AAECC 5, 193–212 (1994)

AAECC
Applicable Algebra in Engineering, Communication and Computing

Refutational Theorem Proving for Hierarchic First-Order Theories

Leo Bachmair[1], **Harald Ganzinger**[2], **Uwe Waldmann**[2]

[1] Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794, USA, leo@sbcs. sunysb.edu
[2] Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, {hg,uwe}@mpi-sb.mpg.de

Received November 16, 1992; revised version May 17, 1993

Abstract. We extend previous results on theorem proving for first-order clauses with equality to hierarchic first-order theories. Semantically such theories are confined to conservative extensions of the base models. It is shown that superposition together with variable abstraction and constraint refutation is refutationally complete for theories that are sufficiently complete with respect to simple instances. For the proof we introduce a concept of approximation between theorem proving systems, which makes it possible to reduce the problem to the known case of (flat) first-order theories. These results allow the modular combination of a superposition-based theorem prover with an arbitrary refutational prover for the primitive base theory, whose axiomatic representation in some logic may remain hidden. Furthermore they can be used to eliminate existentially quantified predicate symbols from certain second-order formulae.

Keywords: Theorem proving, Resolution, Superposition, Constraints, Quantifier elimination, Term rewriting, Algebraic specification

1 Introduction

The motivation to investigate theorem proving for hierarchic specifications is similar to the reasons to develop constraint logic programming (Jaffar and Lassez [10]). Given a (refutational) theorem prover or a constraint solver for a base theory, we want to be able to use it in a modular fashion, like a subroutine in a program; whereas the axiomatic structure of the base specification ought to remain hidden – in fact, need not even be expressible in first-order logic. Like Bürckert [6, 7] we consider not only Horn clauses but arbitrary first-order clauses. (On the other hand, we obtain only refutation proofs, not an enumeration of all solutions or a complete set of solved forms as one would expect of a programming language.) Moreover we consider a framework with equality, thus providing the possibility to define new functions.

We extend the techniques of Bachmair and Ganzinger [4] for theorem proving with equational first-order clauses to hierarchic first-order specifications with equality over a many-sorted signature. If a set of clauses satisfies a property similar to sufficient completeness, we can embed every refutationally complete theorem prover for a compact constraint specification in a refutationally complete theorem prover for a hierarchic specification.

The paper is organized as follows: Sect. 2 serves mainly to clarify the notations used in the paper; we repeat some basic definitions concerning flat and hierarchic algebraic specifications, reduction orderings, and rewrite systems.[1] In Sect. 3 we introduce a general concept of refutational theorem proving, redundancy, and refutational completeness. In Sect. 4 we describe the hierarchic superposition calculus and then reduce the problem of theorem proving in hierarchic theories to the known case of flat first-order theories, using a notion of approximation between refutational proof systems.

Section 5 is dedicated to an ostensibly quite different topic. We sketch how our results can also be used to transform certain second-order formulae into equivalent first-order formulae by elimination of existentially quantified predicate symbols. The quantifier elimination algorithm presented by Gabbay and Ohlbach [9] is essentially an instance of our technique.

2 Preliminaries

2.1 Signatures, Clauses, and Specifications

A *signature* is a pair (Σ, Ω), where Σ is a set of *sorts* and Ω is a set of *operator symbols* over Σ. If X is a set of variables with sorts in Σ, then the set of terms over (Σ, Ω) and X is denoted by $\mathrm{T}_\Omega(X)$; T_Ω is short for $\mathrm{T}_\Omega(\emptyset)$.

An *Ω-equation* is a multiset $\{s, t\}$ (usually written $s \approx t$) where s and t are terms from $\mathrm{T}_\Omega(X)$ of the same sort. We encode every predicate as a function into a set with one distinguished element. Non-equational atoms are thus turned into equations $P(t_1, \ldots t_n) \approx true_P$; for simplicity, this is usually abbreviated by $P(t_1, \ldots, t_n)$.[2] A *(first-order) clause* is a pair of multisets of equations, written $\Gamma \to \Delta$. The multiset Γ is called the *antecedent*; the multiset Δ, the *succedent*. A clause $B_1, \ldots, B_m \to B'_1, \ldots, B'_n$ represents an implication $B_1 \wedge \cdots \wedge B_m \supset B'_1 \vee \cdots \vee B'_n$; the empty clause $\bot$, a contradiction. The set of all Ω-clauses is denoted by F_Ω. A *ground* expression (i.e., a term, equation, formula, etc.) is an expression containing no variables.

An *Ω-algebra A* consists of a Σ-sorted family of carrier sets $\{A_\xi\}_{\xi \in \Sigma}$ and of a function $A_f : A_{\xi_1} \times \cdots \times A_{\xi_n} \to A_{\xi_0}$ for every $f : \xi_1 \cdots \xi_n \to \xi_0$ in Ω. The *interpretation* t^A of a ground term t is defined recursively by $f(t_1, \ldots, t_n)^A = A_f(t_1^A, \ldots, t_n^A)$ for $n \geqq 0$. An algebra A is called *term-generated*, if every element of an A_ξ is the interpretation of some ground term of sort ξ. An algebra A is said to *satisfy* a ground equation $s \approx t$, if s and t have the same interpretation in A; it is said to *satisfy* a ground clause $\Gamma \to \Delta$ if at least one equation of Δ is satisfied by A or if at least one equation of Γ is not satisfied by A. We also say that a ground clause C is *true in* A, if A satisfies

[1] The reader is referred to the general surveys on algebraic specification and rewrite systems by Wirsing [16] and Dershowitz and Jouannaud [8], respectively.

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

C; and that C is *false in* A, otherwise. A term-generated algebra A is said to *satisfy* a non-ground clause $\Gamma \rightarrow \Delta$ if it satisfies all ground instances $\Gamma\sigma \rightarrow \Delta\sigma$; it is called a *model* of a set N of clauses, if it satisfies all clauses of N.[3] We say that N *implies* N', and write $N \models N'$, if every model of N is a model of N'; $N \models C$ is short for $N \models \{C\}$.

A *specification* is a triple $SP = (\Sigma, \Omega, \mathscr{C})$, where (Σ, Ω) is a signature and $\mathscr{C}$ is a class of term-generated Ω-algebras called *models* of the specification SP (denoted by Mod_{SP}). We assume that $\mathscr{C}$ is closed under isomorphisms. If $\mathscr{C}$ is the class of all Ω-models of a certain set of Ω-*axioms* Ax, then we write (Σ, Ω, Ax) instead of $(\Sigma, \Omega, \mathscr{C})$. We require that (Σ, Ω) is a *sensible* signature, i.e., that the term algebra T_Ω has no empty sorts.

2.2 Hierarchic Specifications

A *hierarchic specification* is a pair (SP, SP') of specifications, where $SP = (\Sigma, \Omega, \mathscr{C})$ is called the *base specification* and $SP' = (\Sigma', \Omega', Ax')$ the *body* of the hierarchic specification, and $\Sigma \subseteq \Sigma'$ and $\Omega \subseteq \Omega'$. The sorts in $\Sigma' \backslash \Sigma$, the operator symbols in $\Omega' \backslash \Omega$, and the axioms in Ax' are called *enrichment*. In this paper we consider only enrichments Ax' that consist of first-order clauses. By the definition of a specification, $\mathscr{C}$ contains only term-generated algebras. If necessary, we may extend Ω by an infinite number of constant symbols not occurring in Ax' and turn every non-term generated algebra into a term-generated one (by choosing as the interpretations of new constants those elements of A_ξ that do not otherwise correspond to any ground term).[4] A term or clause that consists only of base operators and variables of base sorts is called *base term* or *base clause*, respectively.

If A' is an Ω'-algebra, the restriction of A' to Ω, written $A'|_\Omega$, is the Ω-algebra that is obtained from A' by removing all carrier sets A'_ξ for $\xi \in \Sigma' \backslash \Sigma$ and all functions A'_f for $f \in \Omega' \backslash \Omega$. Note that $A'|_\Omega$ is not necessarily term-generated even if A' is term-generated.

The models of a hierarchic specification $HSP = (SP, SP')$ are those models of SP' that extend some model in $\mathscr{C}$ and neither collapse any of its sorts nor add new elements to them:

$$\mathrm{Mod}_{HSP} = \{A' \in \mathrm{Mod}_{SP'} : A'|_\Omega \in \mathscr{C}\}.$$

Let N and N' be two sets of Ω'-clauses. We say that N *implies* N' *relative to* $\mathscr{C}$ (and write $N \models_{\mathscr{C}} N'$), if every model of N whose restriction to Ω is in $\mathscr{C}$ is a model of N'. Note that $N \models_{\mathscr{C}} N'$ follows from $N \models N'$. If $N \models_{\mathscr{C}} \bot$, we call N *inconsistent relative to* $\mathscr{C}$; otherwise, we call it *consistent relative to* $\mathscr{C}$. If N is a set of base clauses, it is consistent relative to $\mathscr{C}$ if and only if some algebra of $\mathscr{C}$ is a model of N.

Refutational theorem proving for hierarchic theories amounts to checking whether one of the following three equivalent properties holds for a given set of Ω'-clauses N: (i) N is false in all models of HSP, (ii) $\mathrm{Mod}_{HSP_N} = \varnothing$, where HSP_N is the hierarchic specification $(SP, (\Sigma', \Omega', Ax' \cup N))$, or (iii) $Ax' \cup N$ is inconsistent relative to $\mathscr{C}$.

