Associative-Commutative Superposition*

Leo Bachmair[1] and Harald Ganzinger[2]

[1] Department of Computer Science, SUNY at Stony Brook,
Stony Brook, NY 11794, U.S.A,
leo@cs.sunysb.edu

[2] Max-Planck-Institut für Informatik,
Im Stadtwald, D-66123 Saarbrücken, Germany,
hg@mpi-sb.mpg.de

Abstract. We present an associative-commutative paramodulation calculus that generalizes the associative-commutative completion procedure to first-order clauses. The calculus is parametrized by a selection function (on negative literals) and a well-founded ordering on terms. It is compatible with an abstract notion of redundancy that covers such simplification techniques as tautology deletion, subsumption, and simplification by (associative-commutative) rewriting. The proof of refutational completeness of the calculus is comparatively simple, and the techniques employed may be of independent interest.

1 Introduction

Rewrite techniques are one of the more successful approaches to equational reasoning. In theorem proving these techniques usually appear in the form of completion-like procedures, such as ordered completion (Bachmair, Dershowitz and Plaisted 1989, Hsiang and Rusinowitch 1987), associative-commutative completion (Peterson and Stickel 1981), or basic completion (Bachmair, Ganzinger, Lynch, et al. 1992, Nieuwenhuis and Rubio 1992). Traditionally completion procedures were formulated for sets of equations (unit clauses), but ordered completion has been generalized to arbitrary non-unit clauses, resulting in several variants of paramodulation called superposition (Rusinowitch 1991, Zhang 1988, Bachmair and Ganzinger 1990, Pais and Peterson 1991), and the basic strategy has actually first been developed for first-order clauses. Associativity and commutativity have been built into ordered paramodulation (Paul 1992, Rusinowitch and Vigneron 1991), a calculus (Hsiang and Rusinowitch 1991) that does not generalize completion, but includes similar rewrite techniques; and Wertz (1992) designed an associative-commutative superposition calculus. Unfortunately, the completeness proofs proposed for these calculi are technically involved and quite complicated.

The calculus described in this paper is obtained by applying the technique of extended rules (Peterson and Stickel 1981) to a superposition calculus of Bachmair and

* The research described in this paper was supported in part by the NSF under research grant INT-9314412, by the German Science Foundation (Deutsche Forschungsgemeinschaft) under grant Ga 261/4-1, 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 6028 (Construction of Computational Logics). The first author was also supported by the Alexander von Humboldt Foundation.

Ganzinger (1994). A similar calculus has been discussed by Wertz (1992), and our completeness proof, like Wertz's proof, is based on the model construction techniques we originally proposed in (Bachmair and Ganzinger 1990). The main difference with our current approach is that we use *non-equality* partial models to construct an equality model. These modifications were motivated by our recent work on rewrite techniques for transitive relations in general (Bachmair and Ganzinger 1993), and allow us to more naturally deal with associative-commutative rewriting.

Completion procedures are based on commutation properties (often called "critical pair lemmas") of the underlying rewrite relation. We believe that our approach may be of independent interest in that it provides a general method for extending such procedures from unit clauses to Horn clauses to full clauses. Clauses are interpreted as conditional rewrite rules (with positive and negative conditions); new clauses need to be inferred with suitably designed inference rules, so that this clausal rewriting relation is well-defined and satisfies the required commutation properties. Associative-commutative superposition represents a non-trivial application of this general methodology.

The next section contains basic notions and terminology of theorem proving and rewriting. The associative-commutative superposition calculus is described in Section 3, and proved complete in Section 4.

2 Preliminaries

2.1 Clauses

We consider first-order languages with equality. A *term* is an expression $f(t_1, \ldots, t_n)$ or x, where f is a function symbol of arity n, x is a variable, and $t_1, \ldots, t_n$ are terms. For simplicity, we assume that equality is the only predicate in our theory. By an *atomic formula* (or *atom*) we mean a multiset $\{s, t\}$, called an *equality* and usually written $s \approx t$, where s and t are terms.[3] A *literal* is either a multiset (of multisets) $\{\{s\}, \{t\}\}$, called a *positive* literal and also written $s \approx t$, or a multiset $\{\{s, t\}\}$, called a *negative* literal and written $s \not\approx t$ or $\neg(s \approx t)$.[4] A *clause* is a finite multiset of literals. We write a clause by listing its literals $\neg A_1, \ldots, \neg A_m, B_1, \ldots, B_n$, or 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.

