidable, nevertheless we show that under certain conditions SUP(NLA) can be turned into a sound and complete procedure for FOL(NLA) with practically useful applications. In most aspects, SUP(NLA) follows the schema exploited to obtain the SUP(LA) combination, but the undecidability of non-linear arithmetic causes a necessity to take special care of cases when the background NLA-solver is unable to certainly solve problems passed to it. Yet, a comprehensive preprocessing of NLA-problems considerably helps the solver to solve them. We have implemented the resulting approach in a system combination of automated theorem prover SPASS and SMT-solver iSAT. The obtained tool SPASS(iSAT) has been successfully applied to (dis)prove safety properties of various traffic collision avoidance protocols using the very same formalization in a fully automatic, "push-button" manner.

### 6.1.1 Hierarchic Specification of FOL(NLA)

Here we define a hierarchic specification and its ingredients, base specification and body, for hierarchic combination FOL(NLA) of first-order theory FOL with non-linear arithmetic NLA. The notions (re)defined here are valid from now on until the end of Chapter 6.

We write $\mathsf{Sp}$ to denote the signature of NLA defined as follows:

$$\mathsf{Sp} = (\mathsf{R}, \Omega)$$

where:

– $\Omega$ the set of arithmetic operators given as:

$$\begin{array}{lll}\Omega = \{+, -, \times, & & \text{// basic arithmetic functions} \\ & \mathit{abs}, \mathit{min}, \mathit{max}, & \text{// absolute, minimum, and maximum functions} \\ & \mathit{sin}, \mathit{cos}, & \text{// trigonometric functions} \\ & \mathit{exp}, \mathit{nrt}, \mathit{pow}, & \text{// exponential, } n\text{-th root, and power functions} \\ & \leq, <, \approx, >, \geq, & \text{// arithmetic relations} \\ & \mathbb{R}\} & \text{// real numbers (numeric constants)}\end{array}$$

– $\mathsf{R}$ the only arithmetic sort,

with underlying set $\mathcal{X}$ of all variables of the arithmetic sort $\mathsf{R}$.

We set the class $\mathscr{C}$ of base algebras to consist of the standard model $\mathcal{M}$ of non-linear arithmetic (and all algebras isomorphic to it). For the sake of simplicity, we simply write $\mathcal{M}$ in place of $\mathscr{C}$. The model $\mathcal{M}$ maps the arithmetic sort $\mathsf{R}$ to the set $\mathbb{R}$ of all real numbers[1], i.e. $\mathcal{M}(\mathsf{R}) = \mathbb{R}$, and interprets the functions and relations in $\Omega$ as intended. Clearly, the algebra $\mathcal{M}$ is term-generated (as there is a one-to-one correspondence of $\mathcal{M}$ universe's elements to numeric constants in $\mathbb{R} \subset T_\Omega$). We

[1]We ambiguously write $\mathbb{R}$ to denote the subset of $\Omega$ consisting of all arithmetic constants (syntactical objects) and the set of reals (the universe of $\mathcal{M}$).

write Sp to denote the base specification of non-linear arithmetic, defined as

$$\mathsf{Sp} = (\mathsf{Sp}, \mathcal{M})$$

The operators and the sorts of Sp are further extended with free (uninterpreted) operators to $\Omega' \supseteq \Omega$ and $\mathcal{S}' \ni \mathsf{R}$, respectively; the set of arithmetic variables is extended with non-base variables to $\mathcal{X}' \supseteq \mathcal{X}$. Recall, that every non-base variable in $\mathcal{X}' \setminus \mathcal{X}$, if any, is of a free sort $\mathsf{S}'' \in \mathcal{S}' \setminus \mathsf{R}$, whereas free operator symbols may have arguments of the arithmetic sort or range into it. These constitute a FOL(NLA) signature $\Sigma'$:

$$\Sigma' = (\mathcal{S}', \Omega')$$

underlied by the set of variables $\mathcal{X}'$.

Let $Ax'$ be a set of formulae built over $\Sigma'$. The pair

$$\mathsf{Sp}' = (\Sigma', Ax')$$

defines the body of a hierarchic FOL(NLA) specification

$$\mathsf{HSp} = (\mathsf{Sp}, \mathsf{Sp}').$$

As usual, we call operators in $\Omega'' = \Omega' \setminus \Omega$, sorts in $\mathcal{S}'' = \mathcal{S}' \setminus \mathsf{R}$, and axioms $Ax'$ the enrichment. In contrast to the NLA specification Sp, the enrichment

$$(\mathcal{S}'', \Omega'', Ax')$$

is not fixed beforehand and is defined by an input clause set $N$: the symbols in $N$ that do not otherwise occur in Sp define $\mathcal{S}''$ and $\Omega''$. For the sake of simplicity, in the rest we set $Ax' = \emptyset$, which, according to the discussion of alternative formulations of refutational theorem proving for hierarchic theories (see Section 3.2), is not a limitation, but rather a more convenient way of arguing in the context of the combination of FOL with NLA.

Thus, refutational theorem proving for the hierarchic combination FOL(NLA) is aimed at answering a question, whether a given clause set $N$ over the FOL(NLA) signature $\Sigma'$ is inconsistent relative to $\mathcal{M}$, i.e. is there no model of $N$ whose restriction to the NLA signature Sp is (isomorphic to) $\mathcal{M}$. Since $\mathcal{M}$ is the only model of NLA, we write $\models_{\mathcal{M}}$ to denote the entailment relative to the theory of NLA (please, recall Definition 3.25 of the entailment relative to a base theory).

◄ EXAMPLE 6.1

The following is a typical clause over the hierarchic specification HSp:

$$\begin{aligned} D = \big(& M(x_1, y_1, x_2, y_2, t), \delta \geq 0 \rightarrow \\ & M(x_1 + t, y_1 - (cos(t+\delta) - cos(t)), \\ & \quad x_2 - t, y_2 + cos(t+\delta) - cos(t), \\ & \quad t + \delta)\big) \end{aligned}$$

where all $x_i$ and $y_i$, $t$ and $\delta$ are variables, and $M$ a free predicate operator. Abstraction of the clause $D$ yields

$$\begin{aligned} D' = \big(& \delta \geq 0, \\ & x_1' \approx x_1 + t, y_1' \approx y_1 - (cos(t+\delta) - cos(t)), \\ & x_2' \approx x_2 - t, y_2' \approx y_2 + cos(t+\delta) - cos(t), \\ & t' \approx t + \delta \parallel \\ & M(x_1, y_1, x_2, y_2, t) \rightarrow \\ & M(x_1', y_1', x_2', y_2', t')\big) \end{aligned}$$

■

### 6.1.2 SUP(NLA) Application Issues

Next, we consider the most important peculiarities of the application of the hierarchic calculus to FOL(NLA) clauses. Similar to the case of SUP(LA), see Section 5.1.2, application of the hierarchic versions of the standard rules (Hierarchic Equality Resolution, Hierarchic Superposition, etc..) to FOL(NLA) clauses does not require NLA reasoning at all, because:

– the rules apply independently from premises' constraints (as the rules' conditions are stipulated exclusively by the free parts of premises), and

– the conclusion's constraint is simply an instance of the conjunction of the premises' constraints, which can be easily computed by concatenating the constraints and applying the free part unifier; since the unifier is simple, its application reduces to identifying a subset of the constraints' variables.

In contrast, the Constraint Refutation rule and hierarchic reduction rules are further subjected to restrictions on the premises' constraints, that require performing calculations with respect to the background theory. Recall the Constraint Refutation rule, Definition 3.31:

$$\mathcal{I}\frac{\Lambda_1 \parallel \rightarrow \quad \dots \quad \Lambda_n \parallel \rightarrow}{\square}$$

where $(\Lambda_1 \parallel \rightarrow), \dots, (\Lambda_n \parallel \rightarrow) \models_{\mathscr{C}} \bot$. In Section 3.3 we have shown that the condition of the rules holds if and only if the existential closure $\exists \vec{x}^i . \bigwedge \Lambda_i$ of the constraint of *at least one base clause* among $C_1 = (\Lambda_1 \parallel \rightarrow), \dots, C_n = (\Lambda_n \parallel \rightarrow)$, where $\vec{x}^i = var(\Lambda_i)$, is valid in the base theory (satisfiable by every base algebra). For the NLA theory considered here, which is compact and where the clauses, e.g., do not share parameters, it is sufficient to consider the case $n = 1$, thereby reducing the condition of the rule to simply testing whether the existential closure $\exists \vec{x}^1 . \bigwedge \Lambda_1$ is satisfiable in the single model $\mathcal{M}$ of NLA.

Consider the Hierarchic Tautology Deletion rule, Definition 3.96:

$$\mathcal{R}\frac{\Lambda \parallel \Gamma \rightarrow \Delta}{}$$

if

(i) $\models \Gamma \rightarrow \Delta$, or

(ii) $\exists \vec{x} . \bigwedge \Lambda \models_{\mathscr{C}} \bot$, for $\vec{x} = var(\Lambda)$.

According to the second condition the existential closure of the base part of the premise has to be unsatisfiable in the base theory, which for the NLA theory case reduces to checking satisfiability of $\exists \vec{x} . \bigwedge \Lambda$ in the single model $\mathcal{M}$ of NLA.

Recall that the hierarchic superposition calculus is complete if the base theory is compact and all extension terms are sufficiently defined, Theorems 3.82 and 3.94. Compactness follows from the fact that the base theory is given by a standard model $\mathcal{M}$ of NLA. However, even if all extension terms are sufficiently defined, in practice it happens that satisfiability of an NLA constraint $\Lambda$ cannot be decided. But in order to establish a base clause $\Lambda \parallel \rightarrow$ to be a contradiction, NLA reasoning on $\Lambda$ must *not* return the inconclusive answer UNKNOWN. This is indispensable for proving a conjecture. Taking this into account, we use the following more practical formulation of the completeness theorem.