---

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

[4] Nevertheless it should be noted that our framework differs somewhat from the classical case of hierarchic equational specifications, where both the base specification and the enrichment are given by (conditional) equations. This will become particularly apparent in Sect. 4.5.

2.3 Rewrite Systems and Orderings

A binary relation $\Rightarrow$ on terms is called a *rewrite relation* if $s \Rightarrow t$ implies $u[s\sigma] \Rightarrow u[t\sigma]$, for all terms s, t and u, and substitutions σ. A transitive, well-founded rewrite relation is called a *reduction ordering*.

A set of equations E is called a *rewrite system* with respect to a reduction ordering $\succ$ if we have $s \succ t$ or $t \succ s$, for all equations $s \approx t$ in E. If all equations in E are ground, we speak of a ground rewrite system. Equations in E are also called (*rewrite*) *rules*. When we speak of "the rule $s \approx t$" we implicitly assume that $s \succ t$. By $\Rightarrow_E$ we denote the smallest rewrite relation for which $s \Rightarrow_E t$ whenever $s \approx t \in E$ and $s \succ t$. A term s is said to be in *normal form* (with respect to E) if there is no term t such that $s \Rightarrow_E t$. A term is also called *irreducible*, if it is in normal form, and *reducible*, otherwise.

Any ordering $\succ$ on a set S can be extended to an ordering $\succ_{\text{mul}}$ on finite multisets over S as follows: $M \succ_{\text{mul}} N$ if (i) $M \neq N$ and (ii) whenever $N(x) > M(x)$ then $M(y) > N(y)$, for some y such that $y \succ x$. If $\succ$ is a total [well-founded] ordering, so is $\succ_{\text{mul}}$. Given a set (or multiset) S and an ordering $\succ$ on S, we say that x is *maximal* relative to S if there is no $y \in S$ with $y \succ x$; and *strictly maximal* if there is no $y \in S$ with $y \succeq x$. If $\succ$ is an ordering on terms, then the corresponding multiset ordering $\succ_{\text{mul}}$ is an ordering on equations, which we denote by $\succ^{\text{e}}$.

We have defined clauses as pairs of multisets of equations. Alternatively, clauses may also be thought of as multisets of *occurrences* of equations. We identify an occurrence of an equation $s \approx t$ in the antecedent of a clause with the multiset (of multisets) $\{\{s, t\}\}$, and an occurrence in the succedent with the multiset $\{\{s\}, \{t\}\}$. We identify clauses with finite multisets of occurrences of equations. By $\succ^{\text{o}}$ we denote the twofold multiset ordering $(\succ_{\text{mul}})_{\text{mul}}$ of $\succ$, which is an ordering on occurrences of equations; by $\succ^{\text{c}}$ we denote the multiset ordering $\succ^{\text{o}}_{\text{mul}}$, which is an ordering on clauses. If $\succ$ is a total [well-founded] ordering, so are $\succ^{\text{e}}$, $\succ^{\text{o}}$, and $\succ^{\text{c}}$. A clause $C = \Gamma \rightarrow \Delta$, $s \approx t$ is called *reductive* for $s \approx t$, if $s \succ t$ and $s \approx t$ is a strictly maximal occurrence of an equation in C.

A reduction ordering is called *complete*, if it is total on ground terms. In the following we assume that $\succ$ is a complete reduction ordering on terms (though our results can also be extended to orderings that are contained in some complete ordering). Lexicographic path orderings (with a total precedence) are examples of such orderings.

3 Refutational Theorem Proving

The central part of a refutational theorem prover is a deductive inference system that computes new formulae from given ones. We may think of an inference system as a function $\mathcal{I}$ that maps a set N of formulae to a set of inferences

$$I = \frac{M}{C},$$

where $M \subseteq N$. The formulae in M are called *premises* of I. The formula C is called *conclusion* and is denoted by $\text{concl}(I)$. If J is a set of inferences, then $\text{concl}(J) = \{\text{concl}(I) \mid I \in J\}$. Typically, an inference system is *sound* with respect to a given semantical consequence relation $\models$, that is, $M \models C$, for all inferences I. We assume

the following properties about consequence relations: (i) $N_2 \subseteq N_1$ implies $N_1 \models N_2$, (ii) if $N_1 \models N_2$ and $N_1 \models N_3$, then $N_1 \models N_2 \cup N_3$, and (iii) if $N_1 \models N_2$ and $N_2 \models N_3$, then $N_1 \models N_3$.

Traditionally, refutational theorem proving has been formally described as a closure process that systematically adds the conclusions of inferences to a given set of formulae until a "closed" (or "saturated") set N^* is reached, where the conclusion of every inference in $\mathscr{I}(N^*)$ is already contained in N^*. An inference system is called refutationally complete, if a saturated set of formulae is unsatisfiable if and only if it contains a contradictory formula, say $A \wedge \neg A$. (In this paper we stick to a clausal framework, where the only contradictory formula is the empty clause $\perp$.) In practice, saturated sets tend to be very large, often infinite, even though most formulae may not actually be needed for deriving a contradiction. Thus, techniques are employed to discard redundant formulae and a weaker notion of saturation is needed. For that purpose, we introduce a general notion of redundancy that applies not only to formulae but also to inferences.

More formally, let $\mathscr{R}_{\text{For}}$ be a mapping from sets of formulae to sets of formulae and $\mathscr{R}_{\text{Inf}}$ be a mapping from sets of formulae to sets of inferences. The sets $\mathscr{R}_{\text{For}}(N)$ and $\mathscr{R}_{\text{Inf}}(N)$ are meant to specify formulae and inferences, respectively, deemed to be redundant in the context of a given set N. Under certain conditions, formulae in $\mathscr{R}_{\text{For}}(N)$ may be removed from N, while inferences in $\mathscr{R}_{\text{Inf}}(N)$ may be ignored. For instance, $\mathscr{R}_{\text{For}}(N)$ may consist of all tautologies and formulae subsumed by N. (We emphasize that $\mathscr{R}_{\text{For}}(N)$ need not be a subset of N and that $\mathscr{R}_{\text{Inf}}(N)$ will usually also contain inferences whose premises are not in N.)

The following conditions characterize a reasonable notion of redundancy for refutational theorem proving:

Definition 1. A pair $\mathscr{R} = (\mathscr{R}_{\text{Inf}}, \mathscr{R}_{\text{For}})$ is called a *redundancy criterion (with respect to an inference system $\mathscr{I}$ and a consequence relation $\models$)*, if the following conditions are satisfied for all sets of formulae N and N':

(i) $N \setminus \mathscr{R}_{\text{For}}(N) \models \mathscr{R}_{\text{For}}(N)$.
(ii) If $N \subseteq N'$, then $\mathscr{R}_{\text{For}}(N) \subseteq \mathscr{R}_{\text{For}}(N')$.
(iii) If $I \in \mathscr{I}(N')$ and $\text{concl}(I) \in N$, then $I \in \mathscr{R}_{\text{Inf}}(N)$.
(iv) If $N' \subseteq \mathscr{R}_{\text{For}}(N)$, then $\mathscr{R}_{\text{Inf}}(N) \subseteq \mathscr{R}_{\text{Inf}}(N \setminus N')$.

Inferences in $\mathscr{R}_{\text{Inf}}(N)$ and formulae in $\mathscr{R}_{\text{For}}(N)$ are said to be *redundant* with respect to N.

Condition (i) requires that redundant formulae be consequences of the non-redundant ones. Condition (ii) expresses a monotonicity property of redundancy. Condition (iii) states that an inference is redundant with respect to N if its conclusion is already present in N (regardless of whether or not the premises are in N). Finally, condition (iv) indicates that a redundant inference must remain redundant if redundant formulae are deleted.

Definition 2. A binary relation $\vdash$ on sets of formulae is called a *derivation relation (with respect to an inference system $\mathscr{I}$, a redundancy criterion $\mathscr{R}$, and a consequence relation $\models$)*, if it satisfies the following properties for all sets of formulae N and N':

(i) If $N \vdash N'$, then $N \models N'$.
(ii) If $N \vdash N'$, then $N \setminus N' \subseteq \mathscr{R}_{\text{For}}(N')$.
(iii) If $I \in \mathscr{I}(N)$, then $N \vdash N \cup \{\text{concl}(I)\}$.

A derivation relation extends an inference system in such a way that we can not

only compute inferences and add their conclusions but that we are also allowed to delete or to simplify formulae or to add lemmas. This is possible as long as all removed formulae are redundant and all new formulae are logical consequences of the old ones.

Note that $N \vdash N'$ implies $N' \models N$ by condition (i) of Definition 1.

Definition 3. A triple $(\mathscr{I}, \mathscr{R}, \vdash)$ consisting of an inference system $\mathscr{I}$, a redundancy criterion $\mathscr{R}$, and a derivation relation $\vdash$ is called a *theorem proving system.* A set of formulae N is called *saturated with respect to* a theorem proving system $(\mathscr{I}, \mathscr{R}, \vdash)$, if $\mathscr{I}(N) \subseteq \mathscr{R}_{\mathrm{Inf}}(N)$.

In other words, a set N is saturated if all inferences from it are redundant.

A sequence $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ is called an $(\mathscr{I}, \mathscr{R}, \vdash)$-*derivation* (or simply *derivation*, if the theorem proving system is clear from the context). The set $N_\infty = \bigcup_i \bigcap_{j \geq i} N_j$ of all *persisting* formulae is called the *limit* of the derivation.