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 set N is *satisfiable* if it has a model, and *unsatisfiable* otherwise. For instance, any set containing the empty clause is unsatisfiable.

[3] The symmetry of equality is thus built into the notation.

[4] This somewhat unusual definition of literals as multisets of multisets of terms is motivated by our subsequent definition of an ordering on clauses in Section 2.2.

An interpretation is called an *equality interpretation* if it satisfies the *reflexivity* axiom

$$x \approx x$$

the *transitivity* axiom

$$x \not\approx y \vee y \not\approx z \vee x \approx z$$

and all *congruence* axioms

$$x \not\approx y \vee f(\ldots, x, \ldots) \approx f(\ldots, y, \ldots)$$

where f ranges over all function symbols. (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.

We will consider the problem of checking whether a given set of clauses has an equality model, and for that purpose have to reason about Herbrand interpretations. Two concepts are useful in this context: rewrite systems and reduction orderings.

2.2 Reduction Orderings

An *ordering* is a transitive and irreflexive 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.[5] By a *rewrite ordering* we mean a transitive and irreflexive rewrite relation; by a *reduction ordering*, a transitive and well-founded rewrite relation. (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 $y \succ x$. If $\succ$ is a total (resp. well-founded) ordering, so is its multiset extension. If $\succ$ is an ordering on terms, then its multiset extension is an ordering on equations; its twofold multiset extension, an ordering on literals; and its threefold multiset extension, an ordering on clauses.

2.3 Associative-Commutative Rewriting

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. Note that if I is an equality interpretation, then the set of all pairs (s, t) and (t, s), 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 equations $s \approx t$, for which $s \sim t$, is an equality interpretation. In short, equality interpretations can be described by congruence relations. Rewrite systems can be used to reason about congruence relations.

---

[5] In our notation for terms we follow Dershowitz and Jouannaud (1990).

By a *rewrite system* we mean a binary relation on terms. 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.

We assume that some function symbols f are *associative* and *commutative*, i.e., satisfy the axioms

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

For the rest of the paper, 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 binary relation containing the pairs $(f(x, f(y, z)), f(f(x, y), z))$ and $(f(x, y), f(y, x))$. 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 there exists an AC-compatible reduction ordering $\succ$ such that $s \succ t$, for all rules $s \to t$ in R.

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 there exist terms u' and v' such that $u \to_{AC\backslash R}^* u' \leftrightarrow_{AC}^* v' \leftarrow_{AC\backslash R}^* v$. Given an AC-compatible reduction ordering, we also write $u \leftrightarrow_{AC\cup R}^{\preceq t} v$ (resp. $u \leftrightarrow_{AC\cup R}^{\prec t} v$) to indicate that there exists a sequence

$$t_0 \leftrightarrow_{AC\cup R} \cdots \leftrightarrow_{AC\cup R} t_n$$

such that $t \succeq t_i$ (resp. $t \succ t_i$), for all i with $0 \leq i \leq n$.

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*. We shall use AC-convergent rewrite systems to describe equality models of AC.

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'.

The local AC-confluence property is satisfied whenever it can be shown to hold for certain "minimal" rewrite sequences $u \leftarrow_{AC\backslash R} t \to_R v$ that are also called "critical overlaps." We only have to consider certain ground rewrite systems R where critical overlaps involve extended rules in R^e. This is summarized in the following lemma.