◄ THEOREM 6.2
SUP(NLA)
Practical Completeness

*Let $N$ be a (locally) sufficiently complete set of FOL(NLA) clauses. Then $N$ is unsatisfiable if SUP(NLA) derives $\square$; $N$ is satisfiable, if the SUP(NLA) calculus terminates and the saturated set $N^*$ does not contain $\square$ nor a clause $\Lambda \parallel \rightarrow$.*

This completeness theorem takes care of the fact that in practice an NLA procedure will not be able to decide the satisfiability of a constraint, in general. Then it may happen that clauses of the form $\Lambda \parallel \rightarrow$ can neither be refuted nor deleted and have to be kept. Thus being able to practically decide constraint satisfiability is crucial for precision. For termination the same applies to constraint implication, needed for subsumption. Consider the Hierarchic Subsumption Deletion rule, Definition 3.97:

$$\mathcal{R}\frac{\Lambda_1 \parallel \Gamma_1 \rightarrow \Delta_1 \qquad \Lambda_2 \parallel \Gamma_2 \rightarrow \Delta_2}{\Lambda_1 \parallel \Gamma_1 \rightarrow \Delta_1}$$

where, for a simple matcher $\sigma$,

(i) $\Gamma_1\sigma \subseteq \Gamma_2$, $\Delta_1\sigma \subseteq \Delta_2$,

(ii) $\models_{\mathscr{C}} \forall \vec{x}.\exists \vec{y}.(\bigwedge \Lambda_2 \rightarrow \bigwedge \Lambda_1\sigma)$, for $\vec{x} = var(C_2) \cap \mathcal{X}$ and $\vec{y} = var(\Lambda_1\sigma) \setminus var(C_2)$,

(iii) $(\Lambda_2 \parallel \Gamma_2 \rightarrow \Delta_2) \neq \square$.

The third condition of the Hierarchic Subsumption Deletion rule is stipulated by the fact, that the rule missing the condition would allow subsuming an empty clause by an unsatisfiable base clause. The condition is intended rather for *theoretical purposes* and does not make much sense *in practice* as all/most theorem provers stop executing whenever an empty clause is derived. Note that $y \in var(\Lambda_1\sigma) \setminus var(C_2)$ implies $y \notin (var(\Gamma_1) \cup var(\Delta_1))$. In contrast to the theory of LA, the base specification Sp of the non-linear arithmetic with transcendental functions does not enable quantifier elimination, and hence such variables $\vec{y}$ cannot be all eliminated in $\Lambda_1\sigma$, in general. Nevertheless, we have developed a series of transformations to be applied to the constraints $\Lambda_1\sigma$ and $\Lambda_2$, which leads to an equivalent formula

$$\forall \vec{x}'.\exists \vec{y}'.(\bigwedge \Lambda_2' \rightarrow \bigwedge \Lambda_1') \quad ◄ (6.1)$$

with $\vec{x}' \subseteq \vec{x}$ and $\vec{y}' \subseteq \vec{y}$. If $\vec{y}' = \emptyset$, then formula (6.1) reduces to

$$\forall \vec{x}'.(\bigwedge \Lambda_2' \rightarrow \bigwedge \Lambda_1')$$

which is valid in the theory of NLA whenever its negation

$$\begin{aligned} &\neg\big(\forall \vec{x}'.(\bigwedge \Lambda_2' \rightarrow \bigwedge \Lambda_1')\big) \\ \models\!\mid\ &\exists \vec{x}'.(\bigwedge \Lambda_2' \wedge \neg \bigwedge \Lambda_1') \end{aligned}$$

is unsatisfiable, and this can be already checked by iSAT. Otherwise, if $\vec{y}'$ is nonempty, checking the validity of formula (6.1) in the theory of NLA is out of the scope of iSAT possibilities due to quantifier alternation. Nonetheless, the *linear relaxation* of (6.1) yields an LA formula which can be examined by an appropriate SMT solver for *linear* arithmetic; in our experimentation we have used Z3 for this purpose. We discuss the mentioned transformations in detail in Section 6.2.2.

## 6.2 Constraint Solving

Mapping iSAT to the above reasoning tasks in the context of SUP(NLA), there are two problems to be solved: the formula with quantifier alternation needed for the redundancy check which is not an SMT formula and the case iSAT answers UNKNOWN. The former is solved by first reducing the number of existentially quantified variables through equational propagation. Then we test whether a linear relaxation of the result where the non-linear functions are considered as uninterpreted functions can already be proven (here we use Z3 [dMB08c]). If this does fail and there are no existentially quantified variables left, we pass the result to iSAT. All simplification is implemented on the SPASS side. The latter problem is solved by an extension of iSAT called *strong satisfaction check*. A disadvantage of the interval based reasoning approach of iSAT is the loss of precision when intervals are propagated through equations. Equations frequently induce point solutions. Here, without the strong satisfaction check, iSAT would terminate with an UNKNOWN answer, providing narrow intervals. The strong satisfaction extension then takes those narrow intervals and tries to compute a certificate for a (point) solution. It turns out that this extension turned most of iSAT's UNKNOWN answers in the SUP(NLA) context into definite YES answers.

Here we give an overview of the main steps of the iSAT algorithm and describe the core idea of the strong satisfaction check, Section 6.2.1. In Section 6.2.2 we present the constraint transformation techniques aimed at simplifying the inputs of iSAT, including single constraint and constraint implication formulae, arising as the background theory tasks during a SPASS(iSAT) run on FOL(NLA) problems.

### 6.2.1 iSAT Procedure. Strong Satisfaction Check

In order to explain how iSAT works, we need to introduce few auxiliary notions typical for SMT solving in the domain of NLA theory. After having disposed of this preliminary step we continue with presenting the iSAT procedure and the strong satisfaction check.

**Preliminaries.** A ground substitution $\nu : \mathcal{X} \to \mathbb{Q}$ mapping arithmetic variables $\mathcal{X}$ to reals $\mathbb{R}$ is called a ***base assignment***. An ***interval assignment/valuation*** $\iota$ is a mapping from base variables $\mathcal{X}$ to intervals over $\mathbb{R}$. A base assignment $\nu$ is said to be ***contained*** in an interval assignment $\iota$ if $x\nu \in x\iota$ for all $x \in dom(\nu)$; in this case, the interval assignment $\iota$ is said to ***contain*** the base assignment $\nu$.

An ***SMT formula with respect to the theory of non-linear arithmetic over the reals***, or simply ***SMT formula*** for short, is an arbitrary quantifier-free Boolean combination of atoms over the NLA signature $\Sigma$, for instance

$$\phi = \left(sin(y^2) \leq 0.1 \to (x \leq 0 \vee z > \sqrt{x^2 + y^2})\right).$$

An SMT formula $\phi$ is ***satisfiable*** if there exists a base assignment $\nu$ such that $var(\phi) \subseteq dom(\nu)$ and $\mathcal{M} \models \phi\nu$, where $\mathcal{M}$ is the model of NLA; in this case, the base assignment $\nu$ is called ***satisfying*** for $\phi$. An SMT formula $\phi$ is ***inconsistent under an interval assignment*** $\iota$ if and only if $\iota$ contains no base assignment satisfying $\phi$, i.e. if $\phi$ is false at every point in the interval assignment $\iota$. Please, note that this language is very expressive as total division $z = x/y$ can be coded as multiplication $x = y \cdot z \wedge y \neq 0$, the constant $\pi$ can be coded as the solution to the variable $x$ in the formula $\phi \equiv x > 3.1 \wedge x < 3.2 \wedge sin(x) = 0$ and the integers as solutions to the variable $y$ in $\phi \wedge sin(x \cdot y) = 0$.

A ***primitive constraint*** $e$ is an NLA atom containing at most one total[1] arithmetical operation symbol and at most three variables, for instance: $x \geq sin(y)$, $x = y + z$, $z < 3.7$. A primitive constraint $e$ is called ***consistent under an interval assignment*** $\iota$ if $\iota$ contains a base assignment $\nu$ satisfying $e$, i.e. if $x\nu \in x\iota$, for all $x \in var(e)$, and $\mathcal{M} \models e\nu$.

An SMT formula is said to be in ***conjunctive form***, or ***CF*** for short, if it is a conjunction of disjunctions of primitive constraints[2]; the disjunctions of a CF SMT formula are called ***clauses***. Thus, an CF SMT formula $\phi$ is satisfiable if and only if there exists some base assignment $\nu$ satisfying at least one primitive constraint in each clause of $\phi$. Please note, that given an interval valuation $\iota$, a CF SMT formula $\phi$ can be inconsistent under $\iota$, even if in each clause of $\phi$ there is a primitive constraint $e$ consistent under $\iota$.

**The iSAT procedure.** iSAT is an SMT solver for the theory of non-linear arithmetic involving transcendental functions like exponential and trigonometric functions which is *undecidable* in general. The iSAT algorithm is a generalization of the Davis-Putnam-Logemann-Loveland (DPLL) procedure [DP60, DLL62] (with clause learning) using *interval constraint propagation* (ICP) [BG06]. iSAT manipulates *interval valuations (assignments)* of the variables by alternating *deduction* and *splitting* phases, interspersed with *non-chronological backtracking* whenever an empty interval valuation is detected. Next we give a brief presentation of the iSAT algorithm and strong satisfaction check (for a formal exposition, we refer an interested reader to [FHR$^+$06, FHT$^+$07]; a very detailed and easy-to-read survey is given in [Tei12]).

iSAT accepts an arbitrary input SMT formula and automatically rewrites it into conjunctive form. All variables in the given input formula have to be accompanied with their respective bounded intervals[3]. Nevertheless, if the variable bounds cannot be estimated, the lower and upper interval bounds can be chosen arbitrarily small and arbitrarily large, respectively.

