Combining Algebra and Universal Algebra in First-Order Theorem Proving: The Case of Commutative Rings*

Leo Bachmair**
Harald Ganzinger***
Jürgen Stuber***

Abstract. We present a general approach for integrating certain mathematical structures in first-order equational theorem provers. More specifically, we consider theorem proving problems specified by sets of first-order clauses that contain the axioms of a commutative ring with a unit element. Associative-commutative superposition forms the deductive core of our method, while a convergent rewrite system for commutative rings provides a starting point for more specialized inferences tailored to the given class of formulas. We adopt ideas from the Gröbner basis method to show that many inferences of the superposition calculus are redundant. This result is obtained by the judicious application of the simplification techniques afforded by convergent rewriting and by a process called symmetrization that embeds inferences between single clauses and ring axioms.

1 Introduction

1.1 Motivation

Specifications of programs include both symbols with their usual mathematical meaning as well as additional function symbols that correspond to program entities. Axioms such as

$$\forall l : \texttt{sum}(\texttt{mapSquare}(l)) \geq 0$$

$$\forall x, l : \texttt{mapSquare}(x : l) = x * x : \texttt{mapSquare}(l)$$

where `mapSquare` is supposed to square the elements of a list and `sum` computes the sum of the elements of a list, involve symbols such as $+$, $*$, $\geq$ or 0 with their

* The research described in this paper was supported in part by the NSF under research grant INT-9314412, by the German Ministry for Research and Technology (Bundesministerium für Forschung und Technologie) under grant ITS 9102/ITS 9103 and by the ESPRIT Basic Research Working Group 6112 (COMPASS).
** Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794, U.S.A. leo@cs.sunysb.edu
*** Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany. {hg|juergen}@mpi-sb.mpg.de

standard mathematical meaning in addition to the functions of the program of which some behaviour is expressed.

Specifications based on standard mathematical structures, such as linear orderings, rings, or fields, require specially designed reasoning methods. A naive approach whereby one simply adds an axiomatization of a structure (e.g., the axioms of a commutative ring or the axioms of a linear order) to a specification and searches for proofs according to some universal deductive mechanism, such as resolution or paramodulation, is not very useful in practice. The axioms of the standard structure, both by themselves and in interaction with the axioms for program functions, usually create a huge search space. Consequently, in such theorem proving methods the additional knowledge provided by an embedded mathematical domain, which typically simplifies a mathematician's proof search, invariably clutters up the search space, so that many automated provers will no longer be able to find certain proofs they had been finding before.

A more promising approach appears to be the integration of decision procedures into heuristic provers and into resolution- or paramodulation-based provers, as this may allow one to more effectively cut down the proof search space. Some success has been obtained in limited scenarios. For example, replacing syntactic unification by unification modulo an equational theory allows one to effectively deal with certain classes of equational theories. Commutative semigroups have been extensively treated in approaches such as associative-commutative completion and their extensions to paramodulation-based calculi for first-order logic. This method is limited in that most nontrivial equational theories do not admit effective equation solving. But even for finitary and effectively solvable theories the number of unifiers may be excessively large, e.g., doubly exponential in the case of associativity and commutativity theories, which in fact presents a major obstacle to applying existing associative-commutative provers in practice.

Constraint-based theorem proving (Bürckert 1990, Kirchner, Kirchner and Rusinowitch 1990) is an alternative approach that is mainly applicable to hierarchic specifications, where new function symbols are defined persistently (with no junk and no confusion) over the primitive standard structures; see also (Bachmair, Ganzinger and Waldmann 1994). In requirement specifications where one may not want to define the new functions explicitly, these conditions are hardly ever met in practice. As a result, a constraint solver for the primitive theory can not be simply added as a black box, and even in cases where the new function symbols are completely defined in terms of the primitive ones, the (typically, mutually recursive) interaction between the prover and the constraint solver may present serious complications, see Boyer and Moore (1988) for a discussion of these issues in the context of their heuristic prover. The prover PVS (Owre, Rushby and Shankar 1992) is another instance where decision procedures are integrated in a heuristic manner, in this case using ideas of Nelson and Oppen (1979). In this paper we will not only be concerned with the practical aspects of such integration but also with questions of completeness of the resulting methods. Our results apply to specifications in which function symbols need not be completely defined in terms of the given primitive, mathematical symbols.

1.2 Our Approach

The deductive core of our method is a refutational, clausal-based inference system. We attempt to prove that a formula is a theorem by showing that the set consisting of its negation plus all axioms is unsatisfiable. All formulas are assumed to be in clause form. More specifically, we consider sets of clauses $N \cup R$, where R is a clausal axiomatization of the theory of commutative rings with a unit element and the clauses in N specify properties of uninterpreted function symbols.

The theorem proving method we design is *saturation-based*, like resolution and paramodulation (but unlike model elimination and other goal-oriented methods). In saturation-based methods, inferences are applied exhaustively to given clauses until either a contradiction (the empty clause) is obtained or else no further inferences are possible (and the final set is saturated). A saturation-based prover may be thought of as computing a mapping C from clause sets to clause sets, with the property that $\mathsf{C}(N)$ contains the empty clause if, and only if, N is unsatisfiable. If N is satisfiable, $\mathsf{C}(N)$ may be finite or infinite, but in either case will be closed under the given inferences. In our previous work on this subject (Bachmair and Ganzinger 1994c) we have developed a concept of *redundancy* and a refined notion of *saturation up to redundancy* so as to be able to model simplification mechanisms, such as subsumption, tautology deletion, and rewriting, that are indispensable for the efficiency of theorem provers. The refined notion of saturation does not require the application of non-redundant inferences or inferences from redundant clauses. This work is of particular relevance to the present paper as it provides the right framework for the specialized techniques we propose for dealing with certain built-in mathematical structures.

Finite, consistent sets $\mathsf{C}(N)$ often provide a basis for efficient theorem provers for the theory represented by N. An example are convergent rewrite systems, which are saturated under superposition and allow the validity of equations to be determined by (non-deterministic) rewriting. If a given clause set N can not be finitely saturated, it may still be possible to saturate some subset of N. For instance, if the given clause set is $N \cup R$, and R can be saturated, we may "partially evaluate" the saturation function and replace R by $\mathcal{R}$. The saturation of the partially evaluated set $N \cup \mathcal{R}$ then involves only inferences where at least one premise comes from N. The main inferences in our deductive calculus are binary, i.e., have two premises. Further analysis may be profitable also for "mixed" inferences with one premise from N and one from $\mathcal{R}$. If, for a clause C, one can find a set $\mathcal{S}(C)$, such that $\mathcal{S}(C) \cup \mathcal{R}$ is saturated, then no explicit inferences need to be made with C, but C may be replaced by $\mathcal{S}(C)$. This may pay off if the mapping $\mathcal{S}$ can be more efficiently computed by a special-purpose device than by off-the-shelf saturation. We will describe, for the case of commutative rings, such a function $\mathcal{S}$, called *symmetrization*, that maps any ground clause to a finite set, and design inference rules to enumerate possibly infinite sets $\mathcal{S}(C)$ for non-ground clauses. A key ingredient of the symmetrization process is the use of $\mathcal{R}$ for simplification of clauses in N. Such simplification also cuts down on the number of inferences between two clauses in N.

In sum, our approach aims at a refinement of the general saturation scheme in which the axioms of a built-in theory are replaced by more efficient special-purpose inferences that may be used for (don't-care nondeterministic) simplification.

1.3 The Case of Commutative Rings

We apply this methodology to commutative rings and present the ring structure by a convergent rewrite system. For the free commutative ring over a set of constants the word problem can be decided by applying a specialized variant of the Gröbner basis method (Buchberger 1965, Buchberger 1985, Kandri-Rody and Kapur 1988). We essentially embed this procedure in the more general context of languages with additional function symbols other than constants. Though decidability is lost for the extended languages, our goal is to adapt the techniques for the more specialized decision procedures and replace inferences with ring axioms by deterministic simplification procedures.

We do not simply use the Gröbner basis method as a black box, but achieve its integration with the general-purpose deductive mechanism of superposition on the level of the inference systems. That is, we use a rewrite-based reformulation of the Gröbner basis method to manipulate formulas in a way so as to cut down the number of general superposition inferences. More specifically, we attempt to eliminate the addition function from terms that need to be considered for superposition (and hence would require associative-commutative unification). For example, during the computation of a Gröbner basis a polynomial equation

$$xy^2z + 2xy = x + 3$$

is transformed into a rewrite rule

$$xy^2z \to -2xy + x + 3$$

with a monomial on its left-hand side. The effect is that inferences with rules, which depend only on left-hand sides, need not deal with addition. Symmetrization is the generalization of this transformation of a polynomial equation into a rule. It is the key idea in combining universal algebra and algebra "on the level of inferences." To make it work sufficiently well and to be able to obtain interesting theoretical results, such as refutational completeness, a number of technical problems need to be solved.

Most importantly, even though the elimination of occurrences of addition in the manner indicated above is readily seen to be logically sound, one needs to show that the transformation is compatible with the superposition calculus in the sense that refutational completeness is preserved. The problem looks deceptively simple, but is non-trivial. It is at this point that our concept of redundancy proves to be very useful. Refutational completeness, as we will see, is preserved if the transformation of a formula can be justified by *smaller* formulas with respect to an appropriate ordering on clauses. One of the main technical problems is to find a suitable ordering. A closely related problem is that of selecting an appropriate canonical presentation of the ring axioms.

Another problem is that we need to deal with clauses with variables, whereas polynomials, as manipulated in the Gröbner basis method, are essentially ground expressions (the indeterminates in a polynomial are constants in the logical sense). Many simplification steps on the ground level can not be replicated on the general level and need to be approximated via specially designed inference rules.

2 Preliminaries

2.1 Clauses

We consider first-order languages with equality as the only predicate symbol. Thus the only atomic formulas are equalities between terms built from variables and function symbols.[4] An important tool in our proposed theorem proving method are well-founded orderings. We typically specify such an ordering on the term level and then uniformly extend it to formulas and clauses via multiset extensions. Therefore we formally define formulas and clauses in terms of multisets.

By an *atomic formula* (or *atom*) we mean a multiset (of multisets) $\{\{s\},\{t\}\}$, called an *equality* and usually written $s \approx t$, where s and t are terms.[5] Equalities are also called *positive literals*, whereas by a *negative* literal we mean a multiset $\{\{s,t\}\}$, written $s \not\approx t$ or $\neg(s \approx t)$. A *clause* is a finite multiset of literals. We write a clause as a disjunction $\neg A_1 \vee \cdots \vee \neg A_m \vee B_1 \cdots \vee B_n$. An expression is said to be *ground* if it contains no variables.