Lemma 1. *Let R be a ground rewrite system contained in an AC-compatible reduction ordering $\succ$ and suppose that no left-hand side of a rule in R can be rewritten modulo AC by any other rule in R or any extended rule in R^e. Furthermore, let t be a ground term, such that for all ground instances $f(u,u') \to f(u,u'')$ and $f(v,v') \to f(v,v'')$ of extended rules in R^e, where $f(u,u') \leftrightarrow^*_{AC} f(v,v')$ and $t \succeq f(u,u')$, we have $f(u,u'') \downarrow_{AC\backslash R^e} f(v,v'')$. Then $w \downarrow_{AC\backslash R^e} w'$ for all ground terms w and w' such that $w \leftrightarrow^{\preceq t}_{AC \cup AC\backslash R} w'$.*

If the same conditions are only satisfied for ground instances of extended rules with $t \succ f(u,u')$, then $w \downarrow_{AC\backslash R^e} w'$ for all ground terms w and w' such that $w \leftrightarrow^{\prec t}_{AC \cup AC\backslash R} w'$.

The lemma can be proved by standard techniques from term rewriting.

3 Associative-Commutative Superposition

We formulate our inference rules in terms of a reduction ordering. Furthermore, negative literals in a clause may be marked, in which case they are said to be *selected*. We assume that at least one literal is selected in each clause in which the maximal literal is negative. (If the maximal literal is positive, there may be no selected literals.)

Let $\succ$ be a well-founded AC-compatible reduction ordering, such that $s \leftrightarrow^*_{AC} t$ or $s \succ t$ or $t \succ s$, for all ground terms s and t. (Such orderings have been described by Narendran and Rusinowitch (1991) and Rubio and Nieuwenhuis (1993).) 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 $C\sigma \vee s\sigma \approx t\sigma$ such that $s\sigma \succ t\sigma$ and $s\sigma \approx t\sigma \succ C\sigma$. By an *extended clause* we mean a reductive clause $C \vee f(x,s) \approx f(x,t)$, where $f \in AC$, s is a term $f(u,v)$, and x is a variable not occurring in C, s, or t. We also say that $C \vee f(x,s) \approx f(x,t)$ is an *extension* of $C \vee s \approx t$. We will see that only extensions of certain reductive clauses without selected literals are needed. (Such extensions are themselves reductive.)

Associativity and commutativity are built into the inference rules via AC-unification. Two terms u and v are *AC-unifiable* if $u\sigma \leftrightarrow^*_{AC} v\sigma$, 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 $u\sigma \leftrightarrow^*_{AC} v\sigma$ there exist substitutions $\tau \in CSU_{AC}(u,v)$ and ρ, such that $x\sigma \leftrightarrow^*_{AC} (x\tau)\rho$, for all variables x in u and v. We assume that a function CSU_{AC} is given and call a substitution in $CSU_{AC}(u,v)$ a *most general AC-unifier* of u and v.

3.1 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.)

$$AC\text{-}\mathbf{Superposition}: \qquad \frac{C,\, s \approx t \quad D,\, u[s'] \approx v}{C\sigma,\, D\sigma,\, u[t]\sigma \approx v\sigma}$$

where (i) σ is a most general AC-unifier of s and s', (ii) the clause $C\sigma \vee s\sigma \approx t\sigma$ is reductive for $s\sigma \approx t\sigma$ and $C \vee s \approx t$ either contains no selected literals or is an extended clause,[6] (iii) the clause $D\sigma \vee u\sigma \approx v\sigma$ is reductive for $u\sigma \approx v\sigma$ and either contains no selected literals or is an extended clause, and (iv) s' is not a variable.

$$\mathbf{Negative}\ AC\text{-}\mathbf{Superposition}: \qquad \frac{C,\, s \approx t \quad D,\, u[s'] \not\approx v}{C\sigma,\, D\sigma,\, u[t]\sigma \not\approx v\sigma}$$

where (i) σ is a most general AC-unifier of s and s', (ii) the clause $C\sigma \vee s\sigma \approx t\sigma$ is reductive for $s\sigma \approx t\sigma$ and $C \vee s \approx t$ either contains no selected literals or is an extended clause, (iii) the literal $u \not\approx v$ is selected in $D \vee u \not\approx v$ and $v\sigma \not\succeq u\sigma$, and (iv) s' is not a variable.