Definition 4. A derivation $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ is called *fair*, if

$$\mathrm{concl}(\mathscr{I}(N_\infty) \setminus \mathscr{R}_{\mathrm{Inf}}(N_\infty)) \subseteq \bigcup_j N_j.$$

Lemma 5. *For every derivation* $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ *we have* $\mathscr{R}_{\mathrm{Inf}}(\bigcup_j N_j) \subseteq \mathscr{R}_{\mathrm{Inf}}(N_\infty)$. *Furthermore, if the derivation is fair, then the limit* N_∞ *is saturated.*

Proof. Suppose that a formula C is contained in $\bigcup_j N_j \setminus N_\infty$. Since C has been deleted at some point of the derivation, there must exist some k such that $C \in N_k \setminus N_{k+1} \subseteq \mathscr{R}_{\mathrm{For}}(N_{k+1}) \subseteq \mathscr{R}_{\mathrm{For}}(\bigcup_j N_j)$. In other words, $\bigcup_j N_j \setminus N_\infty \subseteq \mathscr{R}_{\mathrm{For}}(\bigcup_j N_j)$. By property (iv) this implies $\mathscr{R}_{\mathrm{Inf}}(\bigcup_j N_j) \subseteq \mathscr{R}_{\mathrm{Inf}}(\bigcup_j N_j \setminus (\bigcup_j N_j \setminus N_\infty)) = \mathscr{R}_{\mathrm{Inf}}(N_\infty)$.

Suppose that the derivation is fair and that there exists an inference $I \in \mathscr{I}(N_\infty) \setminus \mathscr{R}_{\mathrm{Inf}}(N_\infty)$. By fairness, its conclusion is contained in $\bigcup_j N_j$, hence $I \in \mathscr{R}_{\mathrm{Inf}}(\bigcup_j N_j)$. Since $\mathscr{R}_{\mathrm{Inf}}(\bigcup_j N_j) \subseteq \mathscr{R}_{\mathrm{Inf}}(N_\infty)$, this contradicts our assumption. □

Definition 6. Let $\mathscr{N}$ be a set of sets of formulae. A theorem proving system $(\mathscr{I}, \mathscr{R}, \vdash)$ is called *refutationally complete for* $\mathscr{N}$, if for every saturated set $N \in \mathscr{N}$ we have $N \models \bot$ if and only if $\bot \in N$.

Like the notion of saturation, most results on theorem proving systems do not depend on the particular choice of a derivation relation. In such cases, we will from now on omit the derivation relation and write $(\mathscr{I}, \mathscr{R})$ instead of $(\mathscr{I}, \mathscr{R}, \vdash)$.

4 Hierarchic Refutational Theorem Proving

4.1 The Hierarchic Superposition Calculus

The key idea of the hierarchic superposition calculus that we describe is to split all clauses into a base part and a non-base part such that all inferences work only on the non-base part. More formally, we say that a term or equation is *pure*, if it does not contain both a base operator and a non-base operator. A clause is called *abstracted*, if all its equations are pure. Every clause C can be transformed into an equivalent abstracted clause in the following way: whenever a subterm t whose top symbol is a base operator occurs immediately below a non-base operator symbol (or vice versa), then it is replaced by a new variable x (or "abstracted out") and the

equation $x \approx t$ is added to the antecedent of C. This transformation is repeated until all terms in the clause are pure; then the abstraction operation is applied to equations between a non-variable base term and a non-base term.

A substitution is called *simple*, if it maps every variable of a base sort to a base term. If σ is a simple substitution, $t\sigma$ is called a *simple instance* of the term t (analogously for equations and clauses). The set of all simple ground instances of a clause C is denoted by $\mathrm{sgi}(C)$, analogously $\mathrm{sgi}(N)$ is the set of all simple ground instances of all clauses in N. Note that a base term cannot contain variables of a non-base sort, hence a simple instance of a base term is also a base term.

In the rest of this paper we assume that every base ground term is smaller than every ground term that contains an operator symbol from $\Omega' \setminus \Omega$. This property can be easily enforced for lexicographic and other path orderings by choosing a suitable precedence on operator symbols.

Inference System 7. The following inference rules are only applied to abstracted clauses. Except for constraint refutation inferences, no premise of an inference is a base clause.

Equality resolution:
$$\frac{\Lambda, u \approx v \to \Pi}{\Lambda\sigma \to \Pi\sigma}$$

where (i) σ is simple and a most general unifier of u and v and (ii) there exists a simple substitution ψ such that $(u \approx v)\sigma\psi$ is a maximal occurrence of an equation in the ground clause $(\Lambda, u \approx v \to \Pi)\sigma\psi$.[5]

Ordered factoring:
$$\frac{\Gamma \to \Delta, B, B'}{\Gamma\sigma \to \Delta\sigma, B\sigma}$$

where (i) σ is simple and a most general unifier of B and B' and (ii) there exists a simple substitution ψ such that $B\sigma\psi$ is a maximal occurrence of an equation in the ground clause $(\Gamma \to \Delta, B, B')\sigma\psi$.

Superposition, left:
$$\frac{\Gamma \to \Delta, s \approx t \quad u[s'] \approx v, \Lambda \to \Pi}{u[t]\sigma \approx v\sigma, \Gamma\sigma, \Lambda\sigma \to \Delta\sigma, \Pi\sigma}$$

where (i) s' is not a variable, (ii) σ is simple and a most general unifier of s and s', and (iii) there exists a simple substitution ψ such that (a) the ground clause $(\Gamma \to \Delta, s \approx t)\sigma\psi$ is reductive for $(s \approx t)\sigma\psi$, (b) $u\sigma\psi \succ v\sigma\psi$, and (c) $(u \approx v)\sigma\psi$ is a maximal occurrence of an equation in the ground clause $(u \approx v, \Lambda \to \Pi)\sigma\psi$.

Superposition, right:
$$\frac{\Gamma \to \Delta, s \approx t \quad \Lambda \to u[s'] \approx v, \Pi}{\Gamma\sigma, \Lambda\sigma \to u[t]\sigma \approx v\sigma, \Delta\sigma, \Pi\sigma}$$

where (i) s' is not a variable, (ii) σ is simple and a most general unifier of s and s', and (iii) there exists a simple substitution ψ such that (a) the ground clause $(\Gamma \to \Delta, s \approx t)\sigma\psi$ is reductive for $(s \approx t)\sigma\psi$ and (b) the ground clause $(\Lambda \to u \approx v, \Pi)\sigma\psi$ is reductive for $(u \approx v)\sigma\psi$.

---

[5] We might replace condition (ii) by "$(u \approx v)\sigma$ is a maximal occurrence of an equation in $(\Lambda, u \approx v \to \Pi)\,\sigma$ and $u \approx v$ is not a base equation." This property looks easier but it rules out fewer inferences; for example, if f and a are non-base operators and x is a variable of base sort, then the equations $f(x) \approx f(x)$ and $f(a) \approx f(a)$ are incomparable, but $(f(x) \approx f(x))\psi$ is smaller than $(f(a) \approx f(a))\psi$ for every substitution ψ that maps x to a ground base term.

Equality factoring: $$\frac{\Gamma \to \Delta, s \approx t, s' \approx t'}{\Gamma\sigma, t\sigma \approx t'\sigma \to \Delta\sigma, s'\sigma \approx t'\sigma}$$

where (*i*) σ is simple and a most general unifier of s and s' and (ii) there exists a simple substitution ψ such that (a) $s\sigma\psi \succ t\sigma\psi$, (b) $s'\sigma\psi \succ t'\sigma\psi$, and (c) $(s \approx t)\sigma\psi$ is a maximal occurrence of an equation in the ground clause $(\Gamma \to \Delta, s \approx t, s' \approx t')\sigma\psi$.

Merging paramodulation: $$\frac{\Gamma \to \Delta, s \approx t \quad \Lambda \to u \approx v[s'], u' \approx v', \Pi}{\Gamma\sigma, \Lambda\sigma \to u\sigma \approx v[t]\sigma, u\sigma \approx v'\sigma, \Delta\sigma, \Pi\sigma}$$

where (i) s' is not a variable, (ii) σ is simple and the composition of $\tau\rho$ of a most general unifier τ of s and s' and a most general unifier ρ of $u\tau$ and $u'\tau$, and (iii) there exists a simple substitution ψ such that (a) the ground clause $(\Gamma \to \Delta, s \approx t)\sigma\psi$ is reductive for $(s \approx t)\sigma\psi$, (b) the ground clause $(\Lambda \to u \approx v, u' \approx v', \Pi)\sigma\psi$ is reductive for $(u \approx v)\sigma\psi$, (c) $u\tau\psi \succ v\tau\psi$, and (d) $v\sigma\psi \succ v'\sigma\psi$.

Constraint refutation: $$\frac{M}{\bot}$$

where the finite set M of base clauses is inconsistent relative to $\mathscr{C}$, i.e., no algebra in $\mathscr{C}$ is an Ω-model of $M \subseteq \mathrm{F}_\Omega$.

The inference system consisting of equality resolution, equality factoring, superposition, and constraint refutation is denoted by $\mathscr{HE}$, the inference system consisting of equality resolution, ordered factoring, superposition, merging paramodulation and constraint refutation by $\mathscr{HP}$. The letter $\mathscr{H}$ stands for either $\mathscr{HE}$ or $\mathscr{HP}$. In each case the following additional restrictions are imposed: (a) the premises of an inference rule must not share any variables and (b) if C and C' are the premises of a paramodulation inference then $C'\sigma\psi \succ^{\mathrm{c}} C\sigma\psi$.[6]

It is evident that the two inference systems $\mathscr{HE}$ and $\mathscr{HP}$ are sound with respect to $\models_{\mathscr{C}}$. Note that the conditions of the inference rules may be undecidable for a given ordering $\succ$. On the other hand, a derivation relation allows us to derive more formulae than those which are required by the inference system. Thus, in an implementation it is always possible to replace the conditions by weaker (decidable) restrictions.