2.2 Reduction Orderings

A (strict) *ordering* is a transitive and irreflexive binary relation; a *quasi-ordering*, a reflexive and transitive binary relation. A *rewrite relation* is a binary relation $\succ$ on terms such that $s \succ t$ implies $u[s\sigma] \succ u[t\sigma]$, for all terms s, t, $u[s\sigma]$, and $u[t\sigma]$ in the given domain.[6] An *equivalence* (*relation*) is a reflexive, transitive, and symmetric binary relation. A *congruence* (*relation*) is an equivalence such that $s \sim t$ implies $u[s] \sim u[t]$, for all terms s, t, $u[s]$, and $u[t]$ in the given domain.

The reflexive closure of a strict ordering is a quasi-ordering. On the other hand, for any quasi-ordering $\succeq$ we may define a corresponding equivalence ($s \sim t$ if $s \succeq t$ and $t \succeq s$) and strict ordering ($s \succ t$ if $s \succeq t$ but not $t \succeq s$). A quasi-ordering thus induces a strict ordering on the equivalence classes defined by its associated equivalence. Formally, we will be dealing mainly with quasi-orderings

4 Non-equality atoms $P(t_1, \ldots, t_n)$ can be represented by equating them to a distinguished constant $\top$ (that is of a different sort than the ordinary terms t_i).

5 The symmetry of equality is thus built into the notation.

6 In our notation for terms we follow Dershowitz and Jouannaud (1990).

with associated equivalences containing a given associative-commutative equational theory. But for simplicity, we will usually specify the strict part of the ordering only.

A *rewrite ordering* is an ordering that is also a rewrite relation. A well-founded rewrite ordering is called a *reduction ordering.* (A binary relation $\succ$ is *well-founded* if there is no infinite sequence $t_1 \succ t_2 \succ \ldots$ of elements.)

Any ordering $\succ$ on a set S can be extended to an ordering on finite multisets over S (which for simplicity we also denote by $\succ$) as follows: $M \succ N$ if (i) $M \neq N$ and (ii) whenever $N(x) > M(x)$ then $M(y) > N(y)$, for some y such that $y \succ x$. (Here $>$ denotes the usual "greater-than" relation on the natural numbers.) If $\succ$ is a total (resp. well-founded) ordering, so is its multiset extension. If $\succ$ is an ordering on terms, then its twofold multiset extension is an ordering on literals, and its threefold multiset extension, an ordering on clauses. We usually use the same symbol to denote an ordering $\succ$ on terms and its corresponding extensions to literals and clauses. Note that our definition of positive and negative literals guarantees that $\neg A \succ A$, for all atoms A.

2.3 Herbrand Interpretations

A (*Herbrand*) *interpretation* is a set I of ground atomic formulas. We say that an atom A is *true* (and $\neg A$ *false*) *in* I if $A \in I$; and that A is *false* (and $\neg A$ *true*) in I if $A \notin I$. A ground clause is *true* in an interpretation I if at least one of its literals is true in I; and is *false* otherwise. A (non-ground) clause is said to be true in I if all its ground instances are true. The *empty clause* is false in every interpretation. We say that I is a *model* of a set of clauses N (or that N is *satisfied* by I) if all elements of N are true in I. A clause C is called a *logical consequence* of N if C is true in all models of N. A set N is *satisfiable* if it has a model, and *unsatisfiable* otherwise. For instance, any set containing the empty clause is unsatisfiable.

We denote by RA the set consisting of the reflexivity axiom $x \approx x$, by T the set consisting of the transitivity axiom $x \not\approx y \vee y \not\approx z \vee x \approx z$, and by F the set consisting of all congruence axioms $x \not\approx y \vee f(\ldots, x, \ldots) \approx f(\ldots, y, \ldots)$ where f ranges over all function symbols.

An interpretation is called an *equality interpretation* if it satisfies $RA \cup T \cup F$. (Recall that symmetry is already built into the notation.) We say that I is an *equality model* of N if it is an equality interpretation satisfying N. A set N is *equality satisfiable* if it has an equality model, and *equality unsatisfiable* otherwise. It can be shown that a set of clauses is equality satisfiable in our sense if and only if it is has a standard first-order model in which equality is interpreted as identity.

2.4 Associative-Commutative Rewriting

We have to reason about the satisfiability of clause sets $N \cup R \cup AC$, where $R \cup AC$ is a clausal specification of commutative rings. We describe Herbrand interpretations by (ground) rewrite systems; deal with interpretations indirectly, via rewriting; and design general inference rules that reflect these rewrite manipulations. The form of rewriting most appropriate for the class of problems we consider is associative-commutative rewriting.

If I is an equality interpretation then the set of all pairs (s,t) for which $s \approx t$ is true in I is a congruence relation on ground terms. Conversely, if $\sim$ is a congruence relation, then the set of all ground atoms $s \approx t$, for which $s \sim t$, is an equality interpretation. In short, equality interpretations can be described by congruence relations. Rewrite systems provide an effective tool for dealing with congruence relations.

A *rewrite system* is a binary relation on terms. The elements of a rewrite system are called (*rewrite*) *rules* and written $s \to t$. If R is a rewrite system, we denote by $\to_R$ the smallest rewrite relation containing R. The transitive-reflexive closure of $\to_R$ is denoted by $\to_R^*$; while $\leftrightarrow_R^*$ denotes the smallest congruence relation containing R. A rewrite system R is said to be *terminating* if the rewrite relation $\to_R$ is well-founded.

A function symbol f is said to be *associative* if it satisfies the axiom

$$f(x, f(y,z)) \approx f(f(x,y),z)$$

and *commutative* if it satisfies

$$f(x,y) \approx f(y,x).$$

Let AC be a set of such axioms. We write $f \in AC$, if AC contains the associativity and commutativity axioms for f, and also use AC to denote the set of rewrite rules $f(x,f(y,z)) \to f(f(x,y),z)$ and $f(x,y) \to f(y,x)$. We say that two terms u and v are *AC-equivalent* if $u \leftrightarrow_{AC}^* v$.

A reduction ordering $\succ$ is *AC-compatible* if $u' \leftrightarrow_{AC}^* u \succ v \leftrightarrow_{AC}^* v'$ implies $u' \succ v'$, for all terms u, u', v, and v'. If $\succ$ is AC-compatible, we write $u \succeq v$ to indicate that $u \succ v$ or $u \leftrightarrow_{AC}^* v$. We say that a rewrite system R is *AC-terminating* if it is contained in an AC-compatible reduction ordering.

Associativity and commutativity are built into rewriting via matching. If R is a rewrite system, we denote by $AC\backslash R$ the set of all rules $u' \to v\sigma$, such that $u' \leftrightarrow_{AC}^* u\sigma$ for some rule $u \to v$ in R and some substitution σ. The rewrite relation $\to_{AC\backslash R}$ corresponds to rewriting by R via AC-matching. We say that a term t can be *rewritten* (*modulo AC*), or is *AC-reducible*, if $t \to_{AC\backslash R} t'$. Terms that cannot be rewritten are said to be in *(AC-)normal form* or *(AC-)irreducible* (with respect to R). We write $u \downarrow_{AC\backslash R} v$ if u and v can be rewritten to AC-equivalent terms, i.e., if there exist terms u' and v' such that $u \to_{AC\backslash R}^* u' \leftrightarrow_{AC}^* v' \leftarrow_{AC\backslash R}^* v$. If R is AC-terminating and we have $u \downarrow_{AC\backslash R} v$, for all terms u and v with $u \leftrightarrow_{AC\cup R}^* v$, then R is called *AC-convergent.*

It is a standard result from the theory of term rewriting (see Dershowitz and Jouannaud 1990 for details and further references) that an AC-terminating rewrite system R is AC-convergent if (i) $u \downarrow_{AC\backslash R} v$ whenever $u \leftarrow_{AC\backslash R} t \leftrightarrow_{AC} v$ (a property called *local AC-coherence* of R) and (ii) $u \downarrow_{AC\backslash R} v$ whenever $u \leftarrow_{AC\backslash R} t \to_R v$ (a property called *local AC-confluence* of R).

Any rewrite system R can easily be extended to a locally AC-coherent system by a technique proposed by Peterson and Stickel (1981). An *extended rule* is a rewrite rule of the form $f(x,u) \to f(x,v)$, where $f \in AC$, u is a term $f(s,t)$, and x is a variable not occurring in u or v. We also say that $f(x,u) \to f(x,v)$ is an *extension* of $u \to v$. If R is a rewrite system, we denote by R^e the set R plus all extensions of rules in R. Any rewrite system R^e is locally AC-coherent: if $u \leftarrow_{AC\backslash R^e} t \leftrightarrow_{AC} v$, then $u \leftrightarrow^*_{AC} u' \leftarrow_{AC\backslash R^e} v$, for some term u'.

Checking for local AC-confluence is more difficult and usually involves the computation of so-called AC-critical pairs, which are obtained by the inference rule of AC-Superposition described next.

3 Associative-Commutative Superposition

3.1 Preliminaries

Two key aspects of the clause sets we have to deal with are the presence of equality and of associative-commutative function symbols. The standard inference system for dealing with equality in a clausal setting is paramodulation. The calculus we use as the deductive basis for our method is associative-commutative superposition (Bachmair and Ganzinger 1993). Superposition, as presented in (Bachmair and Ganzinger 1994c), is a refinement of paramodulation in which inferences are controlled by a selection function and a reduction ordering.

We call a literal L *maximal* in a clause $C \vee L$ if $L \not\prec L'$ for all literals L' in C. A *selection function* is a mapping that assigns to each clause a subset (possibly empty) of its negative literals. These negative literals are said to be *selected*. We assume that at least one literal is selected in each clause in which the maximal literal is negative. No literal needs to be selected in a clause whose maximal literal is positive. We say that a clause $C \vee s \approx t$ is *reductive* for $s \approx t$ (with respect to $\succ$) if there exists a ground instance $(C \vee s \approx t)\sigma$ such that $s\sigma \succ t\sigma$ and $(s \approx t)\sigma \succ C\sigma$ and no literal in C is selected. A clause C is called *universally reductive* if all ground instances of C are reductive and no literal in C is selected.

We say that $C \vee f(x,s) \approx f(x,t)$ is an *extension* of $C \vee s \approx t$ if $f \in AC$, s is a term $f(u,v)$, $C \vee s \approx t$ is reductive for $s \approx t$ and x is a variable not occurring in C, s, or t. In this case $C \vee f(x,s) \approx f(x,t)$ is called an *extended clause*. In an extended clause no literal may be selected.

Associativity and commutativity are built into the inference rules via AC-unification. Two terms u and v are *AC-unifiable* if $u\sigma \overset{*}{\leftrightarrow}_{AC} v\sigma$, for some substitution σ. If two terms u and v are AC-unifiable, then there exists a *complete set of AC-unifiers* $CSU_{AC}(u,v)$, such that for any substitution σ with $u\sigma \overset{*}{\leftrightarrow}_{AC} v\sigma$ there exist substitutions $\tau \in CSU_{AC}(u,v)$ and ρ, such that $x\sigma \overset{*}{\leftrightarrow}_{AC} (x\tau)\rho$, for all variables x in u and v. We assume that a function CSU_{AC} is given.