$$\mathbf{Reflective}\ AC\text{-}\mathbf{Resolution}: \qquad \frac{C,\, u \not\approx v}{C\sigma}$$

where σ is a most general AC-unifier of u and v and $u \not\approx v$ is a selected literal in the premise.

$$AC\text{-}\mathbf{Factoring}: \qquad \frac{C,\, s \approx t,\, s' \approx t'}{C\sigma,\, t\sigma \not\approx t'\sigma,\, s'\sigma \approx t'\sigma}$$

where σ is a most general AC-unifier of s and s', $t'\sigma \not\succ t\sigma$, the literal $s\sigma \approx t\sigma$ is maximal in $C\sigma$, and the premise contains no selected literals.

3.2 Lifting Properties

In proving the refutational completeness of the superposition calculus $\mathsf{S}^{\succ}_{AC}$, we shall have to argue about ground instances of clauses and inferences. The connection between the general level and the ground level is usually stated in the form of so-called "lifting" properties.

Let $C_1 \ldots C_n$ be clauses and let

$$\frac{C_1\sigma \ldots C_n\sigma}{D}$$

be a ground inference, i.e., all clauses $C_i\sigma$ and D are ground. (We assume that each clause $C_i\sigma$ is marked in the same way as C_i; that is, $L\sigma$ is selected in $C\sigma$ if and only

[6] In other words, selection is ignored for extended clauses.

if L is selected in C.) We say that this ground inference can be *AC-lifted* if there is an inference

$$\frac{C_1 \ldots C_n}{C}$$

such that $C\sigma \leftrightarrow^*_{AC} D$. In that case, we also say that the ground inference is an *AC-instance* of the general inference.

Lifting is no problem for resolution and factoring inferences, but is more difficult for superposition inferences. For example, if N contains two clauses $f(x,x) \approx x$ and $a \approx b$, where $a \succ b$, then there is a superposition inference from ground inferences

$$\frac{a \approx b \quad f(a,a) \approx a}{f(a,b) \approx a}$$

but no superposition inference from the clauses themselves. To prove completeness, one has to show that all necessary inferences can be AC-lifted. Let us illustrate the conditions under which AC-superposition inferences can be lifted. (The case of negative AC-superposition is similar.)

Let $C \vee s \approx t$ and $D \vee u[s'] \approx v$ be two clauses, each either containing no selected literals or being an extended clause. An AC-superposition inference

$$\frac{C\tau,\ s\tau \approx t\tau \quad D\tau,\ u[s']\tau \approx v\tau}{C\tau,\ D\tau,\ u[t]\tau \approx v\tau}$$

from ground instances of these clauses, where $s'\tau \leftrightarrow^*_{AC} s\tau$, can be AC-lifted if s' is a non-variable subterm of u. (*Proof.* The ordering restrictions require that $s\tau \approx t\tau$ is strictly maximal in $C\tau$ and $s\tau \succ t\tau$, and $u\tau \approx v\tau$ is strictly maximal in $D\tau$ and $u\tau \succ v\tau$. Consider the inference

$$\frac{C,\ s \approx t \quad D,\ u[s'] \approx v}{C\sigma,\ D\sigma,\ u[t']\sigma \approx v\sigma}$$

where σ is a most general AC-unifier of s and s', such that for some substitution ρ we have $x\tau \leftrightarrow^*_{AC} (x\sigma)\rho$, for all variables x occurring in s' or s, and $y\tau = (y\sigma)\rho$, for all other variables y. It can easily be shown that $C\sigma \vee s\sigma \approx t\sigma$ is reductive for $s\sigma \approx t\sigma$ and $D\sigma \vee u\sigma \approx v\sigma$ is reductive for $u\sigma \approx v\sigma$. Also, s' is not a variable, so that the inference is an AC-superposition inference. Moreover, the conclusion of the ground inference is AC-equivalent to an instance of the conclusion of the general inference.)

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 1994), which we shall now adapt to the AC-case.