During the ***deduction*** phase, iSAT searches for clauses in which all but one atom are inconsistent under the current interval valuation. Such atoms are called ***unit atoms***. Then, the discovered unit atoms are used to deduce new interval bounds by repeatedly applying ICP until no new interval can be obtained. This step is essentially similar to the *unit propagation* in SAT solving. Since ICP may yield infinitely many interval contraction steps, the process of deducing new intervals is stopped if the difference between the values of new and old bounds is below some user-defined *progress parameter* $\delta$.

[1]Totality is needed to obviate the issue with undefined values of partial operations. Practically, this is not a strong requirement as most common partial arithmetic operators can be expressed by their inverse operation, for example, the constraint $y = 1/x$ in which $1/x$ is undefined for $x = 0$ can be rephrased as $y \cdot x = 1 \wedge x \neq 0$.

[2]Any SMT formula can be converted into an equi-satisfiable CF SMT formula in linear time [Her10].

[3]From a practical point of view, the requirement for variable bounds is not a strong restriction as variables encoding such physical quantities as distance, attitude, or velocity are naturally limited.

If the interval of some variable $x$ becomes empty, then a ***conflict resolution*** procedure is called to analyze the reason for the conflict in a very similar way as in the DPLL procedure. If the conflict cannot be resolved iSAT stops execution with result NOsignalizing that the given formula is unsatisfiable. Otherwise, a conflict clause is built (learnt) from the reason of the conflict and added to the formula, followed by a non-chronological backjump by at least one level back, undoing some of the decisions and their accompanying deductions performed so far.

iSAT finds a ***solution***, if at least one atom in each clause is satisfied by every point in the current interval valuation. In this case the solver stops with result YES. In general, equations like $x = y \cdot z$ can only be satisfied by point intervals. However, reaching such point intervals by ICP cannot be guaranteed for continuous domains. One option to mitigate this problem is to stop the search when all intervals have a width smaller than a certain threshold, the so-called *minimum splitting width* $\varepsilon$. The resulting interval valuation can be considered as an ***approximate solution***. Since the given problem cannot be decided, iSAT answers UNKNOWN.

iSAT performs a ***decision*** by splitting an interval, if after the deduction phase neither a conflict nor an (approximate) solution has been found. A decision heuristics is used to select one of the intervals whose width is still greater than or equal to the minimum splitting width $\varepsilon$. The search is then resumed using this new interval bound which potentially triggers new deductions as described above.

**Strong Satisfaction Check.** As we have already discussed, the ICP methods used by iSAT core to deduce a satisfying interval valuation is very weak to find a solution over a continuous domain for an SMT formula containing equations, because equations can only be satisfied by point intervals, in general. The drawback of this uncertainty is that it may happen that clauses of the form $\Lambda \parallel \rightarrow$ can neither be refuted nor deleted and have to be kept, which may lead to an inability to prove a conjecture (refute a given clause set). This issue is of a particular importance for our SPASS(iSAT) case study, as the axiomatization of the experimental problems (see Section 6.3) heavily involves equational NLA atoms. Here we briefly describe the essential algorithmic details of the strong satisfaction check aimed at mitigating the problem of ICP weakness to deduce point solutions. For a formal exposition of the strong satisfaction check algorithm and essential technical details of its steps we refer the reader to [FHT+07] as well as [Ked08, EKS+11].

The strong satisfaction check procedure takes an approximate solution $\iota$ (an interval valuation, a "box" with interval assignments for every variable), that is provided every time iSAT returns an UNKNOWN answer, and tries to build a certificate of the existence of a solution to the current CF SMT formula $\phi$ in the following way:

- In the first step, the procedure selects from each clause in $\phi$ a primitive constraint consistent under the approximate solution[1] $\iota$, putting them into the set $S = E(S) \cup I(S)$ consisting of the selected equations and inequations, respectively.

[1]Please, note that a set of primitive constraints, each of which is consistent under the interval valuation $\iota$, might still be inconsistent under $\iota$

– In the next step, treating all equations $E(S)$ contained in the set $S$ as assignments, the procedure orients them such that:

  • no variable is defined more than once (for instance, in $\{x = sin(y),\ x = y + z\}$ the variable $x$ is doubly defined, and only once in $\{x = sin(y),\ y = x - z\}$); and

  • the assignments are acyclic (for instance, in $\{x = sin(y),\ y = x - z\}$ there is a cycle "$x$ depends on $y$, and $y$ depends on $x$", but not in $\{x = sin(y),\ z = x - y\}$).

  After having oriented the equations in $E(S)$, every variable in $E(S)$ becomes either *defined* (by an assignment in which it occurs on the left-hand-side) or *undefined*.

– In the last step, the intervals $y\iota$ of the undefined variables $y$ are safely propagated via the system of reoriented equations building this way a new interval valuation $\iota'$ such that intervals $x\iota'$ are computed for every variable $x \in var(E(S))$.

In Section 6.3 we show that the strong satisfaction check is one of the crucial ingredients ensuring a conclusive behavior of SPASS(iSAT) on our experimental problems.

### 6.2.2 Constraint and Implication Simplification

The iSAT procedure is be able to decide the satisfiability of a constraint, in general, returning the inconclusive answer UNKNOWN. For disproving, i.e. having SUP(NLA) terminate on a set of clauses without finding an empty clause, it is crucial to be able to decide (sufficiently many) subsumption implication checks. For our case study the first-order part of the clauses is inherently recursive, for instance, the continuous linear movement clause is of the form

$$\Lambda \parallel L(x_1, y_1, x_2, y_2, t) \to L(x_1', y_1', x_2', y_2', t')$$

where the positive and negative occurrence of the $L$ literal have a first-order unifier. It turns out that SUP(NLA) termination can only be achieved by successful subsumption applications that require a pipelining of NLA simplification related to the form of constraints generated by the superposition calculus. Here, we present a series of simplifications that helps to gain a precise terminating behavior of SUP(NLA) for our experimental case study of collision avoidance protocols (Section 6.3).

Every time a new clause $\Lambda \parallel \Gamma \to \Delta$ is derived, the following simplifications are repeatedly performed on the constraint $\Lambda$ of the clause:

– ***Constant propagation***: if $\Lambda$ contains a literal

$$L = (ax \approx b),$$

  where $a, b \in \mathbb{R}$ and $x \in \mathcal{X}$, then the substitution $\sigma = [x \mapsto b/a]$ is applied onto every literal in the constraint $\Lambda$ except $L$. Moreover, if the variable $x$ does not occur in the free part $\Gamma \to \Delta$ of the clause, then the literal $L$ is deleted after applying $\sigma$ onto literals of $\Lambda$.

- ***Deletion of duplicates***: if the constraint contains syntactically equivalent literals, say
$$\Lambda = \{L_1, \dots, L_k, L_{k+1}, \dots, L_n\},$$
where $L_i = L_j$, for all $k+1 \le i, j \le n$, then only one of them is kept yielding a modified constraint $\Lambda'$:
$$\Lambda' = \{L_1, \dots, L_k, L_{k+1}\}.$$
- ***Product distribution over sum***: every term
$$s = t \cdot (\sum_{i=1}^{n} t_i)$$
occurring in $\Lambda$, where $t, t_1, \dots, t_n \in T_\Omega(\mathcal{X})$, is transformed to the term
$$s' = \sum_{i=1}^{n} (t \cdot t_i).$$
- ***Reduction of homogeneous summands***: every term
$$s = \sum_{i=1}^{n} (a_i t) + t'$$
occurring in $\Lambda$, where $a_1, \dots, a_n \in \mathbb{R}$ and $t, t' \in T_\Omega(\mathcal{X})$, is reduced to
$$s' = at + t',$$
where $a$ is computed as the sum of all coefficients $a_i$: $a = \sum_{i=1}^{n} a_i$.
- ***Reduction of homogeneous multipliers***: every term
$$s = \prod_{i=1}^{k'} b_i \cdot \prod_{i=1}^{k} t^{a_i} \cdot t'$$
occurring in $\Lambda$, where $a_1, \dots, a_k, b_1, \dots, b_{k'} \in \mathbb{R}$ and $t, t' \in T_\Omega(\mathcal{X})$, is reduced to
$$s' = bt^a \cdot t',$$
where $a$ is computed as the sum of all coefficients $a_i$: $a = \sum_{i=1}^{k} a_i$, and $b$ is computed as the product of all coefficients $b_i$: $b = \prod_{i=1}^{k'} b_i$.

If the Hierarchic Subsumption Deletion rule, Definition 3.97, is to be applied to two abstracted FOL(NLA) clauses $C_1 = \Lambda_1 \parallel \Gamma_1 \to \Delta_1$ and $C_2 = \Lambda_2 \parallel \Gamma_2 \to \Delta_2$, we check the encompassment condition of the rule
$$\models_{\mathscr{C}} \forall \vec{x}. \exists \vec{y}. (\bigwedge \Lambda_2 \to \bigwedge \Lambda_1 \sigma),$$
where $\sigma$ is the free part simple subsumption matcher, $\vec{x} = var(C_2) \cap \mathcal{X}$, and $\vec{y} = var(\Lambda_1 \sigma) \setminus var(C_2)$, by consecutively performing the following steps:

1. *Exhaustive simplification* is first applied to the implication's succedent $\Lambda_1 \sigma$, which comprises the following steps:
   a) first, every variable-to-constant assignment $x \approx a$ occurring in the implication's antecedent $\Lambda_2$, where $x \in \mathcal{X}$ and $a \in \mathbb{R}$, is propagated onto the implication's succedent $\Lambda_1 \sigma$; then,
   b) the transformed succedent is repeatedly reduced by applying each of the above simplifications; and finally,

   c) every literal of the simplified succedent that has a syntactical equivalent in $\Lambda_2$ is deleted.