3.2 Inference Rules

The calculus $\mathsf{S}^{\succ}_{AC}$ consists of the following inference rules. We assume that the premises of an inference share no common variables. If necessary the variables in one premise need to be renamed.

AC-Superposition $$\frac{C \vee p \approx q \qquad D \vee u[p'] \approx r}{(C \vee D \vee u[q] \approx r)\sigma}$$

where (i) $\sigma \in CSU_{AC}(p, p')$, (ii) the clause $(C \vee p \approx q)\sigma$ is reductive for $(p \approx q)\sigma$, (iii) the clause $(D \vee u \approx r)\sigma$ is reductive for $(u \approx r)\sigma$, and (iv) p' is not a variable.

Negative AC-Superposition $$\frac{C \vee p \approx q \qquad D \vee u[p'] \not\approx r}{(C \vee D \vee u[q] \not\approx r)\sigma}$$

where (i) $\sigma \in CSU_{AC}(p, p')$, (ii) the clause $(C \vee p \approx q)\sigma$ is reductive for $(p \approx q)\sigma$, (iii) $u[p'] \not\approx r$ is a selected literal in $D \vee u[p'] \not\approx r$, (iv) $u\sigma \not\preceq r\sigma$, and (v) p' is not a variable.

Reflective AC-Resolution $$\frac{C \vee p \not\approx q}{C\sigma}$$

where (i) $\sigma \in CSU_{AC}(p, q)$, and (ii) $p \not\approx q$ is a selected literal in the premise.

AC-Factoring $$\frac{C \vee p \approx q \vee p' \approx q'}{(C \vee q \not\approx q' \vee p' \approx q')\sigma}$$

where (i) $\sigma \in CSU_{AC}(p, p')$, (ii) $q\sigma \not\prec q'\sigma$, (iii) the literal $(p \approx q)\sigma$ is maximal in $C\sigma$, and (iv) the premise contains no selected literals.

The calculus $\mathsf{S}^{\succ}_{AC}$ is refutationally complete for clause sets $N \cup AC$. Other paramodulation-based calculi with built-in associativity and commutativity (Wertz 1992, Nieuwenhuis and Rubio 1994, Vigneron 1994) also embed relatively sophisticated control techniques in their inference rules. However, the efficiency of theorem provers usually depends not so much on the deductive inference rules that are employed than on the simplification mechanisms it implements. The concept of redundancy described next is sufficiently general and also flexible enough so that all necessary simplification techniques can be formulated conveniently.

3.3 Redundancy

Simplification techniques, such as tautology deletion, subsumption, demodulation, contextual rewriting, etc., represent an essential component of automated theorem provers. These techniques are based on a concept of redundancy (Bachmair and Ganzinger 1994c), which we have adapted to the AC-case in (Bachmair and Ganzinger 1993). The notion of redundancy will be of particular importance in the context of theorem proving for clause sets $N \cup R \cup AC$, as it will enable us to apply simplification techniques for ring theory in a meaningful way to the general clauses in N. In essence, we will extend techniques employed in constructing Gröbner bases for commutative rings to first-order clauses.

For any ground term p, let $T_{\preceq p}$ be the set of all ground instances $p_1 \not\approx p_2 \vee p_2 \not\approx p_3 \vee p_1 \approx p_3$ of the transitivity axiom for which $p \succeq p_i$ for $i \in 1 \ldots 3$; and $T_{\prec p}$ be the set of all ground instances for which $p \succ p_i$ for $i \in 1 \ldots 3$.

Let N be a set of clauses and C be a ground clause. Our intention is to specify when C is not needed for a potential refutation of $N \cup AC$. A ground clause C (which need not be an instance of N) is said to be *AC-redundant* with respect to N (and ordering $\succ$) if there exist ground instances $C_1, \ldots, C_k$ of N, such that $C \succ C_j$ for all $j \in 1 \ldots k$, and C is true in every model of $AC \cup RA \cup F \cup T_{\preceq p} \cup \{C_1, \ldots, C_k\}$, where p is the maximal term in C. A non-ground clause is called AC-redundant if all its ground instances are.

We emphasize that AC-redundancy is defined with respect to arbitrary interpretations, not just equality interpretations. In particular, there are restrictions on the use of the transitivity axiom. The restrictions are of more theoretical than practical significance, as all the simplification techniques that we will need can be formalized using AC-redundancy. In (Wertz 1992) and (Nieuwenhuis and Rubio 1994) a different notion of redundancy is used, where restrictions are imposed, not on transitivity, but instead on the use of the associativity and commutativity axioms. Our definition has the advantage that reduction modulo AC is a simplification, even in cases where the maximal term of a clause is reduced.

For example, consider the (unit) clauses $s[u'] \approx t$ and $u \approx v$, where $u' \leftrightarrow^*_{AC} u\sigma$, for some substitution σ. If $u\sigma \succ v\sigma$ and either $s \succ u\sigma$ or else $t \succ v\sigma$, then $s[u'] \approx t$ is AC-redundant with respect to any set N containing $u \approx v$ and $s[v\sigma] \approx t$. In practice, this allows one to replace $s[u'] \approx t$ by $s[v\sigma] \approx t$ in the presence of the equation $u \approx v$. In other words, the subterm u' can be rewritten modulo AC to a smaller term $v\sigma$. Simplification by rewriting modulo AC is admissible.

Let π be a ground inference with conclusion B and maximal premise C and let p be the maximal term in C. We say that π is *AC-redundant* with respect to N if there exist ground instances $C_1, \ldots, C_k$ of N, such that (a) B is true in every model of $AC \cup RA \cup F \cup T_{\prec p} \cup \{C_1, \ldots, C_k\}$, and either (b.i) both premises of π are ground instances of extended clauses and $\{\{\{p\}\}\} \succ C_j$,[7] for all j, or else (b.ii) $C \succ C_j$, for all j. A non-ground inference is called AC-redundant if all its ground instances are AC-redundant.

One way to render an inference in $\mathsf{S}^{\succ}_{AC}$ redundant is to add its conclusion to the set N. Thus, one may saturate any given set of clauses by systematically computing all possible inferences. (Since new clauses may enable new inferences, this process may continue indefinitely and result in an infinite saturated set.)

We say that a set of clauses N is *saturated up to AC-redundancy* if all inferences, the premises of which are non-redundant clauses in N or extensions of non-redundant clauses in N, are AC-redundant.

[7] Note that $\{\{\{p\}\}\} \succ C_j$ simply means that all terms in C_j are smaller than p.

Theorem 1 Bachmair and Ganzinger 1993. *Let N be a set of clauses that is saturated up to AC-redundancy with respect to the associative-commutative superposition calculus $\mathsf{S}^{\succ}_{AC}$. Then $N \cup AC$ is equality unsatisfiable if and only if it contains the empty clause.*

This theorem provides the basis for our refutational theorem proving method. It formally establishes the refutational completeness of the calculus $\mathsf{S}^{\succ}_{AC}$. To determine whether a given set of clauses is unsatisfiable, simply saturate it. If the set is indeed unsatisfiable, the empty clause (a contradiction) will eventually be generated.

It may seem strange that extended clauses are redundant, but inferences with them are not. But note that the above theorem requires extensions of *non-redundant* clauses only. Since extended clauses are redundant, extensions of extensions are consequently not needed. (It is important to avoid extensions of extensions, as the process may continue indefinitely and thus preclude the possibility of finitely saturating any non-trivial set of clauses.)

4 Commutative Rings

We assume that the first-order language under consideration contains function symbols 0, 1, $-$, $+$ and $\cdot$, which are intended to denote the operations of a commutative ring with unit, plus an additional constant -1 meant to denote $-(1)$. These are called *interpreted*, and all other function symbols are called *uninterpreted*. Also, terms with an interpreted or uninterpreted function symbol at the top will be called *interpreted* or *uninterpreted*, respectively. Moreover, we will also consider variables as uninterpreted terms.

The variety of commutative rings with a unit element can be specified by the set of equalities $R \cup AC$, where AC contains the associativity and commutativity axioms for addition and multiplication,

$$
\begin{aligned}
x + y &\approx y + x \\
x + (y + z) &\approx (x + y) + z \\
x \cdot y &\approx y \cdot x \\
x \cdot (y \cdot z) &\approx (x \cdot y) \cdot z
\end{aligned}
$$

and R is the set of equations

$$
\begin{aligned}
x + 0 &\approx x \\
-x + x &\approx 0 \\
x \cdot 1 &\approx x \\
x \cdot (y + z) &\approx (x \cdot y) + (x \cdot z)
\end{aligned}
$$

The defining equation for the constant -1 is

$$-(1) \approx -1.$$

Using standard techniques from associative-commutative rewriting, we obtain an AC-convergent rewrite system for the theory defined by the above equations. By $\mathcal{R}$ we denote the following set of rules:

$$x + 0 \to x \tag{1}$$
$$x \cdot 0 \to 0 \tag{2}$$
$$x \cdot 1 \to x \tag{3}$$
$$x \cdot (y + z) \to (x \cdot y) + (x \cdot z) \tag{4}$$
$$-x \to -1 \cdot x \tag{5}$$
$$1 + (-1) \to 0 \tag{6}$$
$$(-1) \cdot (-1) \to 1 \tag{7}$$
$$x + (-1 \cdot x) \to 0 \tag{8}$$

Lemma 2. *The rewrite system $\mathcal{R}^e$ is AC-convergent for the theory $R \cup AC$ over any language containing the function symbols from R plus any number of additional uninterpreted function symbols.*

The rewrite system $\mathcal{R}^e$ is similar to other convergent systems for commutative rings, except that we have extended the language by the constant -1 and rewrite $-x$ to a product, $-1 \cdot x$. From now on we may assume that terms and clauses do not contain the unary $-$ anymore.

Extensions are actually needed only for the rules (6)–(8), but not for (1)–(4). (The latter rules serve as their own extensions, so to say. For example, the extended rule $y + (x + 0) \to y + x$ is AC-equivalent to an instance, $(y + x) + 0 \to y + x$ of the original rule; for details see Bachmair 1991.)

From now on, we will denote arbitrary terms by p, q, r, u or v and uninterpreted terms by s or t. Terms of the form $t_1 \cdots t_m$ with all t_i different from -1 are called *products* and denoted by ϕ or ψ; terms of the form ϕ or $(-1) \cdot \phi$, *signed products* and denoted by μ or ν. In the notation for [signed] products we also admit $m = 0$, that is, a product denoted by ϕ may be 1, and a signed product μ may be 1 or -1. We often use juxtaposition to denote multiplication and simply write pq instead of $p \cdot q$. With this nomenclature, the normal forms defined by $\mathcal{R}^e$ are sums of signed products, $\mu_1 + \cdots + \mu_n$ where the uninterpreted terms in each signed product μ_i are in normal form. It should also be understood that we may have $n = 0$ in which case the sum is 0.