Operationally, constraint refutation is implemented by a (specialized) theorem prover for the base specification, while the other rules form the deductive part of a general purpose hierarchic theorem prover.

Lemma 8. *If the premises of an $\mathscr{H}$-inference are abstracted clauses, then the conclusion is also abstracted.*[7]

Proof. Remember that base ground terms are always smaller than ground terms that contain a non-base operator symbol. Due to the ordering constraints of the inference rules, the simple substitution σ is always the most general unifier of terms

[6] As in (Bachmair and Ganzinger [4]), we might also extend these inference systems by selection functions, in which case base equations must not be selected.

[7] It is also possible and, perhaps, preferable in practice, to relax the notion of an abstracted clause by applying variable abstraction only to non-variable base terms occurring below a non-base operator. In this case, however, the inference rules may generate non-abstracted clauses, such that it becomes necessary to insert an abstraction step after each inference step.

that contain no base operator. Consequently, for every variable x of a base sort, $x\sigma$ is a base term but does not contain a base operator, i.e., it is a variable; and for every variable y of a non-base sort, $y\sigma$ does not contain a base operator. This implies that the application of σ to a pure equation yields again a pure equation. □

4.2 Approximations

We will prove that the hierarchic superposition calculus is refutationally complete by lifting Bachmair and Ganzinger's refutational completeness result for the flat case [4]. To this end we introduce a concept of approximation between theorem proving systems:

Definition 9. Suppose that $\models_1$ and $\models_2$ are two semantical consequence relations, and that $(\mathscr{I}_i, \mathscr{R}_i)$ is a theorem proving system with respect to $\models_i$ for $i \in \{1,2\}$. The symbol $\perp_i$ denotes a contradiction for $\models_i$. Moreover, suppose that the function α maps the set $\mathscr{N}_1$ of sets of formulae (with respect to $\models_1$) to the set $\mathscr{N}_2$ of sets of formulae (with respect to $\models_2$). We say that $(\mathscr{I}_1, \mathscr{R}_1, \models_1, \mathscr{N}_1)$ *approximates* $(\mathscr{I}_2, \mathscr{R}_2, \models_2, \mathscr{N}_2)$ *via* α, if the following three conditions are satisfied for every $N \in \mathscr{N}_1$: (i) if N is saturated with respect to $(\mathscr{I}_1, \mathscr{R}_1)$, then $\alpha(N)$ is saturated with respect to $(\mathscr{I}_2, \mathscr{R}_2)$, (ii) if $N \models_1 \perp_1$, then $\alpha(N) \models_2 \perp_2$, and (iii) if $\perp_2 \in \alpha(N)$ and N is saturated, then $\perp_1 \in N$.

Lemma 10. *If* $(\mathscr{I}_1, \mathscr{R}_1, \models_1, \mathscr{N}_1)$ *approximates* $(\mathscr{I}_2, \mathscr{R}_2, \models_2, \mathscr{N}_2)$ *via* α *and if* $(\mathscr{I}_2, \mathscr{R}_2)$ *is refutationally complete for* $\mathscr{N}_2$, *then* $(\mathscr{I}_1, \mathscr{R}_1)$ *is refutationally complete for* $\mathscr{N}_1$.

4.3 The Superposition Calculus

For the purpose of approximation we only need the ground version of Bachmair and Ganzinger's superposition calculus, which is described below.

Inference System 11.

Equality resolution: $$\frac{\Lambda, u \approx u \to \Pi}{\Lambda \to \Pi}$$

where $u \approx u$ is a maximal occurrence of an equation in the premise.

Ordered factoring: $$\frac{\Gamma \to \Delta, B, B}{\Gamma \to \Delta, B}$$

where B is a maximal occurrence of an equation in the premise.

Superposition, left: $$\frac{\Gamma \to \Delta, s \approx t \quad u[s] \approx v, \Lambda \to \Pi}{u[t] \approx v, \Gamma, \Lambda \to \Delta, \Pi}$$

where (i) the first premise is reductive for $s \approx t$, (ii) $u \succ v$ and (iii) $u \approx v$ is a maximal occurrence of an equation in the second premise.

Superposition, right: $$\frac{\Gamma \to \Delta, s \approx t \quad \Lambda \to u[s] \approx v, \Pi}{\Gamma, \Lambda \to u[t] \approx v, \Delta, \Pi}$$

where (i) the first premise is reductive for $s \approx t$, (ii) the second premise is reductive for $u \approx v$, and (iii) $u \approx v \succ^c s \approx t$.

Equality factoring: $$\frac{\Gamma \to \Delta, s \approx t, s \approx t'}{\Gamma, t \approx t' \to \Delta, s \approx t'}$$

where (i) $s \succ t$, (ii) $s \succ t'$, and (iii) $s \approx t$ is a maximal occurrence of an equation in the premise.

Merging paramodulation: $$\frac{\Gamma \to \Delta, s \approx t \quad \Lambda \to u \approx v[s], u \approx v', \Pi}{\Gamma, \Lambda \to u \approx v[t], u \approx v', \Delta, \Pi}$$

where (i) the first premise is reductive for $s \approx t$, (ii) the second premise is reductive for $u \approx v$, and (iii) $v \succ v'$.

The inference system consisting of equality resolution, equality factoring, and superposition is denoted by $\mathscr{E}$, the inference system consisting of equality resolution, ordered factoring, superposition, and merging paramodulation by $\mathscr{P}$. The letter $\mathscr{F}$ stands for either $\mathscr{E}$ or $\mathscr{P}$.

An essential property of the above inference rules is that the conclusion of a ground inference is always simpler (with respect to the ordering $\succ^c$) than the maximal premise (which is always the second premise in the case of the paramodulation inference).

Definition 12. Let N be a set of ground clauses. We define $\mathscr{R}^{\mathscr{F}}_{\text{For}}(N)$ to be the set of all clauses C such that there exist clauses $C_1, \ldots, C_n \in N$ that are smaller than C with respect to $\succ^c$ and $C_1, \ldots, C_n \models C$. We define $\mathscr{R}^{\mathscr{F}}_{\text{Inf}}(N)$ to be the set of all inferences I such that either a premise of I is in $\mathscr{R}^{\mathscr{F}}_{\text{For}}(N)$ or else there exist clauses $C_1, \ldots, C_n \in N$ that are smaller with respect to $\succ^c$ than the maximal premise of I and $C_1, \ldots, C_n \models \text{concl}(I)$.

The following results are due to Bachmair and Ganzinger [4].[8]

Theorem 13. *The inference systems* $\mathscr{E}$ and $\mathscr{P}$, *and* $\mathscr{R}^{\mathscr{F}} = (\mathscr{R}^{\mathscr{F}}_{\text{Inf}}, \mathscr{R}^{\mathscr{F}}_{\text{For}})$ *satisfy the following properties:*

(i) $\mathscr{R}^{\mathscr{F}}$ *is a redundancy criterion with respect to* $\models$.
(ii) $(\mathscr{E}, \mathscr{R}^{\mathscr{E}})$ *and* $(\mathscr{P}, \mathscr{R}^{\mathscr{P}})$ *are refutationally complete theorem proving systems.*
(iii) $N \subseteq N'$ *implies* $\mathscr{R}^{\mathscr{F}}_{\text{Inf}}(N) \subseteq \mathscr{R}^{\mathscr{F}}_{\text{Inf}}(N')$.
(iv) $N' \subseteq \mathscr{R}^{\mathscr{F}}_{\text{For}}(N)$ *implies* $\mathscr{R}^{\mathscr{F}}_{\text{For}}(N) \subseteq \mathscr{R}^{\mathscr{F}}_{\text{For}}(N \setminus N')$.[9]

4.4 The Hierarchic Redundancy Criterion

Let I be an inference in $\mathscr{H}$ with premises $C_1, \ldots, C_n$ and conclusion C, where the clauses $C_1, \ldots, C_n$ have no variables in common. Let I' be an inference in $\mathscr{F}$ with premises $C'_1, \ldots, C'_n$ and conclusion C'. If σ is a simple substitution such that $C' = C\sigma$ and $C'_i = C_i\sigma$ for all i, then I' is called a *simple ground instance* of I. The set of all simple ground instances of an inference I is denoted by $\text{sgi}(I)$.

[8] Similar results for a slightly different ordering $\succ^c$ have already appeared in [2]. A refutational completeness proof for $\mathscr{E}$ was first given by Nieuwenhuis [13].
[9] Note that redundancy in our sense is called compositeness in [4].

Definition 14. Let N be a set of abstracted clauses. We define $\mathscr{R}^{\mathscr{H}}_{\mathrm{Inf}}(N)$ to be the set of all inferences I such that either I is not a constraint refutation inference and $\mathrm{sgi}(I) \subseteq \mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(\mathrm{sgi}(N))$, or else I is a constraint refutation inference and $\bot \in N$. We define $\mathscr{R}^{\mathscr{H}}_{\mathrm{For}}(N)$ to be the set of all abstracted clauses C such that $\mathrm{sgi}(C) \subseteq \mathscr{R}^{\mathscr{F}}_{\mathrm{For}}(\mathrm{sgi}(N))$.