   The above three steps yield a constraint $\Lambda_1'$. Note that this simplification technology has a very high potential for the SUP(NLA) calculus because constraints of newly generated clauses are always copies of the constraints from the parent clauses subject to a unifier mapping variables to variables.

2. *Linear Relaxation.* Every occurrence of the transcendental function symbols *sin, cos, exp, nrt, pow* is replaced in $\Lambda_1'$ and $\Lambda_2$ with fresh *uninterpreted* function symbols *sin′*, *cos′*, *exp′*, *nrt′*, *pow′*, respectively. After that, every occurrence of terms with the top symbol being one of *abs, min,* or *max* is recursively rewritten in the following way:

$$\begin{aligned} abs(t) &\rightsquigarrow ite(t \geq 0,\, t,\, -t) \\ min(s_1,\dots,s_n) &\rightsquigarrow ite(s_1 \leq min(s_2,\dots,s_n),\, s_1,\, min(s_2,\dots,s_n)) \\ max(s_1,\dots,s_n) &\rightsquigarrow ite(s_1 \geq max(s_2,\dots,s_n), s_1, max(s_2,\dots,s_n)), \end{aligned}$$

   where "*ite*" stands for the operator "*if then else*" available in most SMT systems. These result in two modified constraints $\overline{\Lambda}_2$ and $\overline{\Lambda}_1$. Then we check if the formula

$$\begin{aligned} &\neg\big(\forall\,\vec{x}.\exists\vec{y}'.(\bigwedge\overline{\Lambda}_2 \rightarrow \bigwedge\overline{\Lambda}_1)\big) \\ &\models\!\mid\, \exists\vec{x}.\forall\vec{y}'.(\bigwedge\overline{\Lambda}_2 \wedge \neg\bigwedge\overline{\Lambda}_1) \end{aligned} \qquad ◂\ (6.2)$$

   where $\vec{y}' = var(\Lambda_1) \setminus var(C_2) = var(\overline{\Lambda}_1) \setminus var(C_2)$ is unsatisfiable in the model of linear arithmetic plus *uninterpreted functions* by passing it to the SMT solver Z3. If Z3 affirms unsatisfiability of Formula (6.2), then the initial encompassment condition holds.

3. Otherwise, i.e. if Z3 does not affirm unsatisfiability of Formula (6.2) at the previous step, and if, besides, $\vec{y}' = var(\Lambda_1) \setminus var(C_2)$ is empty, then we call iSAT to check whether the formula

$$\begin{aligned} &\neg\big(\forall\,\vec{x}.(\bigwedge\Lambda_2 \rightarrow \bigwedge\Lambda_1')\big) \\ &\models\!\mid\, \exists\vec{x}.(\bigwedge\Lambda_2 \wedge \neg\bigwedge\Lambda_1') \end{aligned} \qquad ◂\ (6.3)$$

   is unsatisfiable in the theory of NLA. If iSAT affirms unsatisfiability of Formula (6.3), then the initial encompassment condition holds.

In Section 6.3 we show that the above schema exploited for checking the encompassment condition is the crucial ingredient which guarantees finding a model by SPASS(iSAT) on our experimental problems.

## 6.3 Application: Reasoning about Collision Avoidance Protocols

The application scenario (Section 6.3) is a collision avoidance protocol for moving objects (e.g., robots, aircrafts). The idea of the protocol is to prevent a collision, more precisely, a situation where the objects get too close. In order to achieve this goal, the movement of the objects is put into a maneuver mode, once their distance falls below a given limit. The maneuver mode then takes care by performing appropriate movement in form of sine curves to get the objects across each other and release them afterwards to their initial behavior. We studied the following three scenarios: (i) two objects in 2D space starting with linear movement, (ii) two objects in 3D space starting with linear movement, and (iii) two objects in 3D space starting with arbitrary movement. For all scenarios we can fully automatically prove and disprove (given different parameters) the collision freeness of the protocol. The protocol and the collision freeness property are modelled by a set of FOL(NLA) Horn clauses such that this set is satisfiable iff the protocol is collision free. If not, an unsatisfiability SUP(NLA) proof yields a counterexample.

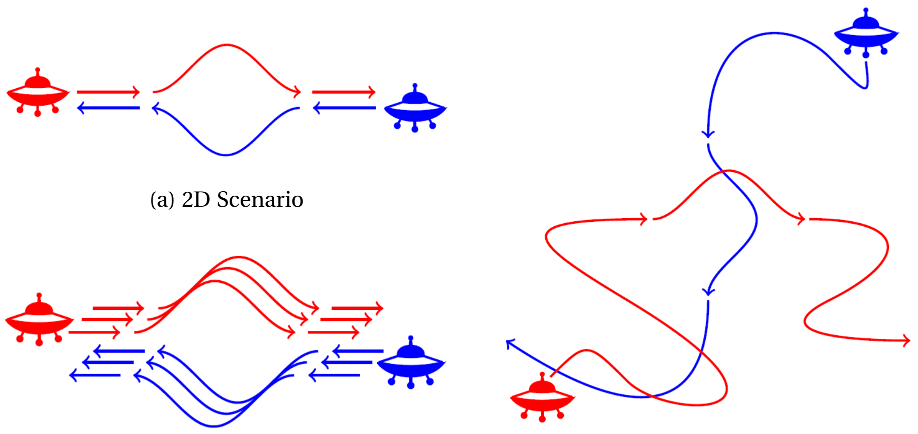
The idea behind the protocol for the basic 2D scenario is the following: initially there are two objects moving on straight lines in 2D-space (see Figure 6.1a); when the distance between the objects gets less or equal to some fixed value, they start maneuvering by *sin*-like trajectories such that at the beginning of the maneuver one of them goes up, the other goes down. The maneuver lasts for one period of *sin*, after that the objects continue straight line moving. Depending on the three involved parameters initial distance, distance to start the maneuver, and distance required for safety, the protocol yields or does not yield a collision. More precisely, the behavior of the objects is modelled by a set of FOL(NLA) Horn clauses such that the minimal model of those clauses describes exactly the set of reachable states by the protocol. The minimal Horn clause model is identical to the SUP(NLA) model assumption for a set of Horn clauses. First-order predicates are used to model the reachable states.

**Axiomatisation of AACAS, 2D.** In the 2D case two aircrafts are initially moving in straight lines, one from left to the right, another – from right to the left. When distance between them gets small enough, they start manoeuvring in a *sin*-like trajectories, such that the aircraft, which is higher at the manoeuvre start, is going first up, then down, and another – first down, then up. During the manoeuvre distance between the aircrafts is required to remain safe:

– Safe start:
$|x_1 - x_2| \geq 10, t = 0 \parallel \to L(x_1, y_1, x_2, y_2, x_1, y_1, x_2, y_2, t)$.

– Linear movement:
$\delta \geq 0, t' = t + \delta, x_1' = x_1 + \delta, x_2' = x_2 - \delta, \sqrt{(x_1' - x_2')^2 + (y_1 - y_2)^2} \geq 6 \parallel$
$L(x_1, y_1, x_2, y_2, x_1^0, y_1^0, x_2^0, y_2^0, t) \to$
$L(x_1', y_1, x_2', y_2, x_1^0, y_1^0, x_2^0, y_2^0, t')$.

– Switching to the manoeuvre mode:
$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} = 6, t_m = 0 \parallel$


(a) 2D Scenario

(b) 3D Scenario: linear initial movement

(c) 3D Scenario: arbitrary initial movement

Figure 6.1: Movement Scenarios. (Source of figures: [EKS+11])

$L(x_1, y_1, x_2, y_2, x_1^0, y_1^0, x_2^0, y_2^0, t) \to$
$M(x_1, y_1, x_2, y_2, t_m, p, x_1^0, y_1^0, x_2^0, y_2^0, t).$

– Manoeuvring.

First aircraft is higher than the second one:
$y_1 \geq y_2, \delta \geq 0, p \geq 3.14, p \leq 3.15, cos(\frac{p}{2}) = 0, t'_m = t_m + \delta, t'_m \leq 2 \cdot p,$
$\Delta_y = cos(t_m) - cos(t'_m), x'_1 = x_1 + \delta, y'_1 = y_1 + \Delta_y, x'_2 = x_2 - \delta, y'_2 = y_2 - \Delta_y \parallel$
$M(x_1, y_1, x_2, y_2, t_m, p, x_1^0, y_1^0, x_2^0, y_2^0, t) \to$
$M(x'_1, y'_1, x'_2, y'_2, t'_m, p, x_1^0, y_1^0, x_2^0, y_2^0, t).$

First aircraft is lower than the second one:
$y_1 < y_2, \delta \geq 0, p \geq 3.14, p \leq 3.15, cos(\frac{p}{2}) = 0, t'_m = t_m + \delta, t'_m \leq 2 \cdot p,$
$\Delta_y = cos(t_m) - cos(t'_m), x'_1 = x_1 + \delta, y'_1 = y_1 - \Delta_y, x'_2 = x_2 - \delta, y'_2 = y_2 + \Delta_y \parallel$
$M(x_1, y_1, x_2, y_2, t_m, p, x_1^0, y_1^0, x_2^0, y_2^0, t) \to$
$M(x'_1, y'_1, x'_2, y'_2, t'_m, p, x_1^0, y_1^0, x_2^0, y_2^0, t).$

– Distance requirement:
$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} < 4 \parallel M(x_1, y_1, x_2, y_2, t_m, p, x_1^0, y_1^0, x_2^0, y_2^0, t) \to$
$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \geq 4.$

The protocol is safe if there is no reachable state in the minimal model that causes a collision. This is obviously not a first-order property but can be attacked by superposition based reasoning as long as the safety condition has the closed form $\exists \vec{x}\, \phi$ where all first-order predicates in $\phi$ occur solely positively. In this case the negation of the safety condition results in a set of purely negative clauses. Then adding such a set to a set of Horn clauses the following holds [HW10]: (i) if $\square$ is derived, then the safety condition does not hold and a counter example can be extracted from the superposition proof; (ii) if SUP(NLA) terminates and neither $\square$ nor a clause $\Lambda \parallel \to$ is derived, then the safety condition holds (see also Theorem 6.2). This is exactly the way we proved (disproved) non-collision, by adding clauses of the form

$$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2} < 4 \parallel M(x_1,y_1,x_2,y_2,t_m,p,x_1^0,y_1^0,x_2^0,y_2^0,t) \rightarrow$$

to the axiomatization.