Let RA denote the set consisting of the reflexivity axiom, F the set of all congruence axioms, and T the set consisting of the transitivity axiom, cf. Section 2. Furthermore, for any ground term s, let $T_{\preceq s}$ be the set of all ground instances

$u \not\approx v \vee v \not\approx w \vee u \approx w$ of the transitivity axiom, for which $s \succeq u$ and $s \succeq v$ and $s \succeq w$; and $T_{\prec s}$ be the set of all ground instances for which $s \succ u$ and $s \succ v$ and $s \succ w$;

Let N be a set of clauses. 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 with $1 \leq j \leq k$, and C is true in every model of $AC \cup RA \cup F \cup T_{\preceq s} \cup \{C_1, \ldots, C_k\}$, where s is the maximal term in C. It can easily be seen that the clauses $C_1, \ldots, C_k$ can be assumed to be non-redundant. A non-ground clause is called AC-redundant if all its ground instances are.

Remark. 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 usual simplification techniques can be formalized using AC-redundancy. Wertz (1992) uses a different notion of redundancy, where restrictions are imposed, not on transitivity, but instead on the use of the associativity and commutativity axioms.

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 $v\sigma$.

A ground inference with conclusion B is called *AC-redundant* with respect to N if there exist ground instances $C_1, \ldots, C_k$ of N, such that B is true in every model of $AC \cup RA \cup F \cup T_{\prec s} \cup \{C_1, \ldots, C_k\}$, where s is the maximal term in C, and such that the C_j are "sufficiently small". In the case of an inference in which both premises are ground instances of extended clauses, sufficiently small means that $\{\{s\}\} \succ C_j$.[7] For any other ground inference it means that $C \succ C_j$, where C is the maximal premise of the inference. 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.

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.

Let us conclude this section by pointing out that extended clauses are redundant, but inferences with them are not. Therefore it is possible to dispense with extended rules altogether, and instead encode inferences with them directly in an "extended" AC-superposition calculus, cf., Rusinowitch and Vigneron (1991).

4 Refutational Completeness

In this section we will show how to define an equality model for saturated clause sets that do not contain the empty clause. The definition of a suitable model uses

[7] $\{\{s\}\} \succ C_j$ is true if and only if all terms in C_j are smaller than s.

induction on $\succ$ and is adapted from Bachmair and Ganzinger (1994). The presence of extended rules causes technical complications, though, and requires certain modifications in the model construction process.

Given a set N of ground clauses, we define a corresponding Herbrand interpretation I using induction on $\succ$. For every clause C in N we define R_C to be the set $\bigcup_{C \succ D} E_D$; and denote by I_C the set $\{u \approx v : u \text{ and } v \text{ ground}, u \downarrow_{AC \setminus R_C^e} v\}$, which we also call the *AC-rewrite closure* of R^e. Furthermore, if C is a clause $C' \vee s \approx t$, where $(s \approx t) \succ C'$ and $s \succ t$, then $E_C = \{s \approx t\}$ if (i) C is false in I_C, (ii) C' is false in the AC-rewrite closure of $(R_C \cup \{s \approx t\})^e$, and (iii) the term s is AC-irreducible by R_C^e. In that case, we also say that C is *productive* and that it *produces* $s \approx t$. In all other cases, $E_C = \emptyset$. Finally, let R be $\bigcup_C E_C$ and let I be the AC-rewrite closure of R^e.

The Herbrand interpretation I is intended to be an equality model of $N \cup AC$, provided N is saturated and does not contain the empty clause. The following lemmas state the essential properties of the interpretations I and I_C.

Lemma 2. *Let C be a ground clause (which need not be in N) and D be a clause in N with $D \succeq C$. If $\neg A$ is a negative literal in C with $A \notin I_D$, then $A \notin I$. As a consequence, if C is true in I_D, then it is also true in I and in any interpretation $I_{D'}$ with $D' \in N$ and $D' \succ D$.*

Proof. Let $\neg A$ be a negative literal in C such that $A \notin I_D$. If $B \in I \setminus I_D$, then $B \succ \neg A$, and therefore $A \notin I$. Also, if A is a positive atom in C and $A \in I_D$, then A is contained in any set $I_{D'}$ with $D' \succ D$ and in I. From this we may conclude that if C is true in I_D, then it is also true in I and in any interpretation $I_{D'}$ with $D' \in N$ and $D' \succ D$.