We will also use the following notation reminiscent of the algebraic notation for polynomials. If c is a positive integer and $t_1, \ldots, t_c$ are terms AC-equivalent to t, we write t^c to denote any product AC-equivalent to $t_1 \cdots t_c$ and ct to denote any sum AC-equivalent to $t_1 + \cdots + t_c$. Furthermore, we write dt, where d is the negative integer $-c$, to denote any sum AC-equivalent to $(-1)t_1 + \cdots + (-1)t_c$. Finally, if c is the number 0 and t is a product, then ct denotes 0 and $s + ct$ denotes s. For example, the term $((x \cdot f(x + y)) \cdot x) + ((x \cdot x) \cdot f(y + x))$ can be more concisely written as $2x^2 f(x + y)$.

If the given language contains no uninterpreted function symbols other than finitely many constants $X_1, \ldots, X_n$, then the *ground* expressions

$$c_1 X_1^{k_{11}} \cdots X_n^{k_{1n}} + \cdots + c_m X_1^{k_{m1}} \cdots X_n^{k_{mn}}$$

correspond to polynomials over indeterminates $X_1, \ldots, X_n$. We are specifically interested in adapting polynomial methods, such as Buchberger's algorithm for constructing Gröbner bases, to more general languages with function symbols and variables.

Termination

The AC-convergence of a rewrite system is established by three properties: local AC-coherence, local AC-confluence, and AC-termination. Extended rules ensure the first property. The second property requires checking of AC-critical pairs, which presents no difficulty in the case of $\mathcal{R}^e$; we refer the reader to the rewriting literature, e.g., (Dershowitz and Jouannaud 1990). For the third property we specify a suitable AC-compatible reduction ordering containing $\mathcal{R}$. This ordering will also play an important role in our deductive theorem proving method.

Theorem 3. *There exists an AC-compatible reduction ordering that is total up to AC on ground terms and contains all rewrite rules in $\mathcal{R}^e$.*

An ordering that fits the bill is any associative path ordering (APO, Bachmair and Plaisted 1985, Bachmair and Dershowitz 1986) based on a precedence on function symbols in which $f_1 \succ \cdots \succ f_n \succ - \succ -1 \succ 1 \succ 0 \succ \cdot \succ +$, where $f_1, \ldots, f_n$ are the given uninterpreted function symbols. Note that any such precedence is admissible for the associative path ordering, as there are no more than two AC-symbols and these are smaller than all other symbols. The uninterpreted function symbols may be listed in any order, but need to be assigned higher precedence than the interpreted symbols. Henceforth let $\succ$ be an APO based on such a precedence.

Convergent rewrite systems are also saturated:

Lemma 4. *$\mathcal{R}^e$ is saturated up to AC-redundancy with respect to $\mathsf{S}^{\succ}_{AC}$.*

In sum, the set $N \cup R \cup AC$ can be replaced by the equivalent set $N \cup \mathcal{R}^e \cup AC$. This provides us with at least two advantages. First, since $\mathcal{R}^e$ is already saturated, no further inferences need to be computed on it. (The replacement of R by $\mathcal{R}^e$ may also be thought of as a compilation process.) Secondly, the rewrite rules in $\mathcal{R}^e$ may be used to simplify clauses in N. The latter aspect will be discussed in the next section.

5 Specialized Simplification Techniques

So simplify the presentation for inferences, our meta-level notation will from now be modulo $AC1$ for $\cdot$ and $+$. That is, when we write $p = c\phi + p'$ then $c\phi$ denotes c products which are AC-equivalent to ϕ and which occur somewhere in the sum, not necessarily in front. Moreover, neither $c\phi$ nor p' need to be present. Thus p may also be of the form $c\phi$, p' or 0, depending on whether $p = 0$ and/or $c = 0$. Similarly, when we write $\phi = \phi_1\phi_2$ then ϕ_1 or ϕ_2 may be missing in which case $\phi_i = 1$.

This section lists the main simplifications such as cancellation that are afforded by the ring axioms. They are presented as rules of the

$$\frac{\{C\} \cup N}{\{D\} \cup N}.$$

Such a rule is called a $\mathcal{R}$-*simplification rule* whenever D is a logical consequence of $\{C\} \cup N \cup \mathcal{R}$ and C is AC-redundant in $\{D\} \cup N \cup \mathcal{R}$. A simplification rule, hence, denotes a legal move of the theorem prover during saturation by which C is replaced by D as the addition of D renders C redundant. Simplification rules may be applied eagerly in a don't care non-deterministic fashion without affecting refutational completeness.

In the simplification rules below, $\bowtie$ denotes either $\approx$ or $\not\approx$.

$\mathcal{R}$-Reduction $$\frac{\{u[l] \bowtie v \vee C\} \cup N}{\{u[r] \bowtie v \vee C\} \cup N}$$

if (i) $l \to_{AC\setminus\mathcal{R}^e} r$ and (ii) $u[l] \bowtie v \vee C \succ l \approx r$.

Cancellation $$\frac{\{c\phi + p \bowtie d\phi + q \vee C\} \cup N}{\{(c-d)\phi + p \bowtie q \vee C\} \cup N}$$

if (i) $|c| \geq |d| > 0$, and (ii) $\mathrm{sign}(c) = \mathrm{sign}(d)$.

Accumulation $$\frac{\{c\phi + p \bowtie -d\phi + q \vee C\} \cup N}{\{(c+d-1)\phi + p \bowtie (-1)\phi + q \vee C\} \cup N}$$

if $c \geq 1$ and $d \geq 2$.

Isolation $$\frac{\{c\phi + p \bowtie d\phi + q \vee C\} \cup N}{\{c\phi \bowtie d\phi + q + (-1)p \vee C\} \cup N}$$

if (i) $c > 0$ and $d = 0$ or $c = -1$ and $d \geq 0$, and (ii) $\phi \succ p$ and $\phi \succ q$.

Standardization $$\frac{\{(-1)\phi \approx q \vee C\} \cup N}{\{\phi \approx (-1)q \vee C\} \cup N}$$

if $\phi \approx (-1)q \vee C$ is universally reductive for $\phi \approx (-1)q$.

Lemma 5. *$\mathcal{R}$-Reduction, Cancellation, Accumulation, Isolation and Standardization are $\mathcal{R}$-simplification rules.*

Proof. This is easily verified. In the case of Standardization $\phi \approx (-1)q \vee C$ is used to reduce $(-1)\phi \approx q \vee C$. □

Note that for Accumulation $(c+1)\phi + p \bowtie q \vee C$ is in general *not* a simplification of $c\phi + p \bowtie (-1)\phi + q \vee C$. Although the former is smaller than the latter in the ordering on clauses, to prove $(c+1)\phi + p \bowtie q \vee C$ from $c\phi + p \bowtie -1\phi + q \vee C$ requires the instance $\phi + (-1)\phi \approx 0$ of $\mathcal{R}$, which is greater than $c\phi + p \bowtie -1\phi + q \vee C$.

6 Symmetrization

The *Symmetrization* of a ground clause C is the effect of partially saturating $\mathcal{R}^e \cup \{C\}$ until all inferences between the ring axioms and the original clause C, as well as the ones that have been derived from C as a result of the saturation process, become redundant. Symmetrization essentially consists of two phases. In the first phase, the clause is simplified until one of a few different so-called *sufficiently simplified* forms is reached. In the second phase, superposition inferences into $\mathcal{R}$ are taken care of by adding a suitable set of clauses.

6.1 Sufficiently Simplified Ground Clauses

In the previous section we have described a set of simplification rules which should be applied whenever possible. For a ground clause this will have the effect that, among others, the larger side of equations becomes irreducible by $\mathcal{R}^e$. Moreover, if $+$ occurs as the top symbol of the larger side, then the larger side is a sum of identical products, a monomial so to say. This syntactic form drastically restricts superposition inferences. Hence we would like to achieve this effect for all ground instances of any non-ground clause. It suffices to simplify those literals that may possibly engage in a superposition inference, and hence are either negative and selected, or else positive and contain the maximal product of their clause. To achieve our aim, certain instances of the above simplification rules will have to be lifted and restated as inferences that apply to general clauses. Before we actually define the lifted inferences, we shall first describe their ground instances. The additional ordering and selection restrictions avoid the computation of normal forms for literals that cannot engage in superposition.

Let p be a sum of signed products $c_1\phi_1 + \cdots + c_n\phi_n$ with coefficients c_i of the form 1 or -1. A signed product $c\psi$ is called *maximal in* p if $\psi \not\prec \phi_j$ for all j; and *strictly maximal in* p if $\psi \not\preceq \phi_j$ for all j. A signed product μ is called *maximal in a literal* $p \bowtie q$ if μ is maximal in p and q; and *maximal in a clause* C if μ is maximal in all literals of C. Note that on the ground level the ordering is total up to AC-equivalence. Thus the conditions for maximality or strict maximality translate to $\psi \succeq \phi_j$ or $\psi \succ \phi_j$, respectively.

Ground Product Reduction $$\frac{\mu[l] + p \bowtie q \vee C}{\mu[r] + p \bowtie q \vee C}$$

where (i) $l \rightarrow_{AC\backslash\mathcal{R}^e} r$, (ii) μ is maximal in p and q, and (iii) $\bowtie$ is $\not\approx$ and $\mu[l] + p \not\approx q$ is selected or else $\bowtie$ is $\approx$ and μ is maximal in C.

All the inferences are restricted so that only the selected literal or any positive literal that contains the maximal product of the premise needs to be considered. (iii), which formalizes this restriction, will apply, mutatis mutandis, to all the subsequent inferences and will, from now on, no longer be explicitly mentioned.

Ground Sum Contraction $$\frac{c\phi + (-d)\phi + p \bowtie q \vee C}{(c-d)\phi + p \bowtie q \vee C}$$

where (i) $c > 0$ and $d > 0$, (ii) ϕ is strictly maximal in p and maximal in q, and (iii) ϕ is irreducible with respect to $\mathcal{R}^e$.

Ground Cancellation $$\frac{c\phi + p \bowtie d\phi + q \vee C}{(c-d)\phi + p \bowtie q \vee C}$$

where (i) $|c| \geq |d| > 0$ and $\mathrm{sign}(c) = \mathrm{sign}(d)$, (ii) ϕ is strictly maximal in p and q, and (iii) ϕ is irreducible with respect to $\mathcal{R}^e$.

Ground Accumulation $$\frac{c\phi + p \bowtie -d\phi + q \vee C}{(c+d-1)\phi + p \bowtie (-1)\phi + q \vee C}$$

where (i) $c \geq 1$ and $d \geq 2$, (ii) ϕ is strictly maximal in p and q, and (iii) ϕ is irreducible with respect to $\mathcal{R}^e$.