**Axiomatisation of AACAS, 3D.** The 3D instance basic protocol extends the movement to 3D, where initially the objects are moving in parallel horizontal planes, and during the maneuver they are changing their heights following a sine-wave trajectory, see Figure 6.1b. The advanced 3D scenario adds to the basic scenario arbitrary initial 3D movement, see Figure 6.1c. Where in particular for this drawing we assume that the ends and starts of the arrows are time synchronization points between the objects. During the manoeuvre distance between the aircrafts is required to remain safe.

- Safe start:
  $|x_1-x_2| \geq 10 \parallel \rightarrow L(x_1,y_1,z_1,v_{x_1},v_{y_1},x_2,y_2,z_2,v_{x_2},v_{y_2},x_1,y_1,z_1,x_2,y_2,z_2)$.
- Movement in an arbitrary direction:
  $(v'_{x_1}=1, v'_{y_1}=0 \vee v'_{x_1}=0, v'_{y_1}=1 \vee v'_{x_1}=-1, v'_{y_1}=0 \vee v'_{x_1}=0, v'_{y_1}=-1)$,
  $(v'_{x_2}=1, v'_{y_2}=0 \vee v'_{x_2}=0, v'_{y_2}=1 \vee v'_{x_2}=-1, v'_{y_2}=0 \vee v'_{x_2}=0, v'_{y_2}=-1)$,
  $\sqrt{(x'_1-x'_2)^2+(y'_1-y'_2)^2+(z'_1-z'_2)^2} \geq 6 \parallel$
  $L(x_1,y_1,z_1,v_{x_1},v_{y_1},x_2,y_2,z_2,v_{x_2},v_{y_2},x_1^0,y_1^0,z_1^0,x_2^0,y_2^0,z_2^0) \rightarrow$
  $L(x'_1,y'_1,z'_1,v'_{x_1},v'_{y_1},x'_2,y'_2,z'_2,v'_{x_2},v'_{y_2},x_1^0,y_1^0,z_1^0,x_2^0,y_2^0,z_2^0)$.
- Switching to the manoeuvre mode:
  $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2} = 6, t_m = 0 \parallel$
  $L(x_1,y_1,z_1,v_{x_1},v_{y_1},x_2,y_2,z_2,v_{x_2},v_{y_2},x_1^0,y_1^0,z_1^0,x_2^0,y_2^0,z_2^0) \rightarrow$
  $M(x_1,y_1,z_1,v_{x_1},v_{y_1},x_2,y_2,z_2,v_{x_2},v_{y_2},t_m,p,x_1^0,y_1^0,z_1^0,x_2^0,y_2^0,z_2^0)$.
- Manoeuvring. First AC is higher than the second one:
  $z_1 \geq z_2, \delta \geq 0, t'_m = t_m+\delta, x'_1 = x_1 + v_{x_1}\cdot\delta, y'_1 = y_1 + v_{y_1}\cdot\delta, x'_2 = x_2 + v_{x_2}\cdot\delta$,
  $y'_2 = y_2 + v_{y_2}\cdot\delta, p \geq 3.14, p \leq 3.15, cos(\frac{p}{2}) = 0, t'_m \leq 2\cdot p, \Delta_z = cos(t_m) - cos(t'_m)$,
  $z'_1 = z_1 + \Delta_z, z'_2 = z_2 - \Delta_z \parallel$
  $M(x_1,y_1,z_1,v_{x_1},v_{y_1},x_2,y_2,z_2,v_{x_2},v_{y_2},t_m,p,x_1^0,y_1^0,z_1^0,x_2^0,y_2^0,z_2^0) \rightarrow$
  $M(x'_1,y'_1,z'_1,v_{x_1},v_{y_1},x'_2,y'_2,z'_2,v_{x_2},v_{y_2},t'_m,p,x_1^0,y_1^0,z_1^0,x_2^0,y_2^0,z_2^0)$.

  First AC is lower than the second one:
  $z_1 < z_2, \delta \geq 0, t'_m = t_m+\delta, x'_1 = x_1 + v_{x_1}\cdot\delta, y'_1 = y_1 + v_{y_1}\cdot\delta, x'_2 = x_2 + v_{x_2}\cdot\delta$,
  $y'_2 = y_2 + v_{y_2}\cdot\delta, p \geq 3.14, p \leq 3.15, cos(\frac{p}{2}) = 0, t'_m \leq 2\cdot p, \Delta_z = cos(t_m) - cos(t'_m)$,
  $z'_1 = z_1 - \Delta_z, z'_2 = z_2 + \Delta_z \parallel$
  $M(x_1,y_1,z_1,v_{x_1},v_{y_1},x_2,y_2,z_2,v_{x_2},v_{y_2},t_m,p,x_1^0,y_1^0,z_1^0,x_2^0,y_2^0,z_2^0) \rightarrow$
  $M(x'_1,y'_1,z'_1,v_{x_1},v_{y_1},x'_2,y'_2,z'_2,v_{x_2},v_{y_2},t'_m,p,x_1^0,y_1^0,z_1^0,x_2^0,y_2^0,z_2^0)$.
- Distance requirement:
  $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2} < 4 \parallel$
  $M(x_1,y_1,z_1,v_{x_1},v_{y_1},x_2,y_2,z_2,v_{x_2},v_{y_2},t_m,p,x_1^0,y_1^0,z_1^0,x_2^0,y_2^0,z_2^0) \rightarrow$
  $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2} \geq 4$.

For all three scenarios we have proved and disproved safety by modifying the above mentioned parameters accordingly. Table 6.1 shows the timing for SPASS(iSAT) runs on all scenarios in seconds. The input as well as the output files are available

| | 2D | 3D | |
|---|---|---|---|
| | linear | linear | arbitrary |
| Proof | 10 | 9 | 41 |
| Model | 6 | 6 | 58 |

Table 6.1: Time Statistics.

| | | None | SSC | IA | IS | CS |
|---|---|---|---|---|---|---|
| Result | | Proof | Unknown | Proof | Proof | - |
| Time (sec.) | | 45 | 520 | 38 | 290 | out |
| Constr. | sat | 356 | 20 | 364 | 355 | - |
| | unsat | 11 | 16 | 19 | 11 | - |
| | unknown | 0 | 440 | 0 | 1 | - |
| Impl. | holds | 281 | 296 | 256 | 280 | - |
| | not holds | 3040 | 3071 | 4218 | 6047 | - |
| | unknown | 0 | 3 | 0 | 29 | - |

Table 6.2: NLA Simplification Impact for Proof Finding

| | | None | SSC | IA | IS | CS |
|---|---|---|---|---|---|---|
| Result | | Model | Unknown | - | Model | - |
| Time (sec.) | | 35 | 36 | out | 188 | out |
| Constr. | sat | 367 | 20 | - | 368 | - |
| | unsat | 31 | 28 | - | 29 | - |
| | unknown | 0 | 350 | - | 3 | - |
| Impl. | holds | 296 | 296 | - | 297 | - |
| | not holds | 3073 | 3071 | - | 6160 | - |
| | unknown | 1 | 3 | - | 44 | - |

Table 6.3: NLA Simplification Impact for Model Finding

from the SPASS homepage (`www.spass-prover.org/prototypes`). For each scenario we have ran one parameter setting where the protocol is collision free and SPASS(iSAT) finds a model and one setting where the objects collide, leading to a proof found by SPASS(iSAT). All runs were have been performed on computers equipped with Intel X5460 CPUs, 8 GB of main memory running Linux. Concerning all experiments we have done, more than 95% of the time has been spent by iSAT for solving NLA constraint proof obligations.

Table 6.2 shows the impact of our simplification pipeline for finding a proof in an unsafe parameter setting for the 3D arbitrary movement scenario. The first row shows which NLA simplification techniques developed in this paper have been omitted. So “None” means we have applied all, “SSC” means we have disabled the strongsat extension, “IA” means we have disabled the linear abstraction for implication testing, “IS” means we have disabled constraint simplification on the generated implication problems, and “CS” means we have disabled the basic constraint simplification techniques. Then the rest of the table shows the results of these settings on the testing of NLA problems during the run. If the basic constraint techniques have been disabled, SPASS(iSAT) does not terminate on the problem. Note that disabling an NLA simplification technique typically results in extra clauses that cannot be subsumed nor detected as a tautology. Therefore, the set of generated and kept clauses for the different cases is different.

Table 6.3 shows the respective impact of our simplification pipeline for finitely saturating the clause set in a safe parameter setting for the same scenario. Here both disabling CS or IA leads to non-termination. Disabling SSC leads to termination where the saturated clause set contains a clause $\Lambda \parallel \rightarrow$ for which iSAT without the strongsat extension cannot decide satisfiability of the constraint and returns UNKNOWN.

Concerning the generated NLA constraint problems, Section 6.2, we have also tried to attack them by applying state-of-the-art computer algebra systems. To this end we have replaced iSAT by Maple[1]. But this approach has not been as successful as the iSAT combination as in many cases Maple has not been able to find a solution and therefore the overall solving process for the considered experiments here has failed. Redlog [DS97] shows a similar behavior. By appropriate approximations of the transcendental functions a reasonable portion of the constraints can be decided, however, in particular the crucial constraints (empty clauses, subsumption check) turn out to be specifically hard.