Lemma 15. *If* $\mathrm{sgi}(N) \models \mathrm{sgi}(C)$, *then* $N \models_{\mathscr{C}} C$.

Proof. Suppose that $\mathrm{sgi}(N) \models \mathrm{sgi}(C)$ and let A' be an Ω'-model of N whose restriction to Ω is contained in $\mathscr{C}$. Since A' does not add new elements to the sorts of $A = A'|_{\Omega}$ and A is a term-generated Ω-algebra, we know that for every Ω'-term t' of a base sort there exists a base term t, such that t and t' have the same interpretation in A'. Consequently, for every ground substitution σ' there exists an equivalent simple ground substitution σ; since $C\sigma$ is valid in A', $C\sigma'$ is also valid. □

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

Lemma 16. $\mathscr{R}^{\mathscr{H}} = (\mathscr{R}^{\mathscr{H}}_{\mathrm{Inf}}, \mathscr{R}^{\mathscr{H}}_{\mathrm{For}})$ *is a redundancy criterion with respect to* $\models_{\mathscr{C}}$.

Proof. We have to check the four conditions of Definition 1. The proof of property (ii) is rather trivial. To check property (i) let D be an arbitrary clause from $\mathrm{sgi}(\mathscr{R}^{\mathscr{H}}_{\mathrm{For}}(N))$. Since $D \in \mathscr{R}^{\mathscr{F}}_{\mathrm{For}}(\mathrm{sgi}(N))$, we have $\mathrm{sgi}(N) \setminus \mathscr{R}^{\mathscr{F}}_{\mathrm{For}}(\mathrm{sgi}(N)) \models D$ and consequently $\mathrm{sgi}(N \setminus \mathscr{R}^{\mathscr{H}}_{\mathrm{For}}(N)) \models D$. We can conclude that $N \setminus \mathscr{R}^{\mathscr{H}}_{\mathrm{For}}(N) \models_{\mathscr{C}} \mathscr{R}^{\mathscr{H}}_{\mathrm{For}}(N)$.

Condition (iii) is obviously satisfied for all constraint refutation inferences. If I is not a constraint refutation inference and $\mathrm{concl}(I) \in N$, we know that $\mathrm{concl}(\mathrm{sgi}(I)) \subseteq \mathrm{sgi}(N)$. As $\mathrm{sgi}(I) \subseteq \mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(\mathrm{sgi}(N))$, the inference I is contained in $\mathscr{R}^{\mathscr{H}}_{\mathrm{Inf}}(N)$. This proves condition (iii).

We come now to the proof of condition (iv). Note that $N' \subseteq \mathscr{R}^{\mathscr{H}}_{\mathrm{For}}(N)$ implies $\mathrm{sgi}(N') \subseteq \mathscr{R}^{\mathscr{F}}_{\mathrm{For}}(\mathrm{sgi}(N))$. If $I \in \mathscr{R}^{\mathscr{H}}_{\mathrm{Inf}}(N)$ is a constraint refutation inference, then $\bot \in N$; since $\bot \notin \mathscr{R}^{\mathscr{H}}_{\mathrm{For}}(N)$, I is contained in $\mathscr{R}^{\mathscr{H}}_{\mathrm{Inf}}(N \setminus N')$. Otherwise $\mathrm{sgi}(I) \subseteq \mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(\mathrm{sgi}(N)) \subseteq \mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(\mathrm{sgi}(N) \setminus \mathrm{sgi}(N')) \subseteq \mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(\mathrm{sgi}(N \setminus N'))$ and thus $\mathscr{R}^{\mathscr{H}}_{\mathrm{Inf}}(N) \subseteq \mathscr{R}^{\mathscr{H}}_{\mathrm{Inf}}(N \setminus N')$. □

Let $A \in \mathscr{C}$ be a term-generated Ω-algebra. For every Ω-ground term t let $m(t)$ be the smallest ground term of the congruence class of t in A. We define a rewrite system E'_A by $\mathrm{E}'_A = \{t \approx m(t) \mid t \in \mathrm{T}_{\Omega},\ t \neq m(t)\}$. Obviously, E'_A is terminating, right-reduced, and confluent. Now let E_A be the set of all rules $l \approx r$ in E'_A such that l is not reducible by $\mathrm{E}'_A \setminus \{l \approx r\}$. It is fairly easy to prove that E'_A and E_A define the same set of normal forms, and from this we can conclude that E_A and E'_A induce the same equality relation on T_{Ω}. We identify E_A with the set of clauses $\{\rightarrow t \approx t' \mid t \approx t' \in \mathrm{E}_A\}$. Let D_A be the set of all clauses $t \approx t' \rightarrow$, such that t and t' are distinct Ω-ground terms in normal form with respect to E_A.

Lemma 17. *Let* $A \in \mathscr{C}$ *be a term-generated* Ω*-algebra and let* C *be a ground base clause. Then* C *is true in* A *if and only if there exist clauses* $C_1, \ldots, C_n$ *in* $\mathrm{E}_A \cup \mathrm{D}_A$ *such that* $C_1, \ldots, C_n \models C$ *and* $C \succeq^{\mathrm{c}} C_i$ *for* $1 \leq i \leq n$.

Let N be a set of abstracted clauses and $A \in \mathscr{C}$ be a term-generated Ω-algebra, then N_A denotes the set $\mathrm{E}_A \cup \mathrm{D}_A \cup \{C\sigma \mid \sigma$ simple, reduced with respect to E_A, $C \in N$, $C\sigma$ ground$\}$.

Lemma 18. *If* N *is a set of abstracted clauses, then* $\mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(\mathrm{sgi}(N)) \subseteq \mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(N_A)$.

Proof. By part (iii) of Theorem 13 we have obviously $\mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(\mathrm{sgi}(N)) \subseteq \mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(\mathrm{E}_A \cup \mathrm{D}_A \cup \mathrm{sgi}(N))$. Let C be a clause in $\mathrm{E}_A \cup \mathrm{D}_A \cup \mathrm{sgi}(N)$ and not in N_A. As $C = C'\sigma$ for some $C' \in N$, it follows from $C'\rho$ and $\mathrm{E}_A \cup \mathrm{D}_A$, where ρ is the substitution that maps every variable x to the E_A-normal form of $x\sigma$. Since C follows from smaller clauses in $\mathrm{E}_A \cup \mathrm{D}_A \cup \mathrm{sgi}(N)$, it is in $\mathscr{R}^{\mathscr{F}}_{\mathrm{For}}(\mathrm{E}_A \cup \mathrm{D}_A \cup \mathrm{sgi}(N))$. Hence $\mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(\mathrm{E}_A \cup \mathrm{D}_A \cup \mathrm{sgi}(N)) \subseteq \mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(N_A)$. □

Theorem 19. *Let $A \in \mathscr{C}$ be a term-generated Ω-algebra and let N be a set of abstracted clauses. If A satisfies $\mathrm{sgi}(N) \cap \mathrm{F}_\Omega$ and N is saturated with respect to $(\mathscr{H}, \mathscr{R}^{\mathscr{H}})$, then N_A is saturated with respect to $(\mathscr{F}, \mathscr{R}^{\mathscr{F}})$.*

Proof. We have to show that every $\mathscr{F}$-inference from clauses of N_A is redundant with respect to N_A, i.e., that it is contained in $\mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(N_A)$. We demonstrate this in detail for the equality resolution and the left superposition rule. The analysis of the other rules is similar. Note that by Lemma 17 every base clause that is true in A and is not contained in $\mathrm{E}_A \cup \mathrm{D}_A$ follows from smaller clauses in $\mathrm{E}_A \cup \mathrm{D}_A$, thus it is in $\mathscr{R}^{\mathscr{F}}_{\mathrm{For}}(N_A)$; every inference involving such a clause is in $\mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(N_A)$.

The equality resolution rule is obviously not applicable to clauses from $\mathrm{E}_A \cup \mathrm{D}_A$. Suppose that I is an equality resolution inference with a premise $C\sigma$, where $C \in N$ and σ is a simple substitution and reduced with respect to E_A. If C is a base clause, then I is in $\mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(N_A)$. If C is not a base clause, then I is a simple ground instance of an $\mathscr{H}$-inference I' from C. Since I' is in $\mathscr{R}^{\mathscr{H}}_{\mathrm{Inf}}(N)$, I is in $\mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(\mathrm{sgi}(N))$, again this implies $I \in \mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(N_A)$.

Obviously a clause from D_A cannot be the first premise of a left superposition inference. Suppose that the first premise is a clause from E_A. The second premise cannot be a non-base clause, since all ground terms in the substitution part of a clause $C\sigma$ are reduced; as it is a base clause, the inference is redundant. Now suppose that I is a left superposition inference with a first premise $C\sigma$, where $C \in N$ and σ is a simple substitution and reduced with respect to E_A. If C is a base clause, then I is in $\mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(N_A)$. Otherwise, we can conclude that the second premise can be written as $C'\sigma$, where $C' \in N$ is not a base clause (without loss of generality, C and C' do not have common variables). We have to distinguish between two cases: If the overlap takes place below a variable occurrence, the conclusion of the inference follows from $C\sigma$ and some instance $C'\rho$, which are both smaller than $C'\sigma$. Otherwise, I is a simple ground instance of an $\mathscr{H}$-inference I' from C. In both cases, I is contained in $\mathscr{R}^{\mathscr{F}}_{\mathrm{Inf}}(N_A)$. □

We emphasize that this reduction of the hierarchic to the flat case would be impossible without the concept of redundancy.