Ground Isolation $$\frac{c\phi + p \bowtie d\phi + q \vee C}{c\phi \bowtie d\phi + q + (-1)p \vee C}$$

where (i) $c > 0$ and $d = 0$ or $c = -1$ and $d \geq 0$, (ii) ϕ is strictly maximal in p and q, and (iii) ϕ is irreducible with respect to $\mathcal{R}^e$.

Ground Standardization $$\frac{(-1)\phi \approx q \vee C}{\phi \approx (-1)q \vee C}$$

if (i) ϕ is irreducible with respect to $\mathcal{R}^e$, and (ii) $\phi \approx (-1)q \vee C$ is reductive for $\phi \approx (-1)q$.

With the exception of Ground Sum Reduction, the above inferences are special cases of the simplification rules in Section 5. Ground Sum Reduction can be obtained as a sequence of reductions with one of the inverse laws (6) or (8).

Applying these inferences exhaustively to a ground literal, we obtain an equivalent literal L in one of two possible so-called *sufficiently simplified forms* $L = c\phi \bowtie r$ or $L = (-1)\phi \bowtie (c-1)\phi + r$, with ϕ irreducible by $\mathcal{R}^e$, $\phi \succ r$, and $c > 0$. More generally, we call an arbitrary literal L *sufficiently simplified* if L is of the form $c\phi \bowtie r$ or $(-1)\phi \bowtie (c-1)\phi + r$, with ϕ irreducible by $\mathcal{R}^e$, ϕ strictly maximal in r, and $c > 0$. A (possibly non-ground) clause is called *sufficiently simplified* if it has a selected literal that is sufficiently simplified, or else if all positive literals that contain a maximal product are sufficiently simplified. The above inferences sufficiently simplify any given ground clause.

6.2 Superpositions into $\mathcal{R}$

We now analyze the critical pairs that are generated when $\{L\} \cup \mathcal{R}^e$ is saturated where L is a sufficiently simplified literal. $\mathcal{S}(L)$, called the *symmetrization* of L, will denote the set of equations which we obtain, and is given below by a case analysis on the structure of L. Let L be of the form (i) $c\phi \approx r$ or else (ii) $(-1)\phi \approx (c-1)\phi + r'$, in which case we denote $(-1)r'$ by r. The definition of symmetrization will also apply to the non-ground case where we may have more than one strictly maximal product. Hence we only assume that ϕ is strictly maximal in L and irreducible by $\mathcal{R}^e$.

$$\mathcal{S}(L) = \{x \approx 0\} \qquad \text{if } c = 1 \text{ and } \phi = 1\text{; or}$$

$$\mathcal{S}(L) = \{\phi \approx r\} \qquad \text{if } c = 1 \text{ and } \phi \neq 1\text{; or}$$

$$\mathcal{S}(L) = \left\{\begin{matrix} cx \approx 0 \\ (-1) \approx (c-1)1 \end{matrix}\right\} \qquad \text{if } c \geq 2 \text{ and } \phi = 1\text{; or}$$

$$\mathcal{S}(L) = \left\{\begin{matrix} c\phi \approx r \\ c(x \cdot \phi) \approx x \cdot r \\ (-1) \cdot \phi \approx (c-1)\phi + (-1)r \end{matrix}\right\} \text{if } c \geq 2 \text{ and } \phi \neq 1.$$

If L is negative and of the above form there are no superpositions into positive literals. Since the maximal side $c\phi$ or $(-1)\phi$ of L is $\mathcal{R}^e$-reduced, there are also no superpositions into L by the rules in $\mathcal{R}^e$. Hence $\mathcal{S}(L)$ is simply defined as $\{L\}$.

Consider now a general clause C that is sufficiently simplified and of the form $C' \vee C''$ where either $C'' = L_1 = p \not\approx q$ and $p \not\approx q$ is selected in C; or else no literal is selected in C and $C'' = L_1 \vee \ldots \vee L_n$ contains all positive literals which contain the maximal product of C. As we have to symmetrize all the L_i, we define

$$\mathcal{S}(C) = \{\, C' \vee L_1^s \vee \ldots \vee L_n^s \mid L_i^s \in \mathcal{S}(L_i) \text{ for } i \in 1 \ldots n \,\}.$$

Symmetrization is defined only for clauses that are sufficiently simplified. A clause C is said to be *symmetrized* if it is in some $\mathcal{S}(D)$. Note that an AC-extension is needed for reductive clauses in $\mathcal{S}(C)$ where the maximal literal has a $+$ or $\cdot$ at the root of the left side. We shall also assume that if a literal is selected in C then the corresponding literal is selected in any of the symmetrizations of C.

Lemma 6. *If C is a ground clause then all superposition inferences in which one of the premises is in $\mathcal{S}(C)$ and the other is in $\mathcal{R}$ are redundant in $\mathcal{S}(C) \cup \mathcal{R}$.*

This lemma can be shown by a somewhat tedious but straightforward case analysis over all inferences.

6.3 Lifting Ground Simplification

In this section we are concerned with lifting ground simplification, which is the preparative step for symmetrization, to the non-ground level. The lifting becomes awkward in cases when a maximal product $xt_1 \ldots t_n$ in a sum contains a factor x that is just a variable. Such a variable may have to be instantiated with a sum of arbitrary length containing positive as well as negative terms. If $n \geq 1$ some of these instances will be computed by superposing with the distributivity rule, cf. Product Superposition below. For the remaining cases we will have to add an extra inference rule to achieve this effect. Clearly, application of these inferences, which are reminiscent of paramodulation with the functional-reflexive axioms for $+$ and $-$, needs to be controlled. It seems as if one could adopt a lazy strategy, by which these *unshielded occurrences* of a variable x are instantiated to a general sum just before they become instantiated by some other inference. For instance, cancelling ba on both sides of $xa \approx ba + c$ should produce $x'a \approx c$ rather than $0 \approx c$, representing the combined effect of instantiating x by $x' + x''$, unifying $x''a$ with ba and cancelling ba. Because of unshielded variables, cancellation and similar inferences have to proceed in a step-wise fashion.

Product Superposition $$\frac{\mu[l'] + p \bowtie q \vee C}{(\mu[r] + p \bowtie q \vee C)\sigma}$$

where (i) $l \to r$ is a rule in $\mathcal{R}^e$, (ii) $\sigma \in CSU_{AC}(l, l')$, (iii) l' is not a variable, (iv) $\mu\sigma$ is maximal in $p\sigma$ and $q\sigma$, and (v) $\bowtie$ is $\not\approx$ and $\mu[l'] + p \bowtie q$ is selected in $u[l'] + p \bowtie q \vee C$ or else $\bowtie$ is $\approx$ and $\mu\sigma$ is maximal in $C\sigma$.

Analogous to the ground case, the restriction (v), to the selected literal or to those positive literals that contain a maximal product, will, mutatis mutandis, apply to the subsequent inference rules. From now on, it will no longer be explicitly mentioned.

Sum Contraction 1 $$\frac{\mu + \phi + p \bowtie q \vee C}{(p \bowtie q \vee C)\sigma}$$

where (i) $\sigma \in CSU_{AC}(\mu, (-1)\phi)$, (ii) $\mu\sigma$ is maximal in $p\sigma$ and $q\sigma$, and (iii) $\mu\sigma$ is irreducible with respect to $\mathcal{R}^e$.

Sum Contraction 2 $$\frac{\mu_1 + \mu_2 + p \bowtie q \vee C}{(2\mu_1 + p \bowtie q \vee C)\sigma}$$

where (i) $\sigma \in CSU_{AC}(\mu_1, \mu_2)$, (ii) $\mu_1\sigma$ is maximal in $p\sigma$ and $q\sigma$, and (iii) $\mu_1\sigma$ is irreducible with respect to $\mathcal{R}^e$.

Cancellation $$\frac{\mu_1 + p \bowtie \mu_2 + q \vee C}{(p \bowtie q \vee C)\sigma}$$

where (i) $\sigma \in CSU_{AC}(\mu_1, \mu_2)$, (ii) $\mu_1\sigma$ is maximal in $p\sigma$ and $q\sigma$, and (iii) $\mu_1\sigma$ is irreducible with respect to $\mathcal{R}^e$.

Accumulation $$\frac{p \bowtie (-1)\phi_1 + (-1)\phi_2 + q \vee C}{(\phi_1 + p \bowtie (-1)\phi_1 + q \vee C)\sigma}$$

where (i) $\sigma \in CSU_{AC}(\phi_1, \phi_2)$, (ii) $\phi_1\sigma$ is maximal in $p\sigma$ and $q\sigma$, and (iii) $\phi_1\sigma$ is irreducible with respect to $\mathcal{R}^e$.

Isolation $$\frac{c\phi + p \bowtie d\phi + q \vee C}{c\phi \bowtie d\phi + q + (-1)p \vee C}$$

where (i) ϕ is strictly maximal in p and q, and (ii) ϕ is irreducible with respect to $\mathcal{R}^e$.

Standardization $$\frac{(-1)\phi \approx q \vee C}{\phi \approx (-1)q \vee C}$$

where (i) $\phi \approx (-1)q \vee C$ is reductive for $\phi \approx (-1)q$ and (ii) ϕ is irreducible with respect to $\mathcal{R}^e$.

Instantiation 1 $$\frac{x + p \approx q \vee C}{(x + p \approx q \vee C)\sigma}$$

where (i) σ sends x to $y + z$ with new variables y and z, and (ii) x is maximal in p and q.

Instantiation 2 $$\frac{\phi x + p \approx q \vee C}{(\phi x + p \approx q \vee C)\sigma}$$

where (i) σ sends x to $(-1)y$ with a new variable y, and (ii) ϕx is maximal in p and q.

The Instantiation and Sum Contraction 2 inferences are in addition problematic in that they are usually not *decreasing*. Hence the usual criterion of redundancy for inferences will never be satisfied, and saturation cannot be effectively achieved. These inferences have, therefore, to be equipped with a more restrictive notion of redundancy. They are called *AC-redundant* in a set of clauses N, if their respective conclusion is redundant in N, or else contained in N. Let us denote the inference system consisting of the inference rules Product Superposition, Sum Contraction 1 and 2, Cancellation, Accumulation, Isolation, Standardization, and Instantiation 1 and 2 by $\mathsf{Simp}^{\succ}_{\mathcal{R}}$. The following lemma expresses that the inference rules in $\mathsf{Simp}^{\succ}_{\mathcal{R}}$ allow to lift simplification in ring structures from the ground level to the first-order level.