Collision avoidance protocols have been studied as benchmarks for various hybrid system analysis and verification tools (e.g. [HHMWT00, PC07, TPS98]). They are related to our collision avoidance protocol. However, these results are hard to compare as the models differ. For example, we have also considered an explicit 3D model where the above approaches all have developed 2D models.

We have developed the first sound and complete combination of FOL(NLA) including an implementation, where in particular, we can cope with UNKNOWN results by an NLA procedure (Theorem 6.2, Tables 6.2, 6.3). In order to decrease the number of UNKNOWN answers when executing the SUP(NLA) calculus we have suggested dedicated simplification techniques. All together with an implementation via SPASS(iSAT) the approach supports fully automatic verification of various scenarios of a non-trivial collision avoidance protocol. We are confident that by continuing the development of the suggested simplification techniques the performance of the overall procedure can be further significantly increased.

[1]Maple is a computer algebra system published by Maplesoft. For more details refer to `http://www.maplesoft.com/products/maple/`

CHAPTER 7

# Conclusion and Future Work

## Summary

In this work we have extended the state of the art of reasoning in hierarchic combinations of the free first-order logic FOL and a background theory T. Built on the Hierarchic Superposition approach of Bachmair, Ganzinger, and Waldmann, we have substantially advanced previous (rather scarce and solitary) results on SUP(T). To the best of our knowledge, we are the first who have pioneered implementing the hierarchic calculus SUP(T), where for the background theory T we have considered linear and non-linear arithmetic, given rise to a new direction in applied research concerned with practical aspects of implementation of the hierarchic superposition calculus. In detail, our contributions are as follows:

– We have developed new versions of the hierarchic sufficient completeness criterion and the hierarchic redundancy criterion, which take into account the background theory and are therefore substantially more effective than the ones existed before.

– We have developed two basic reduction rules Hierarchic Tautology Deletion and Hierarchic Subsumption Deletion, which comply with the new hierarchic redundancy criterion and constitute a good basis for lifting other "flat" reduction rules to the level of hierarchic first-order combinations.

– We have developed and investigated a noval Local Sufficient Completeness criterion, that expresses a much more relaxed condition sufficient to guarantee completeness of SUP(T). The most advantageous feature of the Local Sufficient Completeness criterion is that it suggests a transformation technique of turning an input problem initially lacking sufficient completeness to a sufficiently complete one.

– We have shown that SUP(T) can be turned into a decision procedure for the ground FOL(T) fragment and the non-ground BSHE(GBST) class of Bernays-

Schönfinkel Horn clauses with equality and ground base sort terms. One of the ingredients of obtaining the decidability results is the Basification procedure, presented by us, whose intention is to make an input problem locally sufficiently complete, such that SUP(T) assuredly finds a proof if one exists. The decidability result for the BSHE(GBST) class allows reasoning about and querying ontologies with arithmetical facts.

– We have implemented the SUP(LA) calculus for the hierarchic combination of FOL with theory of linear arithmetic LA in system combinations Spass(LA) of the automated first-order theorem prover Spass and several background LA-solvers. Having several implementations of SUP(LA) helped us to better understand practical issues of the calculus and identify potentially prolific improvements of used implementational concepts.

– By examples we have shown that SUP(LA) can already be effectively used to decide satisfiability of clause sets that are out of scope for other approaches, in particular SMT-based procedures.

– We have also implemented the SUP(NLA) calculus for the hierarchic combination of FOL with theory of non-linear arithmetic NLA over the reals including transcendental functions in a system combination Spass(iSAT), where for the NLA reasoning tasks we use an SMT solver for non-linear arithmetic iSAT.

– We have elaborated the issues related to the hierarchic SUP(NLA) calculus which are due to undecidability of NLA with transcendental functions. To tackle this problem, we have presented a series of simplification and approximation techniques for treating NLA formulae that has eventually enabled a complete automatic behavior of Spass(iSAT) on various scenarios of collision avoidance protocols.

On one hand, our research delivers fundamental contributions: the crucial part of it is devoted to proving theoretical aspects of the Hierarchic Superposition calculus; on the other hand, we have shown that SUP(T) can be successfully used for practical applications. From a methodological point of view, the dissertation is useful as it (i) provides rigorous formal proofs in detail exposing the most essential gists of the presented approach, (ii) assists with schemata of showing completeness and decidability of the SUP(T) calculus applied to various FOL(T) fragments, and (iii) suggests transformation techniques which, coped with the SUP(T) calculus, yield a complete procedure. For this reason, we recommend the dissertation to everybody concerned with SUP(T)-based hierarchic refutational theorem proving, including scientists and students.

## Future Work

Possible directions for future research include:

– The superposition calculus SUP can be turned into a decision procedure for a number of decidable first-order fragments, e.g., [BGW93, JMW98, ARR03, HSG04, BE07, BE08, ABRS09], and is thus a good basis for actually proving decidability of fragments and obtaining efficient implementations. In this connection, one of possible research topics is to extend the obtained results for SUP(T) to the combination of several theories. As long as these theories can be represented in FOL (e.g., lists, arrays) such a combination is straightforward. On the other hand the hierarchic approach cannot be easily extended to deal simultaneously with several different theories outside the free part in a straight forward way as it is the case for SMT based procedures using the Nelson-Oppen method, even if queries are ground. Here further research is needed.

- Another topic for future investigation is the combination of FOL over explicit finite domain clause sets with a combination of background theories $\mathcal{T} = T_1 \cup \ldots \cup T_n$. A restrictive superposition calculus has been proven to be a decision procedure for FOL over finite domain fragment [HW07]. The sweet point of the combination of FOL over finite domain with $\mathcal{T}$, but with non-constant function symbols ranging into the free sort, is that any clause set (even non-ground) over the fragment is sufficiently complete by the transformations suggested in [HW07]. This is due to the fact that the cardinality clause

  $$x \approx a_1 \vee \cdots \vee x \approx a_m$$

  representing the finite domain $\{a_1, \ldots, a_m\}$ is reflected by the clauses

  $$f(\vec{x}) \approx a_1 \vee \cdots \vee f(\vec{x}) \approx a_m \qquad \text{for any } f \in \Omega'' \setminus \{a_1, \ldots, a_m\}$$

  which imply sufficient completeness for any clause set over this fragment.
- Our experiments with the FOL(LA) combination involving base parameters[1] show that very often SUP(LA) produces constraints of a regular shape. From this perspective, taking advantages of automatic generation of invariants studied in [FKW12b, FKW12a] seems to be beneficial for SUP(LA) applied to FOL(LA) fragments with finite domains.
- Development of more sophisticated reduction rules. The flat superposition-based reasoning drastically benefits from exploiting the reduction rule called Contextual Rewriting [NN93, Wei01, Wis12]. We expect that lifting the rule to the level of the hierarchic FOL(T) combination would allow to achieve terminating saturation by SUP(T) on a large class of FOL(T) clauses.
- Engineering of a universal interface between SPASS and an arbitrary background theory solver, bringing the realization of the principle of modularity of the hierarchic SUP(T) calculus to a highest degree.
- Development of effective data structures and specialized algorithms supporting efficient practical implementations of the hierarchic SUP(T) calculus. The recent work [Bro12] of Bromberger (supervised by Sturm, and Weidenbach) devoted to adaption of Dutertre and de Moura's Simplex-based algorithm [DdM06] for SUP(LA) specific LA reasoning tasks has shown that a deep integration of a tuned LA procedure into SPASS(LA) can save 94% of the runtime spent by the solver on reduction elimination.
- Practical application of SUP(T) for analysis of programs, encryption protocols, data transmission protocols, right policies, hardware, hybrid systems, etc.

[1]Not reported in this dissertation.

Until recently, all research of the superposition modulo theory calculus SUP(T) has been done by members of our research group[1]. We expect that this dissertation and recent developments of our research group will attract considerable attention of other scientists around the world exciting them for further investigation of the hierarchic superposition calculus which will abundantly result in new theoretical achievements in the area of Automated Deduction and successful applications of the SUP(T) approach to solving practically important problems.

[1]The “Automation of Logic” group at the Max Planck Institute for Informatics, Saarbrücken, Germany. Actually, two of the three inventors of SUP(T), namely, Harald Ganzinger and Uwe Waldmann, are recent or present members of the group.

# Bibliography

[ABRS05] Alessandro Armando, Maria Paola Bonacina, Silvio Ranise, and Stephan Schulz. On a rewriting approach to satisfiability procedures: Extension, combination of theories and an experimental appraisal. In Bernhard Gramlich, editor, *FroCoS 2005*, volume 3717 of *LNCS*, pages 65–80. Springer, 2005.

[ABRS09] Alessandro Armando, Maria Paola Bonacina, Silvio Ranise, and Stephan Schulz. New results on rewrite-based satisfiability procedures. *ACM Trans. Comput. Log.*, 10(1):129–179, 2009.

[AKW09a] Ernst Althaus, Evgeny Kruglov, and Christoph Weidenbach. Superposition modulo linear arithmetic SUP(LA). In Silvio Ghilardi and Roberto Sebastiani, editors, *Frontiers of Combining Systems: 7th International Symposium, FroCoS 2009*, volume 5749 of *Lecture Notes in Artificial Intelligence*, pages 84–99, Trento, Italy, September 2009. Springer.

[AKW09b] Ernst Althaus, Evgeny Kruglov, and Christoph Weidenbach. Superposition modulo linear arithmetic SUP(LA). Report of SFB/TR 14 AVACS 53, December 2009. http://www.avacs.org.

[ARR03] Alessandro Armando, Silvio Ranise, and Michaël Rusinowitch. A rewriting approach to satisfiability procedures. *Information and Computation*, 183(2):140–164, 2003.