4.5 Sufficient Completeness

So far we have considered only simple ground instances of the clauses in N. To compensate for this fact, we need a property similar to sufficient completeness that makes it possible to prove that every model of $\mathrm{sgi}(N)$ is actually a model of all ground instances of clauses in N.

Bergstra et al. [5] have shown that hierarchic specifications that are not sufficiently complete have a significantly greater expressive power than usual flat specifications. For example, an equational monomorphic hierarchic enrichment

specification may specify any computably hyperarithmetical algebra. Theorem proving in hierarchic specifications includes the problem of synthesizing programs from specifications. Therefore, it is not surprising that some property similar to sufficient completeness is necessary for a refutationally complete theorem proving method.

Definition 20. A set N of clauses is called *sufficiently complete with respect to simple instances*, if for every model A' of $\mathrm{sgi}(N)$ and every ground non-base term t' of a base sort ξ there exists a ground base term t of sort ξ such that $t' \approx t$ is true in A'.

Sufficient completeness with respect to simple instances differs from sufficient completeness in the usual sense (for example as described in [16]) in several ways. First, the term t may depend on the model A'. For example, if a and b are base terms and f is the only non-base operator, then the clause $\rightarrow f(x) = a, f(x) = b$ is sufficiently complete with respect to simple instances, but not sufficiently complete in the usual sense. On the other hand, t' must be equivalent to some t in all models of $\mathrm{sgi}(N)$ and not just in models of N.[10] Third, the usual notion of sufficient completeness applies to equational specifications; and to derive the equality of t and t', not only the enrichment, but also the (conditional) equations of the base specification may be used. In our framework, this is impossible since there are in general no base axioms: the base specification is given by an arbitrary class of models. Of course, in the case of an equational base specification we can compensate for this by duplicating the base axioms in the enrichment. But even then, due to the second distinction sufficient completeness does not imply sufficient completeness with respect to simple instances.

The following lemma demonstrates that sufficient completeness with respect to simple instances is preserved by theorem proving derivations.

Lemma 21. *Let $N_0 \vdash N_1 \vdash N_2 \vdash \ldots$ be an $(\mathscr{H}, \mathscr{R}^{\mathscr{H}})$-derivation. If N_i is sufficiently complete with respect to simple instances for some i, then so are N_∞ and N_k for every $k > i$.*

Proof. If N_i is sufficiently complete with respect to simple instances, then so is $\bigcup_j N_j$. We obtain N_∞ from $\bigcup_j N_j$ by removing a subset of $\mathscr{R}^{\mathscr{H}}_{\mathrm{For}}(\bigcup_j N_j)$. Similarly to part (i) of the proof of Lemma 16 we can now show that $\mathrm{sgi}(N_\infty)$ implies $\mathrm{sgi}(\mathscr{R}^{\mathscr{H}}_{\mathrm{For}}(\bigcup_j N_j))$ and thus $\mathrm{sgi}(N_\infty) \models \mathrm{sgi}(\bigcup_j N_j)$.

The sufficient completeness of N_k for $k > i$ is proved analogously. □

Though sufficient completeness with respect to simple instances is undecidable in general, there are important classes of specifications for which it can be trivially verified: for example, if every new function f into a base sort is coded as a predicate P_f, so that there exist no non-base operator symbols whose codomain is a base sort, a hierarchic specification is obviously sufficiently complete with respect to simple instances. Using a clause $P_f(x_1, \ldots, x_n, y)$, $P_f(x_1, \ldots, x_n, z) \rightarrow y \approx z$, we can ensure that P_f represents indeed a partial function; the totality of P_f, however, cannot be formulated.

In model theoretic investigations, such as (Kreisel and Krivine [12]), the extension of an algebra is defined in a different way, such that the addition of "junk" is allowed. Avenhaus and Becker [1] model this situation using subsorts and

[10] The set $\mathrm{sgi}(N)$ may have models that are not models of N, see Example 26 below.

generalize conditional rewriting to such a framework. We can simulate this behaviour as follows: For every base sort ξ we introduce a new non-base sort ξ' together with an injection function $\iota_\xi: \xi \to \xi'$ and a clause $\iota_\xi(x) \approx \iota_\xi(y) \to x \approx y$. For every base operator $f: \xi_1 \cdots \xi_n \to \xi_0$ we introduce a new non-base operator $f': \xi_1' \cdots \xi_n' \to \xi_0'$ together with an inheritance axiom $z \approx f(x_1, \ldots, x_n) \to f'(\iota_{\xi_1}(x_1), \ldots, \iota_{\xi_n}(x_n)) \approx \iota_{\xi_0}(z)$. In all old non-base operator declarations, we replace every occurrence of a base sort ξ by ξ'. Finally we replace every base operator symbol f in the enrichment by its non-base counterpart f' and every base variable x by $\iota_\xi(x)$. The specification that we obtain in this way is sufficiently complete with respect to simple instances and consists of abstracted clauses.

If for every base sort ξ there exist only finitely many base ground terms t_i $(1 \leqq i \leqq k)$, then sufficient completeness with respect to simple instances can be obtained by adding a clause $y_1 \approx t_1, \ldots, y_k \approx t_k \to f(x_1, \ldots, x_n) \approx y_1, \ldots, f(x_1, \ldots, x_n) \approx y_k$ for every non-base operator symbol f with codomain ξ. (Note that it is not sufficient to add the clause $\to x_\xi \approx t_1, \ldots, x_\xi \approx t_k$, as demonstrated by Example 26.)

Finally, it is sometimes possible to transform a hierarchic specification into a sufficiently complete one by extending the base signature with new constants and function symbols, which are interpreted existentially. Terms of a base sort that are not provably equal to some base term may then be declared to be equal to some term consisting of these new symbols. Unfortunately, this method fails for terms of a base sort containing variables of a non-base sort, and even worse, not every base theorem prover can be extended to cope with existentially quantified terms. If the base theory is first-order, however, adding new constants is unproblematic.

4.6 Refutational Completeness

Definition 22. A specification $(\Sigma, \Omega, \mathscr{C})$ is called *compact*, if every infinite set of Ω-clauses that is inconsistent relative to $\mathscr{C}$ has a finite subset that is also inconsistent relative to $\mathscr{C}$.

Theorem 23. *Suppose that $(\Sigma, \Omega, \mathscr{C})$ is a compact base specification. Let $\mathscr{SC}$ be the set of all sets of clauses that are sufficiently complete with respect to simple instances. For $N \in \mathscr{SC}$ let $\alpha(N)$ be defined in the following way: If N is saturated and if $\mathscr{C}$ contains some term-generated algebra A that satisfies $\mathrm{sgi}(N) \cap \mathrm{F}_\Omega$, we arbitrarily pick some such A and set $\alpha(N) = N_A$, otherwise $\alpha(N) = \{\bot\}$. Then $(\mathscr{H}, \mathscr{R}^{\mathscr{H}}, \models_{\mathscr{C}}, \mathscr{SC})$ approximates $(\mathscr{F}, \mathscr{R}^{\mathscr{F}}, \models, 2^{\mathrm{F}_{\Omega'}})$ via α, where $2^{\mathrm{F}_{\Omega'}}$ is the powerset of $\mathrm{F}_{\Omega'}$.*

Proof. We have to prove the following three properties: (i) if N is saturated with respect to $(\mathscr{H}, \mathscr{R}^{\mathscr{H}})$, then $\alpha(N)$ is saturated with respect to $(\mathscr{F}, \mathscr{R}^{\mathscr{F}})$, (ii) if $N \models_{\mathscr{C}} \bot$, then $\alpha(N) \models \bot$, and (iii) if $\bot \in \alpha(N)$ and N is saturated, then $\bot \in N$.

Let N be saturated with respect to $(\mathscr{H}, \mathscr{R}^{\mathscr{H}})$. If there exists some term-generated algebra $A \in \mathscr{C}$ that satisfies $\mathrm{sgi}(N) \cap \mathrm{F}_\Omega$, then the saturation of $\alpha(N) = N_A$ follows from Theorem 19; otherwise $\alpha(N) = \{\bot\}$ is obviously saturated.

To prove property (ii), assume that $\alpha(N) \not\models \bot$. This implies that $\alpha(N) = N_A$ for some Ω-algebra $A \in \mathscr{C}$ satisfying $\mathrm{sgi}(N) \cap \mathrm{F}_\Omega$; moreover N is saturated. Let A' be a model of $\alpha(N)$. Since $N_A \models \mathrm{sgi}(N)$, A' is also a model of $\mathrm{sgi}(N)$. Using the fact that N is sufficiently complete with respect to simple instances, we know that for every ground term t' of a base sort there exists a base term t such that $t' \approx t$ is true in A'. Consequently, for every ground instance of a clause in N there exists an equivalent

simple ground instance, thus A' is also a model of all ground instances of clauses in N. To see that the restriction of A' to Ω is isomorphic to A and thus in $\mathscr{C}$, note that A' satisfies $E_A \cup D_A$, preventing confusion, and that N is in $\mathscr{SC}$, preventing junk. Since A' satisfies N and $A'|_\Omega \in \mathscr{C}$, we have $N \not\models_{\mathscr{C}} \bot$.