Lemma 7. *Let N be a set of clauses which is saturated up to $\mathcal{R}$-redundancy with respect to $\mathsf{Simp}^{\succ}_{\mathcal{R}}$. Let C be a ground instance of a clause in N.*

1. *C is either redundant or else it is sufficiently simplified.*
2. *If C is non-redundant then C is an instance by σ of a clause $\hat{C}$ in N which is (i) sufficiently simplified, (ii) each maximal product ϕ in C is an instance $\hat{\phi}\sigma$ of a maximal product $\hat{\phi}$ in $\hat{C}$, and (iii) if ϕ and ψ are two maximal products in $\hat{C}$ which are different (modulo AC) then $\phi\sigma$ and $\psi\sigma$ are different products in C.*
3. *If N contains all symmetrizations of non-redundant and sufficiently simplified clauses in N then any $\mathsf{S}^{\succ}_{AC}$-inference between a clause in N and a rule in $\mathcal{R}^e$ is redundant.*

Proof. To show (1), suppose that C is not sufficiently simplified. Then C can be simplified to D by some ground simplification, which can be lifted to an inference in $\mathsf{Simp}^{\succ}_{\mathcal{R}}$. Since this inference is redundant, D is also a ground instance of N and C is redundant.

For (2), suppose that C is the ground instance $C'\sigma$ of a clause C' in N and that C' is not of the desired form. Then an inference with C' is possible, and since this inference is redundant but C itself is not redundant, N contains a clause $\hat{C}$ of the desired form.

(3) follows from Lemma 6. □

The significance of (2) lies in the fact that monomials on the ground level correspond to monomials on the level of general clauses. This is essential for being able to lift the inferences in the Gröbner base computation to equations with variables. To illustrate this, let us consider the example

$$ax + y + f(bz) + z \approx (-1)y + c \tag{9}$$

where $f \succ a \succ b \succ c$. This equation contains the unshielded variables x and y, and the shielded variable z. Maximal products are ax, y and $f(bz)$ and consequently many inferences are possible.

Product superpositions into the (signed) product ax in (9) are possible with the rules (2), (3) and (4) in $\mathcal{R}$, which yield

$$0 + y + f(bz) + z \approx (-1)y + c \tag{10}$$
$$a + y + f(bz) + z \approx (-1)y + c$$
$$ax_1 + ax_2 + y + f(bz) + z \approx (-1)y + c, \tag{11}$$

where x is instantiated to 0, 1 or $x_1 + x_2$, respectively. Similar inferences are also possible for $(-1)y$ and below the uninterpreted function symbol f for bz. Also, the instantiation rules permit y to be instantiated to $y_1 + y_2$ or $(-1)y'$. To continue, we pick (11) and consider Sum Contraction inferences with that equation. Signed products obtained from ax_1, ax_2 and y are AC-unifiable in any combination, and we get

$$y + f(bz) + z \approx (-1)y + c \tag{12}$$
$$ax_2 + f(bz) + z \approx (-1)ax' + c$$
$$ax_1 + f(bz) + z \approx (-1)ax' + c$$
$$2ax_1 + y + f(bz) + z \approx (-1)y + c$$
$$2ax_1 + ax_2 + f(bz) + z \approx (-1)y + c$$
$$ax_1 + 2ax_2 + f(bz) + z \approx (-1)y + c.$$

This time we pick (12)[8] and consider cancellation inferences. The only one is the one where z and $(-1)y$ are unified, which results in

$$y + f(bz) \approx c,$$

but we continue with (12). Accumulation is not applicable, since $(-1)y$ occurs only once on the right side. Isolation gives, with the choice of $(-1)y$ as the maximal product (and sides reversed),

$$(-1)y \approx y + f(bz) + z + (-1)c.$$

Standardization is not applicable here, since y occurs also on the right-hand side.

[8] We also obtain this clause more directly by choosing (10) in the previous step and simplification. This example shows that still a lot of redundant inferences are made. A careful analysis could exclude this case, but it would not be possible to represent the information about previous inferences in a standard first-order language. In a framework of superposition with constraints this superposition into the previously substituted part of a clause could be prevented by putting $x_1 \not\approx 0$ and $x_2 \not\approx 0$ into the constraint. With ordering constraints it would also be possible to enforce a consistent choice of the maximal product throughout a sequence of inferences.

7 Inferences on Sufficiently Simplified Clauses

At this point we have treated all AC-superposition inferences where at least one premise is from $\mathcal{R}^e$. It remains to consider the case where both premises are in N. We may assume that the original clauses have been replaced by the symmetrizations of their simplified forms, and now we have to analyse the overlaps between these. Notice that ground symmetrization produces up to 6 different versions of a literal. Many more versions of a clause are generated if more than one literal contains a maximal product. Simply computing all possible AC-superpositions between the various versions of a clause would give us a complete but not very efficient strategy. Indeed, it turns out that only a small fraction of all possible overlaps needs to be computed. This is described by the following inference rules:

Top Superposition $$\frac{c_1(y_1\phi) \approx y_1 p \vee C \qquad c_2(y_2\psi) \approx y_2 q \vee D}{((c_2 - c_1)(y_1\phi) + y_1 p \approx y_2 q \vee C \vee D)\sigma}$$

where (i) y_1 and y_2 are (different) variables which do not occur in ϕ, ψ, p, q, C or D, or denote 1[9] (ii) $\sigma \in CSU_{AC}(y_1\phi, y_2\psi)$ such that $y_1\sigma$ and $y_2\sigma$ share no common factor $\neq 1$, (iii) $(c_1(y_1\phi) \approx y_1 p \vee C)\sigma$ is symmetrized and reductive with respect to $(c_1(y_1\phi) \approx y_1 p)\sigma$, (iv) $(c_2(y_2\psi) \approx y_2 q \vee D)\sigma$ is symmetrized and reductive with respect to $(c_2(y_2\psi) \approx y_2 q)\sigma$, (v) no premise has a selected literal, and (vi) $c_2 \geq c_1$.

Deep Superposition $$\frac{p \approx q \vee C \qquad c\phi[p'] \approx r \vee D}{(c\phi[q] \approx r \vee C \vee D)\sigma}$$

where (i) $\sigma \in CSU_{AC}(p, p')$, (ii) $(p \approx q \vee C)\sigma$ is symmetrized and reductive with respect to $(p \approx q)\sigma$, (iii) $(c\phi[p'] \approx r \vee D)\sigma$ is symmetrized and reductive with respect to $(c\phi[p'] \approx r)\sigma$, (iv) the occurrence of p' in ϕ is not a variable and is is below an uninterpreted function symbol, and (v) no premise has a selected literal.

Negative Superposition $$\frac{p \approx q \vee C \qquad u[p'] \not\approx r \vee D}{(u[q] \not\approx r \vee C \vee D)\sigma}$$

where (i) $\sigma \in CSU_{AC}(p, p')$, (ii) $(p \approx q \vee C)\sigma$ is symmetrized and reductive with respect to $(p \approx q)\sigma$, and (iii) $u[p'] \approx r \vee D$ is symmetrized, (iv) $u[p'] \approx r$ is selected in $u[p'] \approx r \vee D$, (v) $u\sigma \not\preceq r\sigma$, and (vi) p' is not a variable.

Let $\mathsf{S}^{\succ}_{\mathcal{R}}$ consist of $\mathsf{Simp}^{\succ}_{\mathcal{R}}$ plus the inference rules Top Superposition, Deep Superposition, Negative Superposition, Reflective AC-Resolution and AC-Factoring. This inference system places strong restrictions on superposition at and above uninterpreted function symbols in the top region of terms. Superposition below uninterpreted function symbols and Negative Superposition is restricted only in that we require that the premises be symmetrized. In the present formulation of the inference system we have to consider all symmetrizations of the first premise, however. More work is needed to further strengthen the respective restrictions.

[9] We are saying that the meta-notation for the premises and the conclusion concerning the occurrences of variables y_1 and y_2 is modulo $AC1$.

Theorem 8. *Let $N \cup \mathcal{R}$ be a set of clauses that is saturated up to AC-redundancy with respect to $\mathsf{S}^{\succ}_{\mathcal{R}}$ and that contains all symmetrizations for any non-redundant and sufficiently simplified clause in N. Then $N \cup \mathcal{R}$ is saturated up to AC-redundancy with respect to $\mathsf{S}^{\succ}_{AC}$.*

Proof. We need to show that all AC-Superposition and all Negative AC-Superposition inferences from premises in $N^e \cup \mathcal{R}^e$ are AC-redundant.

If at least one premise is from $\mathcal{R}^e$ then such an inference is redundant by Lemma 7, since N is saturated with respect to $\mathsf{Simp}^{\succ}_{\mathcal{R}}$ and contains the symmetrizations of non-redundant, sufficiently simplified clauses.

It remains to consider the case where both premises are from N. If one of the premises is not symmetrized, it is redundant by Lemma 7 and the inference is redundant, too. Hence we need to consider only inferences between symmetrized clauses C_1 and C_2. Superposition inferences are restricted to the case where either L_2 is a selected literal in C_2 and C_1 is reductive for L_1, or where both C_1 and C_2 are reductive for L_1 and L_2, respectively.

A ground literal has, including extensions, up to six different forms after symmetrization. We consider the two most general cases where each L_i is of one of the two forms

$$x_i + c_i(y_i\phi_i) \bowtie x_i + y_i r_i \tag{13}$$

and

$$(-1)(y_i\phi_i) \bowtie (c_i - 1)(y_i\phi_i) + (-1)y_i r_i, \tag{14}$$

respectively, carrying extension variables x_i and y_i with respect to both $\cdot$ and $+$, respectively. Note that x_i and y_i do not occur inside ϕ_i, r_i or C'_i. Other forms may be seen as special cases, where extension variables are omitted, or where $c_i = 1$, or else where ϕ_i is missing. The analysis of these cases is essentially similar.

We distinguish three cases: (i) Two positive literals superpose, and the superposition is not below an uninterpreted function symbol, (ii) a positive literal superposes into another positive literal, but below an uninterpreted function symbol, and (iii) a positive literal superposes into a negative literal.

(i) Suppose L_1 and L_2 are positive literals.

(a) We start by considering the case where two ground clauses C_i of form (13) superpose.[10] Since AC-contexts can be moved into the substitution for the extension variables x_i and y_i, we need to consider only superpositions at the root in which the peak is an instance of both left-hand sides. The inference is symmetric, hence we may assume without loss of generality that $c_2 \geq c_1$.

If the sums $c_i y_i \phi_i$ did not overlap in some product, the superposition would be trivially redundant, as the substitution for the extension variables x_i would be reducible (Figure 1a). Since at least one product overlaps it follows that $(y_1\phi_1)\sigma =_{AC} (y_2\phi_2)\sigma$ (Figure 1b). By the same argument as before, the products ϕ_1 and ϕ_2 overlap in some term. We obtain the most general nontrivial

[10] This corresponds to the computation of S-polynomials in the Gröbner basis algorithm.