[BDEK07] Bernd Becker, Christian Dax, Jochen Eisinger, and Felix Klaedtke. LIRA: Handling constraints of linear arithmetics over the integers and the reals. In Werner Damm and Holger Hermanns, editors, *Computer Aided Verification, 19th International Conference, Proceedings*, volume 4590 of *Lecture Notes in Computer Science*, pages 307–310. Springer, 2007.

[BE07] Maria Paola Bonacina and Mnacho Echenim. Rewrite-based satisfiability procedures for recursive data structures. *Electronic Notes in Theoretical Computer Science, ENTCS*, 174(8):55–70, 2007.

[BE08] Maria Paola Bonacina and Mnacho Echenim. On variable-inactivity and polynomial tau-satisfiability procedures. *Journal of Logic and Computation*, 18(1), 2008.

[BE10] Maria Paola Bonacina and Mnacho Echenim. Theory decision by decomposition. *Journal of Symbolic Computation*, 45(2):229–260, 2010.

[BFT08] Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. ME(LIA) - model evolution with linear integer arithmetic constraints. In *LPAR 2008*, volume 5330 of *LNCS*, pages 258–273. Springer, 2008.

[BG90] Leo Bachmair and Harald Ganzinger. On restrictions of ordered paramodulation with simplification. In *CADE*, pages 427–441, 1990.

[BG91] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. Research Report MPI-I-91-208, Max-Planck-Institut für Informatik, September 1991. Revised version in the Journal of Logic and Computation 4, 3 (1994), pp. 217–247.

[BG94] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994. Revised version of Max-Planck-Institut für Informatik technical report, MPI-I-91-208, 1991.

[BG01] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 19–99. Elsevier and MIT Press, 2001.

[BG06] F. Benhamou and L. Granvilliers. Continuous and interval constraints. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, Foundations of Artificial Intelligence, chapter 16, pages 571–603. Elsevier, 2006.

[BGN$^+$06] Maria Paola Bonacina, Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Decidability and undecidability results for nelson-oppen and rewrite-based decision procedures. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning, IJCAR 2006*, volume 4130 of *Lecture Notes in Computer Science*, pages 513–527. Springer, 2006.

[BGW92] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Theorem proving for hierarchic first-order theories. In Hélène Kirchner and Giorgio Levi, editors, *Algebraic and Logic Programming, Third International Conference, Volterra, Italy, September 2-4, 1992, Proceedings*, volume 632 of *Lecture Notes in Computer Science*, pages 420–434. Springer, 1992.

[BGW93] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Superposition with simplification as a decision procedure for the monadic class with equality. In Georg Gottlob, Alexander Leitsch, and Daniele Mundici, editors, *Computational Logic and Proof Theory, Third Kurt Gödel Colloquium*, volume 713 of *LNCS*, pages 83–96. Springer, August 1993.

[BGW94] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing (AAECC)*, 5(3/4):193–212, April 1994. Earlier Version: Theorem Proving for Hierarchic First-Order Theories, in Giorgio Levi and Héléne Kirchner, editors, *Algebraic and Logic Programming, Third International Conference*, LNCS 632, pages 420–434, Volterra, Italy, September 2–4, 1992, Springer-Verlag.

[BLdM09] Maria Paola Bonacina, Christopher Lynch, and Leonardo Mendonça de Moura. On deciding satisfiability by DPLL($\Gamma + \mathcal{T}$) and unsound theorem proving. In Renate A. Schmidt, editor, *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, volume 5663 of *LNCS*, pages 35–50. Springer, 2009.

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

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[Bon10] Maria Paola Bonacina. On theorem proving for program checking: historical perspective and recent developments. In Temur Kutsia, Wolfgang Schreiner, and Maribel Fernández, editors, *Proceedings of the 12th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming,PPDP-12*, pages 1–12. ACM, 2010.

[Bro12] Martin Bromberger. Adapting the simplex algorithm for superposition modulo linear arithmetic. Bachelor thesis, Universität des Saarlandes, Saarbrücken, 2012.

[BSST09] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, February 2009.

[BT03] Peter Baumgartner and Cesare Tinelli. The model evolution calculus. In *Proceedings of the 9th International Conference on Automated Deduction,CADE-19*, volume 2741 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2003.

[BT11] Peter Baumgartner and Cesare Tinelli. Model evolution with equality modulo built-in theories. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *Proceedings of the 23rd International Conference on Automated Deduction, CADE-23*, volume 6803 of *Lecture Notes in Computer Science*, pages 85–100. Springer, 2011.

[BW13] Peter Baumgartner and Uwe Waldmann. Hierarchic superposition with weak abstraction. In Maria Paola Bonacina, editor, *CADE-24*, volume 7898 of *LNAI*. Springer, 2013.

[DdM06] Bruno Dutertre and Leonardo Mendonça de Moura. A fast linear-arithmetic solver for DPLL(T). In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4144 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2006.

[Dim09] Dilyana Dimova. On the translation of timed automata into first-order logic, 2009. Supervisors: A. Fietzke, C. Weidenbach.

[DLL62] Martin Davis, George Logemann, and Donald W. Loveland. A Machine Program for Theorem Proving. *Communications of the ACM, CACM*, 5(7):394–397, 1962.

[DM79] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, 1979.

[dMB07] Leonardo Mendonça de Moura and Nikolaj Bjørner. Efficient e-matching for SMT solvers. In Frank Pfenning, editor, *Proceedings of the 21st International Conference on Automated Deduction, CADE-21*, volume 4603 of *Lecture Notes in Computer Science*, pages 183–198. Springer, 2007.

[dMB08a] Leonardo Mendonça de Moura and Nikolaj Bjørner. Engineering DPLL(T) + saturation. In *IJCAR 2008*, volume 5195 of *LNCS*, pages 475–490. Springer, 2008.

[dMB08b] Leonardo Mendonça de Moura and Nikolaj Bjørner. Model-based theory combination. *Electronic Notes in Theoretical Computer Science, ENTCS*, 198(2):37–49, 2008.

[dMB08c] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *TACAS 2008*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[DMB11] Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: Introduction and applications. *Communications of the ACM*, 54(9):69–77, 2011.

[DP60] M. Davis and H. Putnam. A Computing Procedure for Quantification Theory. *Journal of the ACM*, 7(3):201–215, 1960.

[DS95] Andreas Dolzmann and Thomas Sturm. Simplification of quantifier-free formulas over ordered fields. *Journal of Symbolic Computation*, 24:209–231, 1995.

[DS97] Andreas Dolzmann and Thomas Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.

[EK06] Jochen Eisinger and Felix Klaedtke. Don't care words with an application to the automata-based approach for real addition. In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification, 18th International Conference, Proceedings*, volume 4144 of *Lecture Notes in Computer Science*, pages 67–80. Springer, 2006.

[EKK$^+$11] Andreas Eggers, Evgeny Kruglov, Stefan Kupferschmid, Karsten Scheibler, Tino Teige, and Christoph Weidenbach. Superposition modulo non-linear arithmetic. Report of SFB/TR 14 AVACS 80, August 2011. http://www.avacs.org.

[EKS$^+$11] Andreas Eggers, Evgeny Kruglov, Karsten Scheibler, Stefan Kupferschmid, Tino Teige, and Christoph Weidenbach. SUP(NLA) – combining superposition and non-linear arithmetic. In Viorica Sofronie-Stokkermans and Cesare Tinelli, editors, *Frontiers of Combining Systems, 8th International Symposium, FroCos 2011*, Lecture Notes in Computer Science. Springer, 2011. Accepted.

[FHR$^+$06] Martin Fränzle, Christian Herde, Stefan Ratschan, Tobias Schubert, and Tino Teige. Interval constraint solving using propositional SAT solving techniques. In Youssef Hamadi and Lucas Bordeaux, editors, *Proceedings of the CP 2006 First International Workshop on the Integration of SAT and CP Techniques*, pages 81–95, 2006.

[FHT$^+$07] Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex Boolean structure. *JSAT*, 1(3–4):209–236, 2007.

[FHW10] Arnaud Fietzke, Holger Hermanns, and Christoph Weidenbach. Superposition-based analysis of first-order probabilistic timed automata. In *Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, LPAR'10, pages 302–316, Berlin, Heidelberg, 2010. Springer-Verlag.

[Fit90] Melvin Fitting. *First-order logic and automated theorem proving.* Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[FKW12a] Arnaud Fietzke, Evgeny Kruglov, and Christoph Weidenbach. Automatic generation of inductive invariants by SUP(LA). Research Report MPI-I-2012-RG1-002, Max-Planck Institute for Informatics, Saarbrücken, Germany, March 2012.

[FKW12b] Arnaud Fietzke, Evgeny Kruglov, and Christoph Weidenbach. Automatic generation of invariants for circular derivations in SUP(LA). In *Proceedings of the 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 7180 of *Lecture Notes in Computer Science*, pages 197–211. Springer, March 2012.

[FNORC08] Germain Faure, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Sat modulo the theory of linear arithmetic: Exact, inexact and commercial solvers. In Hans Kleine Büning and Xishun Zhao, editors, *Theory and Applications of Satisfiability Testing - SAT*, volume 4996 of *Lecture Notes in Computer Science*, pages 77–90. Springer, 2008.

[FW11] Arnaud Fietzke and Christoph Weidenbach. Superposition as a decision procedure for timed automata. In Stefan Ratschan, editor, *Proceedings of the Fourth International Conference on Mathematical Aspects of Computer and Information Sciences*, MACIS 2011, pages 52–62, 2011.

[GBT07] Yeting Ge, Clark Barrett, and Cesare Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In Frank Pfenning, editor, *Proceedings of the 21st International Conference on Automated Deduction, CADE-21*, volume 4603 of *Lecture Notes in Computer Science*, pages 167–182. Springer, 2007.

[Gie01] Martin Giese. Incremental closure of free variable tableaux. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proceedings of the First International Joint Conference on Automated Reasoning, IJCAR 2001*, volume 2083 of *Lecture Notes in Computer Science*, pages 545–560. Springer, 2001.