For (iii) suppose that $\bot \in \alpha(N)$ and N is saturated. Consequently, there is no model $A \in \mathscr{C}$ that satisfies $\mathrm{sgi}(N) \cap F_\Omega = \mathrm{sgi}(N \cap F_\Omega)$. Since $N \cap F_\Omega$ is inconsistent relative to $\mathscr{C}$, there exists a finite subset of $N \cap F_\Omega$ with the same property, so that there is a constraint refutation inference that derives $\bot$ from N. This inference is redundant if and only if $\bot \in N$. □

By Lemma 10 this implies immediately:

Theorem 24. *If the base specification is compact, then the theorem proving systems* $(\mathscr{HE}, \mathscr{R}^{\mathscr{H}})$ *and* $(\mathscr{HP}, \mathscr{R}^{\mathscr{H}})$ *are refutationally complete for all sets of clauses that are sufficiently complete with respect to simple instances.*

We conclude the section with two examples.

Example 25. Consider the base specification NAT with the signature $(\Sigma, \Omega) = (\{\mathsf{nat}\}, \{0 : \to \mathsf{nat}, \mathsf{succ} : \mathsf{nat} \to \mathsf{nat}, + : \mathsf{nat}\ \mathsf{nat} \to \mathsf{nat}\})$ and the natural numbers $\mathbb{N}$ as the only model (up to isomorphism). Let the enrichment contain the new sort stack, new operator symbols $\mathsf{empty} : \to \mathsf{stack}$, $\mathsf{push} : \mathsf{stack}\ \mathsf{nat} \to \mathsf{stack}$, $\mathsf{pop} : \mathsf{stack} \to \mathsf{stack}$, and $\mathsf{top} : \mathsf{stack} \to \mathsf{nat}$, and the following set $STACK$ of abstracted clauses:

$$\to \mathsf{pop}(\mathsf{push}(s, x)) \approx s \tag{1}$$

$$\to \mathsf{top}(\mathsf{push}(s, x)) \approx x \tag{2}$$

$$\to \mathsf{pop}(\mathsf{empty}) \approx \mathsf{empty} \tag{3}$$

$$x \approx 0 \to \mathsf{top}(\mathsf{empty}) \approx x \tag{4}$$

$$\to \mathsf{push}(\mathsf{pop}(s), \mathsf{top}(s)) \approx s, s \approx \mathsf{empty} \tag{5}$$

$$\mathsf{push}(s, x) \approx \mathsf{empty} \to . \tag{6}$$

Note that clauses (3) and (4) are needed to make $STACK$ sufficiently complete with respect to simple instances. In contrast to purely equational logic, this does not induce inconsistencies, since applications of pop and top can be "guarded" (as for example in clause (5)).

We use a lexicographic path ordering with precedence $\mathsf{pop} \succ \mathsf{top} \succ \mathsf{push} \succ \mathsf{empty} \succ + \succ \mathsf{succ} \succ 0$. All inferences from clauses in $STACK$ are redundant, i.e., $STACK$ is saturated with respect to $\mathscr{H}$. Since $STACK$ does not contain the empty clause, this means that $STACK$ is consistent relative to $\{\mathbb{N}\}$.[11]

Let us test whether the following goal contradicts the hierarchic specification:

$$\mathsf{pop}(\mathsf{push}(\mathsf{push}(\mathsf{empty}, \mathsf{succ}(x)), x)) \approx \mathsf{push}(s, x + y) \to . \tag{7}$$

We transform (7) into the abstracted clause

$$w \approx \mathsf{succ}(x), v \approx x + y, \mathsf{pop}(\mathsf{push}(\mathsf{push}(\mathsf{empty}, w), x)) \approx \mathsf{push}(s, v) \to \tag{8}$$

[11] In fact, as $STACK$ does not contain any base clause, it is consistent relative to *every* (Σ, Ω)-base specification whose class of models is not empty. In other words, for every non-empty class of Ω-algebras $\mathscr{C}$ there exists a model A' of $STACK$ such that $A'|_\Omega \in \mathscr{C}$.

and add it to *STACK*. Now left superposition of (1) and (8) yields

$$w \approx \mathsf{succ}(x), v \approx x + y, \mathsf{push}(\mathsf{empty}, w) \approx \mathsf{push}(s, v) \rightarrow . \tag{9}$$

Applying the equality resolution rule to (9), we obtain

$$w \approx \mathsf{succ}(x), w \approx x + y \rightarrow . \tag{10}$$

Since the base clause (10) is false in $\{\mathbb{N}\}$, the constraint refutation rule allows us to derive $\perp$.

Next we will refute the clause

$$\rightarrow \mathsf{push}(s, \mathsf{succ}(0)) \approx \mathsf{push}(s, 0), \tag{11}$$

which is equivalent to the abstracted clause

$$x \approx \mathsf{succ}(0),\ y \approx 0 \rightarrow \mathsf{push}(s, x) \approx \mathsf{push}(s, y). \tag{12}$$

Right superposition of (12) on (2) yields

$$x \approx \mathsf{succ}(0),\ y \approx 0 \rightarrow \mathsf{top}(\mathsf{push}(s, y)) \approx x. \tag{13}$$

Now we apply the right superposition rule to (2) and (13) and obtain

$$x \approx \mathsf{succ}(0),\ y \approx 0 \rightarrow y \approx x, \tag{14}$$

a base clause that is again false in $\{\mathbb{N}\}$. We then use the constraint refutation rule to derive $\perp$.

Note that sufficient completeness in the usual sense is not strong enough for Theorem 24 to hold, as is demonstrated by the following example.

Example 26. Suppose that $(\{\mathsf{unit}\}, \{\mathsf{u} : \rightarrow \mathsf{unit}\}, \{\mathsf{u}\})$ is the base specification. Assume that the enrichment contains the additional operator symbol $\mathsf{foo} : \rightarrow \mathsf{unit}$ and that the set N consists of the abstracted clauses

$$\rightarrow x \approx \mathsf{u} \tag{1}$$

$$x \approx \mathsf{u}, x \approx \mathsf{foo} \rightarrow . \tag{2}$$

Due to clause (1), N is obviously sufficiently complete, but not sufficiently complete with respect to simple instances. The only model of the base specification (up to isomorphism) satisfies all simple ground instances of clause (1) (the only base clause of N). As there are no inferences from clauses in N, the set N is saturated. However, N does not contain the empty clause, even though it has no model at all.

4.7 Deletion and Simplification

As mentioned earlier, powerful deletion and simplification techniques are usually indispensable for the termination of saturation. The simplification methods (e.g., elimination of redundant atoms, case analysis, contextual reductive rewriting) that have been described by Bachmair and Ganzinger [4] all fit into the framework of Definition 2 and can (with slight variations) be carried over to the hierarchic case. Some care, however, is necessary, since the hierarchic redundancy criterion is more restrictive than its flat counterpart. As a consequence, a simplification that is possible in the flat case may be illegal in a hierarchic derivation relation.

Example 27. Suppose that 0 is a base constant and that the operators f, g, and b are contained in the enrichment. Consider the clauses

$$y \approx 0 \rightarrow \mathsf{f}(x,y) \approx x \tag{1}$$

$$y \approx 0 \rightarrow \mathsf{g}(\mathsf{f}(\mathsf{b},y)) \approx \mathsf{b} \tag{2}$$

In the flat case, we could simplify (2) to

$$\rightarrow \mathsf{g}(\mathsf{b}) \approx \mathsf{b}. \tag{3}$$

Since (2) follows from the smaller clauses (1) and (3), it can be removed. On the other hand, a simple substitution cannot instantiate x by b in clause (1). Thus not every simple ground instance of (2) follows from smaller simple ground instances of (1) and (3). Clause (2) is not redundant with respect to the hierarchic redundancy criterion $\mathscr{R}^{\mathscr{H}}$, hence the simplification is impossible.

It should be noted that we can simplify clauses relative to a particular base specification, although the redundancy criterion $\mathscr{R}^{\mathscr{H}}$ is independent of $\mathscr{C}$. For instance, suppose that all base algebras in $\mathscr{C}$ satisfy the base clause $C = \Gamma \rightarrow \Pi$. Since $\emptyset \models_{\mathscr{C}} C$, we may add it to N. Consequently, every clause $\Gamma, \Delta \rightarrow \Lambda, \Pi$ in N becomes redundant and can be discarded. Obviously it is very useful if the base theorem prover is not only capable of refuting sets of clauses, but if it can also simplify the base part of a clause or test whether a base clause is satisfied by all models of $\mathscr{C}$.

5 Second-Order Quantifier Elimination

Our techniques are not confined to hierarchic theorem proving for constraint algebras. We will now demonstrate how the hierarchic superposition calculus can be used to transform certain second-order formulae with existential quantifiers over predicate variables into equivalent first-order formulae. This is useful, for example, for translating formulae in modal logic into first-order formulae involving the accessibility relation of the Kripke frame (see Van Benthem [15]). Gabbay and Ohlbach [9] describe such an algorithm and several applications of quantifier elimination. Their method turns out to be essentially an instance of our technique.