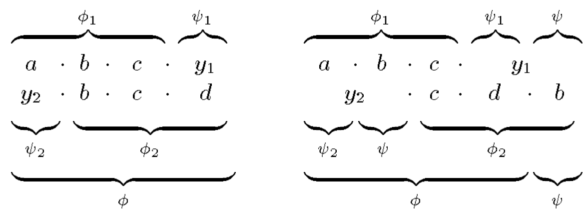
(a) A trivial overlap:
$$\begin{array}{l} abcy_1 + abcy_1 + \qquad\qquad x_1 \\ \qquad x_2 \qquad + bcdy_2 + bcdy_2 + bcdy_2 \end{array}$$

(b) A nontrivial overlap:
$$\begin{array}{l} abcy_1 + abcy_1 + \qquad x_1 \\ \quad x_2 \quad + bcdy_2 + bcdy_2 + bcdy_2 \end{array}$$

(c) A maximal overlap:
$$\begin{array}{l} abcy_1 + abcy_1 + \;\; x_1 \\ bcdy_2 + bcdy_2 + bcdy_2 \end{array}$$

Fig. 1. Overlaps of sums

overlap $(x_1 + c_1(y_1\phi_1))\sigma =_{AC} (x_2 + c_2(y_2\phi_2))\sigma$ where $x_i\sigma = (c - c_i)(y_i\sigma\phi_i)$ for some c. Consider the case where as many products as possible overlap, which implies $c = c_2$. Then $x_2\sigma = 0$ which corresponds to the use of the non-extended symmetrization $c_2(y_2\phi_2) \approx y_2r_2 \vee C_2'$ of C_2 (Figure 1c). Similarly, we require that ϕ_1 and ϕ_2 overlap as much as possible, i.e., $y_1\sigma$ and $y_2\sigma$ have no common factor. By making these restrictions we identify the smallest possible peak. The inference corresponding to this peak is the only one that actually needs to be computed, as it causes all other inferences to become redundant. This is exactly analogous to the situation in the Gröbner basis algorithm, where also only maximal overlaps are considered. In order to refer to this smallest peak in the following, let $\phi = y_1\sigma\phi_1$ and $\psi_i = y_i\sigma$ for $i \in 1, 2$ (Figure 2a). The superposition



(a) A maximal overlap (b) A nonmaximal overlap

Fig. 2. Overlaps of products

inference in question is of the form

$$\frac{x_1 + c_1(y_1\phi_1) \approx x_1 + y_1r_1 \vee C_1' \qquad c_2(y_2\phi_2) \approx y_2r_2 \vee C_2'}{(c_2 - c_1)\phi + \psi_1r_1 \approx \psi_2r_2 \vee C_1' \vee C_2'} \tag{15}$$

Its conclusion can be obtained by top superposition with essentially the same premises, in particular the second, maximal premise is the same, and hence it is redundant.

We now show that other overlaps between clauses in the symmetrizations of the C_i are redundant, whenever inference (15) is redundant. Let us first consider other overlaps between the same two clauses.

(b) Assume that $y_1\sigma$ and $y_2\sigma$ share a common factor $\psi \neq 1$. Then $y_i\sigma = \psi\psi_i$ for $i = 1, 2$ (Figure 2b) and the inference is

$$\frac{x_1 + c_1(y_1\phi_1) \approx x_1 + y_1r_1 \vee C_1' \qquad c_2(y_2\phi_2) \approx y_2r_2 \vee C_2'}{(c_2 - c_1)\psi\phi + \psi\psi_1r_1 \approx \psi\psi_2r_2 \vee C_1' \vee C_2'} \tag{16}$$

The conclusion can be obtained from the conclusion of (15) by multiplying both sides with ψ, using the congruence law for multiplication, and using distributivity. The question is, whether this instance of distributivity is smaller than the maximal premise. The APO distributes $\cdot$ over $+$ and thus the left-hand side $\psi((c_2 - c_1)\phi + \psi_1r_1)$ of the distributivity rule becomes $(c_2 - c_1)(\psi\phi) + \psi\psi_1r_1$ which is smaller than $c_2(\psi\phi)$ in the maximal premise since there are $c_1 > 0$ occurrences less of the maximal power product in it. Since the bound on the redundancy proof for the inference (15) is lower than the bound here, we can reuse it. We conclude that the inference (16) is redundant.

(c) Assume that only $c_0 < c_1$ sum terms from C_1 overlap with C_2. We also allow that $y_1\sigma$ and $y_2\sigma$ share a common factor ψ as in (b). Then the inference is

$$\frac{x_1 + c_1(y_1\phi_1) \approx x_1 + y_1r_1 \vee C_1' \qquad x_2 + c_2(y_2\phi_2) \approx x_2 + y_2r_2 \vee C_2'}{(c_2 - c_0)\psi\phi + \psi\psi_1r_1 \approx (c_1 - c_0)\psi\phi + \psi\psi_2r_2 \vee C_1' \vee C_2'}$$

To show redundancy of this inference, we use the congruence law for $+$ to add $(c_1 - c_0)\psi\phi$ on each side of the conclusion of (16), for which we already have shown redundancy. The redundancy proof of (16) is with respect to a smaller maximal premise than here, so we are again allowed to reuse it.

(d) We now consider the overlap of a clause with a negative left-hand side into a clause which is extended with respect to multiplication. For the same reason as before, we need to consider only overlaps at the root of a product. Here we have $(-1)(y_1\phi_1) =_{AC} y_2\phi_2 =_{AC} (-1)\phi\psi$ and thus $y_1\sigma = \psi\psi_1$ and $y_2\sigma = (-1)\psi\psi_2$. We consider an arbitrary nontrivial but not necessarily maximal overlap and thus allow that $y_1\sigma$ and $y_2\sigma$ have a common factor ψ. We get the inference

$$\frac{(-1)(y_1\phi_1) \approx (c_1 - 1)(y_1\phi_1) + (-1)y_1r_1 \vee C_1' \qquad c_2(y_2\phi_2) \approx y_2r_2 \vee C_2'}{(c_1 - 1)\psi\phi + (-1)\psi\psi_1r_1 + (c_2 - 1)(-1)\psi\phi \approx (-1)\psi\psi_2r_2 \vee C_1' \vee C_2'}$$

Using the inverse law, the conclusion can be reduced to $(-1)\psi\psi_1r_1 + (c_2 - c_1)(-1)\psi\phi \approx (-1)\psi\psi_2r_2 \vee C_1' \vee C_2'$. We can multiply both sides of (16) by -1 and use distributivity to obtain the conclusion. Analogous to (b) the instance of distributivity used is again smaller than the maximal premise.

(e) If we overlap two symmetrizations with a negative left-hand side, that is, of the form $(-1)y_i\phi_i \approx y_i((c_i - 1)\phi_i + (-1)r_i) \vee C_i'$ for $i \in 1, 2$, then we obtain two more cases.

(e1) If the two occurrences of (-1) superpose then the overlapped term is $(-1)\phi\psi =_{AC} (-1)y_1\sigma\phi_1 =_{AC} (-1)y_2\sigma\phi_2$ and again we have $y_i\sigma = \psi\psi_i$ for $i = 1, 2$. This corresponds to the inference

$$\frac{(-1)y_1\phi_1 \approx y_1((c_1 - 1)\phi_1 + (-1)r_1) \vee C_1' \qquad (-1)y_2\phi_2 \approx y_2((c_2 - 1)\phi_2 + (-1)r_2) \vee C_2'}{\psi\psi_1((c_1 - 1)\phi_1 + (-1)r_1) \approx \psi\psi_2((c_2 - 1)\phi_2 + (-1)r_2) \vee C_1' \vee C_2'}$$

The conclusion can be rewritten to $(c_1 - 1)\phi\psi + \psi\psi_1(-1)r_1 \approx (c_2 - 1)\phi\psi + \psi\psi_2(-1)r_2 \vee C_1' \vee C_2'$ and, using the congruence law, simplified to $\psi\psi_1(-1)r_1 \approx (c_2 - c_1)\phi\psi + \psi\psi_2(-1)r_2$. Using instances of the inverse law (8) smaller than $\phi\psi$, we may move $\psi\psi_1(-1)r_1$ and $\psi\psi_2(-1)r_2$ to the other side of the equation. The result is $\mathcal{R}$-equivalent to the conclusion of (16) and thus redundant.

(e2) Otherwise, the peak becomes $(-1)(-1)\phi\psi$ which can be reduced by the smaller rule $(-1)(-1) \to 1$ in $\mathcal{R}$. Inferences in which the maximal term is reducible by a rule smaller than any of the premises are redundant.

(ii) Now suppose the superposition is below some uninterpreted function symbol. In this case it makes essentially no difference whether the premise is extended or not. For simplicity we carry out the proof for clauses which are not extended.

(a) Suppose the superposition is into a clause of the form (13), the inference is

$$\frac{l_1 \approx r_1 \vee C_1' \qquad c_2\phi_2[l_1\sigma] \approx r_2 \vee C_2'}{(c_2 - 1)\phi_2[l_1\sigma] + \phi_2[r_1\sigma] \approx r_2 \vee C_1' \vee C_2'}$$

where $l_1 \approx r_1 \vee C_1'$ may be any clause in the symmetrization and the occurrence of l_1 in ϕ_2 is below an uninterpreted function symbol. The substitution σ is needed to instantiate extension variables which may occur in l_1 and r_1 but not in C_1'. The other copies of $l_1\sigma$ in the conclusion $(c_2 - 1)\phi_2[l_1\sigma] + \phi_2[r_1\sigma] \approx r_2 \vee C_1' \vee C_2'$ can also be rewritten and we obtain $c_2\phi_2[r_1\sigma] \approx r_2 \vee C_1' \vee C_2'$. This is an instance of the conclusion of a Deep Superposition inference, which has the same premises and is redundant since N is saturated. Hence this inference is also redundant.

(b) If on the other hand the superposition is into a clause of the form (14), we get the inference

$$\frac{l_1 \approx r_1 \vee C_1' \qquad (-1)\phi_2[l_1\sigma] \approx (c_2 - 1)\phi_2[l_1\sigma] + (-1)r_2 \vee C_2'}{(-1)\phi_2[r_1\sigma] \approx (c_2 - 1)\phi_2[l_1\sigma] + (-1)r_2 \vee C_1' \vee C_2'} \tag{17}$$

where again the other copies of $l_1\sigma$ in the conclusion can be rewritten and we obtain $(-1)\phi_2[r_1\sigma] \approx (c_2 - 1)\phi_2[r_1\sigma] + (-1)r_2 \vee C_1' \vee C_2'$. Using the instance $\phi_2[r_1\sigma] + (-1)\phi_2[r_1\sigma] \approx 0$ of the inverse law, which is smaller than the maximal premise since the maximal product has been reduced, we obtain $0 \approx c_2\phi_2[r_1\sigma] + (-1)r_2 \vee C_1' \vee C_2'$. We may move $(-1)r_2$ to the other side and arrive at $c_2\phi_2[r_1\sigma] \approx r_2 \vee C_1' \vee C_2'$. This again is an instance of the conclusion of a Deep Superposition inference, which is redundant since N is saturated. Since its maximal premise $c_2\phi_2 \approx r_2 \vee C_2'$ is smaller than the maximal premise $(-1)\phi_2 \approx (c_2 - 1)\phi_2 + (-1)r_2 \vee C_2'$ of (17), this implies that (17) is redundant.