[Gil84] Robert Gilmer. *Commutative Semigroup Rings*. Chicago Lectures in Mathematics. University of Chicago Press, 1984.

[GSSW06] Harald Ganzinger, Viorica Sofronie-Stokkermans, and Uwe Waldmann. Modular proof systems for partial functions with Evans equality. *Information and Computation*, 204(10):1453–1492, 2006.

[Hen96] Thomas Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, pages 278–292, Washington, DC, USA, 1996. IEEE Computer Society.

[Her10] Christian Herde. *Efficient Solving of Large Arithmetic Constraint Systems with Complex Boolean Structure: Proof Engines for the Analysis of Hybrid Discrete–Continuous Systems.* Doctoral dissertation, Carl von Ossietzky Universität Oldenburg, 2010.

[HHMWT00] Thomas Henzinger, Benjamin Horowitz, Rupak Majumdar, and Howard Wong-Toi. Beyond hytech: Hybrid systems analysis using interval numerical methods. In Nancy Lynch and Bruce Krogh, editors, *HSCC 200*, volume 1790 of *LNCS*, pages 130–144. Springer Berlin / Heidelberg, 2000.

[HR91] Jieh Hsiang and Michaël Rusinowitch. Proving refutational completeness of theorem-proving strategies: The transfinite semantic tree method. *Journal of the ACM*, 38(3):559–587, 1991.

[HSG04] Ullrich Hustadt, Renate A. Schmidt, and Lilia Georgieva. A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1:251–276, 2004.

[Hue80] Gérard P. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.

[HW07] Thomas Hillenbrand and Christoph Weidenbach. Superposition for finite domains. Research Report MPI-I-2007-RG1-002, Max-Planck Institute for Informatics, Saarbruecken, Germany, April 2007.

[HW10] Matthias Horbach and Christoph Weidenbach. Superposition for fixed domains. *ACM Transactions on Computational Logic*, 11(4):1–35, 2010.

[IJSS08] Carsten Ihlemann, Swen Jacobs, and Viorica Sofronie-Stokkermans. On local reasoning in verification. In C. R. Ramakrishnan and Jakob Rehof, editors, *Proceedings of TACAS 2008*, volume 4963 of *LNCS*, pages 265–281. Springer, 2008.

[ISS09] Carsten Ihlemann and Viorica Sofronie-Stokkermans. System description: H-pilot. In Renate A. Schmidt, editor, *Proceedings of the 22nd International Conference on Automated Deduction,CADE-22*, volume 5663 of *Lecture Notes in Computer Science*, pages 131–139. Springer, 2009.

[ISS10] Carsten Ihlemann and Viorica Sofronie-Stokkermans. On hierarchical reasoning in combinations of theories. In *Proceedings of the 5th International Joint Conference on Automated Reasoning, IJCAR 2010*, volume 6173 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 2010.

[JMW98] Florent Jacquemard, Christoph Meyer, and Christoph Weidenbach. Unification in extensions of shallow equational theories. In Tobias Nipkow, editor, *Rewriting Techniques and Applications, 9th International Conference, RTA-98*, volume 1379 of *LNCS*, pages 76–90. Springer, 1998.

[Ked08] Nadine Keddis. Strong satisfaction. Bachelorthesis, Albert-Ludwigs-Universität Freiburg, September 2008.

[Kru08] Evgeny Kruglov. Superposition modulo linear arithmetic. Master's thesis, Universität des Saarlandes, March 2008.

[KV07] Konstantin Korovin and Andrei Voronkov. Integrating linear arithmetic into superposition calculus. In Jacques Duparc and Thomas A. Henzinger, editors, *CSL 2007*, volume 4646 of *LNCS*, pages 223–237. Springer, 2007.

[KW11] Evgeny Kruglov and Christoph Weidenbach. SUP(T) decides first-order logic fragment over ground theories. In Stefan Ratschan, editor, *Proceedings of the Fourth International Conference on Mathematical Aspects of Computer and Information Sciences*, MACIS 2011, pages 126–148, 2011.

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

[Lan93] Serge Lang. *Algebra (3. ed.)*. Addison-Wesley, 1993.

[LW93] Rüdiger Loos and Volker Weispfenning. Applying linear quantifier elimination. *Comput. J.*, 36(5):450–462, 1993.

[LW07] Michel Ludwig and Uwe Waldmann. An extension of the knuth-bendix ordering with lpo-like properties. In *Proceedings of the 14th international conference on Logic for programming, artificial intelligence and reasoning*, LPAR'07, pages 348–362, Berlin, Heidelberg, 2007. Springer-Verlag.

[MMZ$^{+}$01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In Jan Rabaey, editor, *Proceedings of the 38th Design Automation Conference, DAC 2001*, pages 530–535. ACM, 2001.

[NN93] Pilar Nivela and Robert Nieuwenhuis. Saturation of first-order (constrained) clauses with the *saturate* system. In Claude Kirchner, editor, *Proceedings of the 5th International Conference on Rewriting Techniques and Applications, RTA-93*, volume 690 of *Lecture Notes in Computer Science*, pages 436–440. Springer, 1993.

[Non00] Andreas Nonnengart. Hybrid systems verification by location elimination. In Nancy A. Lynch and Bruce H. Krogh, editors, *HSCC 2000*, volume 1790 of *Lecture Notes in Computer Science*, pages 352–365. Springer, 2000.

[NOT06] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving sat and sat modulo theories: From an abstract davis–putnam–logemann–loveland procedure to DPLL(T). *J. ACM*, 53:937–977, November 2006.

[NR01] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 371–443. Elsevier and MIT Press, 2001.

[PC07] André Platzer and Edmund Clarke. The image computation problem in hybrid systems model checking. In Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo, editors, *HSCC 2007*, volume 4416 of *LNCS*, pages 473–486. Springer Berlin / Heidelberg, 2007.

[Pug91] William Pugh. The omega test: a fast and practical integer programming algorithm for dependence analysis. In Joanne L. Martin, editor, *Proceedings Supercomputing'91, 1991*, pages 4–13. IEEE Computer Society / ACM, 1991.

[PW06] Virgile Prevosto and Uwe Waldmann. SPASS+T. In Geoff Sutcliffe, Renate Schmidt, and Stephan Schulz, editors, *ESCoR: FLoC'06 Workshop on Empirically Successful Computerized Reasoning*, volume 192, pages 18–33, 2006.

[Rüm08a] Philipp Rümmer. *Calculi for Program Incorrectness and Arithmetic*. PhD thesis, University of Gothenburg, 2008.

[Rüm08b] Philipp Rümmer. A constraint sequent calculus for first-order logic with linear integer arithmetic. In *Proceedings, 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 5330, pages 274–289, 2008.

[RW69] George Robinson and Lawrence Wos. Paramodulation and theorem proving in first order theories with equality. *J. ACM*, pages 135–150, 1969.

[Sch89] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., 1989.

[Seb07] Roberto Sebastiani. Lazy satisability modulo theories. *JSAT*, 3(3-4):141–224, 2007.

[SKW07] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[SS05] Viorica Sofronie-Stokkermans. Hierarchic reasoning in local theory extensions. In *Proceedings of the 20th International Conference on Automated Deduction, CADE-20*, volume 3632 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 2005.

[SWW10] Martin Suda, Christoph Weidenbach, and Patrick Wischnewski. On the saturation of yago. In *Proceedings of the 5th international conference on Automated Reasoning*, IJCAR'10, pages 441–456, Berlin, Heidelberg, 2010. Springer-Verlag.

[Tei12] Tino Teige. *Stochastic Satisfiability Modulo Theories: A Symbolic Technique for the Analysis of Probabilistic Hybrid Systems*. Doctoral dissertation, Carl von Ossietzky Universität Oldenburg, Department of Computing Science, Germany, 2012. Supervisors: Prof. Dr. Martin Fränzle and Prof. Dr.-Ing. Holger Hermanns.

[TPS98] Claire J. Tomlin, George J. Pappas, and Shankar Sastry. Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, April 1998.

[Wal01] Uwe Waldmann. Superposition and chaining for totally ordered divisible abelian groups. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *IJCAR 2001*, volume 2083 of *LNAI*, pages 226–241. Springer, 2001.

[Wal02] Uwe Waldmann. Cancellative abelian monoids and related structures in refutational theorem proving (Part I). *Journal of Symbolic Computation*, 33(6):777–829, June 2002.

[WDF$^+$09] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Martin Suda, and Patrick Wischnewski. Spass version 3.5. In *CADE-22*, volume 5663 of *LNAI*, pages 140–145. Springer, 2009.

[Wei94] Volker Weispfenning. Quantifier elimination for real algebra - the cubic case. In Malcolm A. H. MacCallum, editor, *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 258–263. ACM, 1994.

[Wei97] Volker Weispfenning. Quantifier elimination for real algebra - the quadratic case and beyond. *Appl. Algebra Eng. Commun. Comput.*, 8(2):85–101, 1997.

[Wei01] Christoph Weidenbach. Combining superposition, sorts and splitting. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume 2, chapter 27, pages 1965–2012. Elsevier, 2001.

[Wis12] Patrick Wischnewski. *Efficient Reasoning Procedures for Complex First-Order Theories.* Doctoral dissertation, Universität des Saarlandes, Saarbrücken, November 2012.

[WW12] Christoph Weidenbach and Patrick Wischnewski. Satisfiability checking and query answering for large ontologies. In Pascal Fontaine, Renate Schmidt, and Stephan Schulz, editors, *PAAR-2012: IJCAR'12 Workshop on Practical Aspects of Automated Reasoning*, pages 163–177, 2012.

[ZS00] Hantao Zhang and Mark E. Stickel. Implementing the davis-putnam method. *Journal of Automated Reasoning*, 24(1/2):277–296, 2000.

# Index