Let $\mathscr{H}_0$ denote the inference system $\mathscr{H}$ without the constraint refutation rule. Then we have the following theorem:

Theorem 28. *Let N be a set of abstracted clauses that is sufficiently complete with respect to simple instances and saturated with respect to $\mathscr{H}_0$, and let A be a term-generated Ω-algebra. Then A is a model of $N \cap \mathrm{F}_\Omega$ if and only if there exists a model A' of N whose restriction to Ω is isomorphic to A.*[12]

Proof. The "if" part is obvious. For the "only if" part, let A be some Ω-model of $N \cap \mathrm{F}_\Omega$. The inference system $\mathscr{H}_0$ is independent of $\mathscr{C}$, so we may choose $\mathscr{C}$ arbitrarily. Let $\mathscr{C}$ be the isomorphism class of A, then the constraint refutation rule cannot be applied to N, and as a consequence, N is also saturated with respect to $\mathscr{H}$. Obviously $\bot \notin N \cap \mathrm{F}_\Omega \subseteq N$. Since every specification that has only a finite number of models (up to isomorphism) is compact, we can apply Theorem 24 and obtain $N \not\models_{\mathscr{C}} \bot$. So there exists a model A' of N whose restriction to Ω is in $\mathscr{C}$. □

[12] It is possible to construct one such extension in a similar way as the perfect model in (Bachmair and Ganzinger [3, 4]).

Let $\exists P_1 \cdots P_n \psi$ be a second-order formula where the P_i are predicate variables and ψ is a first-order formula. First we transform ψ into clause form; for simplicity, let us for a moment assume that this can be done without skolemization. Afterwards we split the signature into base and non-base operators, such that the P_i are the only non-base symbols, besides we add infinitely many free constant symbols to the base signature. Using variable abstraction we obtain a set N of abstracted clauses; this is always sufficiently complete with respect to simple instances, as there are no non-base operators whose codomain is a base sort. We saturate N with respect to $\mathscr{H}_0$.[13] Let N^* be the result of the saturation process. By Theorem 28 the Ω-models of $N^* \cap \mathrm{F}_\Omega$ are exactly those term-generated Ω-algebras that are restrictions of a model of N^*. Thus $N^* \cap \mathrm{F}_\Omega$ is equivalent to $\exists P_1 \cdots P_n N^*$ (which in turn is equivalent to $\exists P_1 \cdots P_n \psi$). Note however, that the saturation process is not guaranteed to terminate and that $N^* \cap \mathrm{F}_\Omega$ may be infinite.

What happens if skolemization introduces new (existentially quantified) operator symbols? Of course it does not make sense to consider these as non-base symbols, since sufficient completeness would surely be violated. So we must add them to the base signature. As described by Gabbay and Ohlbach [9], we can then try to use unskolemization techniques (which may fail, however) after N^* has been computed.

Example 29. Consider the formula

$$\exists P \exists Q \exists u \forall x \forall y \begin{pmatrix} \neg P(x) \vee R(x,x) \\ Q(u,x) \vee \neg R(x,u) \\ P(x) \vee P(y) \vee \neg Q(x,y) \end{pmatrix}. \tag{1}$$

Transformation into clause form yields the three clauses

$$P(x) \to R(x,x) \tag{2}$$

$$R(x,a) \to Q(a,x) \tag{3}$$

$$Q(x,y) \to P(x), P(y) \tag{4}$$

where a is a skolem constant. We consider P and Q as non-base symbols and R and a as base symbols, so we have to replace (3) by the abstracted clause

$$z \approx a, R(x,a) \to Q(z,x). \tag{5}$$

Using a lexicographic path ordering with the precedence $Q \succ P \succ R \succ a$, we can now saturate the set of (2), (4), and (5). Left superposition of (5) and (4) yields

$$z \approx a, R(x,a) \to P(z), P(x). \tag{6}$$

Left superposition of (6) and (2) yields the two clauses

$$z \approx a, R(x,a) \to R(z,z), P(x) \tag{7}$$

$$z \approx a, R(x,a) \to P(z), R(x,x). \tag{8}$$

Applying the left superposition rule to (7) and (2) and to (8) and (2), we obtain the same clause

$$z \approx a, R(x,a) \to R(z,z), R(x,x). \tag{9}$$

[13] Of course, the derivation relation $\vdash$ must not depend on a particular class of base models.

The set N^* consisting of the clauses (2) and (4)–(9) is saturated, hence $\exists P \exists Q\ N^*$ is equivalent to $N^* \cap \mathrm{F}_\Omega$, i.e., to clause (9). As we have introduced only skolem *constants*, unskolemization is trivial, so we can finally transform the formula (1) into

$$\exists u \forall x \neg R(x, u) \vee R(u, u) \vee R(x, x). \tag{10}$$

6 Conclusions

We have extended various superposition calculi for first-order clauses to hierarchic specifications. Our results show how a refutational theorem prover for a compact base specification can be combined in a modular way with a superposition theorem prover, provided that the enrichment is sufficiently complete with respect to simple instances. This restriction is void for Horn clauses, as they are used traditionally in constraint logic programming languages, and even for first-order clauses without equality. Bürckert [6, 7] has investigated (unordered) resolution for this case, his method, however, is more permissive concerning the syntactical form of constraints. If equations and new function symbols whose codomain is a base sort are introduced, then sufficient completeness may be rather restrictive. Thus it is not only necessary to determine decidable criteria for sufficient completeness, but it should also be investigated whether some variant of our method works with weaker requirements.

Theorem 28 is apparently very close to persistency conditions for parameterized specifications (except for the fact that in parameterized specifications non-term-generated algebras have to be considered as well, so that the base signature has to be augmented by free constant symbols). The precise relation, however, between our paper and the framework of parameterized specifications (see Kirchner [11] for the equational case) has yet to be analyzed. Furthermore the relationship to different kinds of constraint reasoning and to theory resolution (Stickel [14]) should be clarified.

Finally, it would be interesting to investigate to what extent the performance of Gabbay and Ohlbach's quantifier elimination procedure is improved in practice by our redundancy criterion $\mathscr{R}^{\mathscr{H}}$.

Acknowledgements. We wish to thank Dov Gabbay and Hans Jürgen Ohlbach for directing our attention to the quantifier elimination problem and we are grateful to the referees for their valuable detailed comments. The research described in this paper was partially funded by the DFG project "Programmier- und Beweisumgebung" (Ga 261), by the German Ministry for Research and Technology (BMFT) under grant ITS 9102/ITS 9103, and by the National Science Foundation under grant no. CCR-8901322.

References

1. Avenhaus, J., Becker, K.: Conditional rewriting modulo a built-in algebra. SEKI Report SR-92-11, Fachbereich Informatik, Universität Kaiserslautern (1992)
2. Bachmair, L., Ganzinger, H.: On restrictions of ordered paramodulation with simplification. In: Stickel, M. E. (ed.) 10th International Conference on Automated Deduction. Lecture Notes in Artificial Intelligence, Vol. **449**, pp. 427–441. Berlin, Heidelberg, New York: Springer 1990
3. Bachmair, L., Ganzinger, H.: Perfect model semantics for logic programs with equality. In: Furukawa, K. (ed.) Logic Programming, Proceedings of the Eighth International Conference, pp. 645–659. The MIT Press 1991

4. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. Tech. Rep. MPI-I-91-208, Max-Planck-Institut für Informatik, Saarbrücken (1991). Revised version to appear in J. Logic Comput.
5. Bergstra, J. A., Broy, M., Tucker, J. V., Wirsing, M.: On the power of algebraic specifications. In: Gruska, J., Chytil, M. (eds.) Mathematical Foundations of Computer Science, 10th Symposium. Lecture Notes in Computer Science, Vol. **118**, pp. 193–204. Berlin, Heidelberg, New York: Springer 1981
6. Bürckert, H.-J.: A resolution principle for clauses with constraints. In: Stickel, M. E. (ed.) 10th International Conference on Automated Deduction. Lecture Notes in Artificial Intelligence, Vol. **449**, pp. 178–192. Berlin, Heidelberg, New York: Springer 1990
7. Bürckert, H.-J.: A Resolution Principle for a Logic with Restricted Quantifiers. Lecture Notes in Artificial Intelligence, Vol. **568**. Berlin, Heidelberg, New York: Springer 1991
8. Dershowitz, N., Jouannaud, J.-P.: Rewrite systems. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics, pp. 244–320. Amsterdam, New York, Oxford, Tokyo: Elsevier Science Publishers B. V. 1990
9. Gabbay, D., Ohlbach, H. J.: Quantifier elimination in second order predicate logic. South African Computer Journal **7**, 35–43 (1992). Also appeared in 3rd International Conference on Principles of Knowledge Representation and Reasoning, pp. 425–435. 1992
10. Jaffar, J., Lassez, J.-L.: Constraint logic programming. In: Fourteenth Annual ACM Symposium on Principles of Programming Languages, pp. 111–119 (1987)
11. Kirchner, H.: Proofs in parameterized specifications. In: Book, R. V. (ed.) Rewriting Techniques and Applications, 4th International Conference. Lecture Notes in Computer Science, Vol. **448**, pp. 174–187. Berlin, Heidelberg, New York: Springer 1991
12. Kreisel, G., Krivine, J.-L.: Elements of Mathematical Logic. Amsterdam: North-Holland 1967
13. Nieuwenhuis, R.: First-order completion techniques. Technical report, Universidad Politécnica de Cataluña, Dept. Lenguajes y Sistemas Informáticos (1991)
14. Stickel, M. E.: Automated deduction by theory resolution. J. Autom. Reasoning **1**, 333–355 (1985)
15. Van Benthem, J.: Correspondence theory. In: Gabbay, D. M., Guenthner, F. (eds.) Handbook of Philosophical Logic, Vol. II: Extensions of Classical Logic, pp. 167–247. Dordrecht, Boston, Lancaster: D. Reidel 1984
16. Wirsing, M.: Algebraic specification. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, volume B: Formal Models and Semantics, pp. 675–788. Amsterdam, New York, Oxford, Tokyo: Elsevier Science Publishers B. V. 1990