(iii) Let us now consider superpositions into a negative literal. In this case the symmetrization is obtained by simplification only, hence only one of the forms of a symmetrized clause needs to be present. All non-redundant inferences of this kind are lifted to Negative Superposition, which is Negative AC-Superposition restricted to symmetrized clauses. $\square$

Corollary 9. *Let $N \cup \mathcal{R}$ be a set of clauses that is saturated up to AC-redundancy with respect to $\mathsf{S}^{\succ}_{\mathcal{R}}$ and that contains all symmetrizations for any* *non-redundant and sufficiently simplified clause in N. Then $N \cup \mathcal{R} \cup AC$ is equality unsatisfiable if and only if it contains the empty clause.*

8 A Simple Example

Let us conclude this section with an example. In boolean rings, each element is its own additive inverse. That is, if $xx \approx x$ for all x then $y + y \approx 0$ for all y. In clause form we get

$$xx \approx x \tag{18}$$

$$a + a \not\approx 0 \tag{19}$$

where a is a skolem constant. There are redundant ring theory superposition inferences of (18) with (2) yielding $0 \approx 0$, with (3) yielding $1 \approx 1$, with (4) yielding $((y + z)y) + ((y + z)z) \approx yy + yz + yz + zz$ and after reduction $yy + yz + yz + zz \approx yy + yz + yz + zz$, and a nonredundant one with (6) yielding

$$1 \approx -1. \tag{20}$$

Symmetrization produces

$$x + x \approx 0 \tag{21}$$

which can be used to derive the empty clause from (19).

9 Conclusion

We have proposed a specialized variant of the associative-commutative superposition calculus for commutative rings. We consider our investigation as a first step towards the integration of algebra and universal algebra in saturation-based theorem provers for first-order logic. We have shown that specialized simplification techniques eliminate the need for a large class of superposition inferences which involve the ring axioms. The number of inferences between other clauses is also considerably reduced by a process which we have called symmetrization, following (Le Chenadec 1984), and which represents a partial saturation of a clause with respect to the axioms of the theory. We have also shown that for top-level inferences between symmetrized clauses the effect of the many inferences between the possibly many symmetrizations of two clauses can be cast into on single, rather simple inference. The concept of saturation up to redundancy forms the main basis on which simplification and other optimization techniques may be justified.

This work is related to previous research in which the similarity between Buchberger's algorithm and Knuth-Bendix completion has been exploited. This similarity appears to have first been observed by (Loos 1981) and (Buchberger and Loos 1983). Since then a number of researchers have attempted to clarify, and formalize, the connection between the two methods; see (Buchberger 1987) for a discussion of the fundamental issues and further references. The free commutative ring over a set of free constants is isomorphic to a polynomial ring with coefficients in the integers. This in turn is a special case of a polynomial ring over an Euclidian coefficient ring which was investigated by (Kandri-Rody and Kapur 1988). The essential steps of one variant of their algorithm can be recovered from our calculus when we specialize it to unit equations over free constants. Our method is also an extension of work by (Marché 1994) who considers (possibly failing) Knuth-Bendix completion procedures for equations that contain a particular theory represented by a convergent rewrite system. In contrast, we deal with the non-failing case in the general setting of first-order clauses. Our own previous work on the relation between Gröbner basis methods and superposition (Bachmair and Ganzinger 1994a) has shown how to view the Gröbner basis computation for polynomials over a coefficient field B as a specialized strategy of $AC \cup B$-superposition. The concept of AC-superposition was extended so as to be able to model computation with coefficients as unification modulo the ground theory of B. We have also argued for a constraint-based approach in cases where the coefficient structure is merely a ring. The present paper considerably extends some of the ideas in the latter paper.

Future work will try to improve and extend the results of the present paper in various ways. For instance we would like to consider the case of fields for which one obtains a finite and saturated first-order presentation by simply adding the clause $x \approx 0 \vee x * x^{-1} \approx 1$ to the presentation of ring theory that we have been using here. The method of the present paper should easily extend to that case which can not be handled in a purely equational framework. We also hope to be able to combine our chaining methods for transitive (non-symmetric) relations with the method presented here. A problem of a similar nature has been solved in (Bachmair and Ganzinger 1994b) where we have shown, among other things, how to integrate the quantifier elimination procedure for dense total orderings without endpoints into chaining-based first-order theorem proving.

In this paper we have attempted to reuse previous results about the refutational completeness of our AC-superposition calculus. The completeness proof for the latter is based on a certain model construction technique which we did not want to repeat here. Now it seems to us as if we should rather have started from scratch and should have based the present investigation on a specific form of rewriting in rings. This would have induced an according modification of the model construction. On the other hand it might have saved us a lot of pain related to the concept of symmetrization. Symmetrized variants of clauses, like extensions in the AC-case, should better not be treated as first-class citizens. They should remain hidden. Moreover, replacing $(-1)\phi \approx (c-1)\phi+p$ by $c\phi \approx -p$ ought to be a simplification. This would be the case if all partial models in a modified model construction were rings so that redundancy proofs could apply ring axioms in an unrstricted manner. In such a setting Negative Superposition can be considerably restricted further. Finally, let us again mention the extremely prolific instantiation inferences for which one should try to find effective restrictions and control strategies. All in all we hope that it will be possible to arrive at a setting in which certain awkward technicalities of the present paper will disappear.

References

L. Bachmair and N. Dershowitz, 1986. Commutation, transformation, and termination. *Proc. 8th Conf. on Automated Deduction*, San Diego, CA, LNCS 230, pp. 5–20. Springer.

L. Bachmair and H. Ganzinger, 1993. Associative-commutative superposition. Technical Report MPI-I-93-267, Max-Planck-Institut für Informatik, Saarbrücken. To appear in Proc. CTRS Workshop 1994.

L. Bachmair and H. Ganzinger, 1994a. Buchberger's algorithm: A constraint-based completion procedure. *Proc. 1st Int. Conf. on Constraints in Computational Logics*, Munich, Germany, LNCS 845, pp. 285–301. Springer.

L. Bachmair and H. Ganzinger, 1994b. Ordered chaining for total orderings. *Proc. 12th Int. Conf. on Automated Deduction*, Nancy, France, LNCS 814, pp. 435–450. Springer.

L. Bachmair and H. Ganzinger, 1994c. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation* **4**(3): 217–247. Revised version of Technical Report MPI-I-93-250, Max-Planck-Institut für Informatik, Saarbrücken, 1993.

L. Bachmair and D. Plaisted, 1985. Termination orderings for associative-commutative rewriting systems. *Journal of Symbolic Computation* **1**: 329–349.

L. Bachmair, H. Ganzinger and U. Waldmann, 1994. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing* **5**(3/4): 193–212. Earlier version: Theorem Proving for Hierarchic First-Order Theories. *Proc. 3rd Int. Conf. on Algebraic and Logic Programming*, Volterra, Italy, LNCS 632. Springer, 1992.

L. Bachmair, 1991. *Canonical Equational Proofs*. Birkhäuser, Boston.

R. S. Boyer and J. S. Moore, 1988. Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic. In J. E. Hayes, D. Michie and J. Richards (eds), *Machine Intelligence 11*, chapter 5, pp. 83–124. Clarendon Press, Oxford.

B. Buchberger and R. Loos, 1983. Algebraic simplification. *Computer Algebra: Symbolic and Algebraic Computation*, 2nd edn, pp. 11–43. Springer.

B. Buchberger, 1965. *An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Ideal*, PhD thesis, University of Innsbruck, Austria. In german.

B. Buchberger, 1985. Gröbner bases: An algorithmic method in polynomial ideal theory. In N. K. Bose (ed.), *Recent Trends in Multidimensional Systems Theory*, chapter 6, pp. 184–232. D. Reidel Publishing Company.

B. Buchberger, 1987. History and basic features of the critical pair/completion procedure. *Journal of Symbolic Computation* **3**: 3–38.

H.-J. Bürckert, 1990. A resolution principle for clauses with constraints. *Proc. 10th Int. Conf. on Automated Deduction*, Kaiserslautern, Germany, LNCS 449, pp. 178–192. Springer.

N. Dershowitz and J.-P. Jouannaud, 1990. Rewrite systems. In J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics*, chapter 6, pp. 243–320. Elsevier/MIT Press.

A. KANDRI-RODY AND D. KAPUR, 1988. Computing a Gröbner basis of a polynomial ideal over a euclidean domain. *Journal of Symbolic Computation* **6**: 19–36.
C. KIRCHNER, H. KIRCHNER AND M. RUSINOWITCH, 1990. Deduction with symbolic constraints. *Revue Française d'Intelligence Artificielle* **4**(3): 9–52. Special issue on automatic deduction.
P. LE CHENADEC, 1984. Canonical forms in finitely presented algebras. *Proc. 7th Int. Conf. on Automated Deduction*, Napa, CA, LNCS 170, pp. 142–165. Springer. Book version published by Pitman, London, 1986.
R. LOOS, 1981. Term reduction systems and algebraic algorithms. *Proc. 5th GI Workshop on Artificial Intelligence*, Bad Honnef, Informatik Fachberichte 47, pp. 214–234. Springer.
C. MARCHÉ, 1994. Normalised rewriting and normalised completion. *Proc. 9th Ann. IEEE Symp. on Logic in Computer Science*, Paris, pp. 394–403. IEEE Computer Society Press.
G. NELSON AND D. C. OPPEN, 1979. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems* **2**(2): 245–257.
R. NIEUWENHUIS AND A. RUBIO, 1994. AC-superposition with constraints: no AC-unifiers needed. *Proc. 12th Int. Conf. on Automated Deduction*, Nancy, France, LNCS 814, pp. 545–559. Springer.
S. OWRE, J. M. RUSHBY AND N. SHANKAR, 1992. PVS: A prototype verification system. *11th Int. Conf. on Automated Deduction*, Saratoga Springs, NY, LNCS 607, pp. 748–752. Springer.
G. E. PETERSON AND M. E. STICKEL, 1981. Complete sets of reductions for some equational theories. *Journal of the ACM* **28**(2): 233–264.
L. VIGNERON, 1994. Associative-commutative deduction with constraints. *Proc. 12th Int. Conf. on Automated Deduction*, Saratoga Springs, NY, LNCS 814, pp. 530–544. Springer.
U. WERTZ, 1992. First-order theorem proving modulo equations. Technical Report MPI-I-92-216, Max-Planck-Institut für Informatik, Saarbrücken.