The next lemma follows immediately from the definition of the interpretations I_C and I.

Lemma 3. *The interpretation I and all interpretations I_C are models of the associativity, commutativity, reflexivity, symmetry, and all congruence axioms.*

We emphasize that transitivity need not be satisfied by all interpretations I_C or I, and consequently these interpretations need not be equality interpretations. However, if the clause set N is saturated, then "sufficiently many" instances of the transitivity axiom are true in each interpretation I_C, so that the final interpretation I does indeed satisfy transitivity, and hence is an equality interpretation. The following lemma is essential in this regard.

Lemma 4. *The interpretation I_C is a model of $T_{\preceq t}$ (resp. $T_{\prec t}$) if and only if for all ground terms u and v with $u \leftrightarrow^{\preceq t}_{AC \cup R_C^e} v$ (resp. $u \leftrightarrow^{\prec t}_{AC \cup R_C^e} v$) we have $u \downarrow_{AC \setminus R_C^e} v$.*

Proof. First let us assume that we have $u \downarrow_{AC \setminus R_C^e} v$ for all ground terms u and v with $u \leftrightarrow^{\preceq t}_{AC \cup R_C^e} v$. We show that I_C is a model of $T_{\preceq t}$. Let u, v, and w be ground terms with $t \succeq u$, $t \succeq v$, and $t \succeq w$. If $u \approx v$ and $v \approx w$ are true in I_C, then $u \downarrow_{AC \setminus R_C^e} v$ and $v \downarrow_{AC \setminus R_C^e} w$. In other words, we have $u \leftrightarrow^{\preceq t}_{AC \cup R_C^e} w$; hence $u \downarrow_{AC \setminus R_C^e} w$ and $u \approx w$ is true in I_C.

On the other hand, if I_C is a model of $T_{\preceq t}$, then we have $u \downarrow_{AC\setminus R^e_C} w$ whenever $u \leftarrow_{AC\setminus R^e_C} v \rightarrow_{AC\setminus R^e_C} w$, for some term v with $t \succeq v$. Using Lemma 1, we may conclude that $u' \downarrow_{AC\setminus R^e_C} v'$ for all ground terms u' and v' with $u' \leftrightarrow^{\preceq t}_{AC\cup R^e_C} v'$.

We now come to the main properties of the model construction.

Lemma 5. *Let N be a set of clauses that is saturated up to AC-redundancy, does not contain the empty clause, but contains an extension of every non-redundant reductive clause in N. Let I be the interpretation constructed from the set of all ground instances of $N \cup RA$. Then for every ground instance C of a clause in $N \cup RA$ we have:*

(1) If s is the maximal term in C, then I_C is a model of $T_{\preceq s}$.

(2) If C is an instance $\hat{C}\sigma$ of a clause in N, such that $x\sigma$ is AC-reducible by R^e_C, for some variable x in $\hat{C}$, then C is true in I_C.

(3) If C is an instance of a clause with a selected literal, then it is true in I_C.

(4) If C is non-productive, then it is true in I_C.

(5) If $C = C' \vee A$ produces A, then C is a non-redundant instance of a clause in N with no selected literals and C' is false in I.

Proof. We use induction on $\succ$. Let C be a ground instance of $N \cup RA$ with maximal term s and assume (1)-(7) are already satisfied for all ground instances C' of N with $C \succ C'$.

(1) Since for every ground term t there is at least one clause with maximal term t (namely the ground instance $t \approx t$ of the reflexivity axiom), we may use the induction hypothesis to infer that I_C is a model of $T_{\prec s}$ By Lemmas 1 and 4, it is therefore sufficient to prove that for any two ground instances $f(u, u') \rightarrow f(u, u'')$ and $f(v, v') \rightarrow f(v, v'')$ of extended rules in R^e_C, with $s \leftrightarrow^*_{AC} f(u, u') \leftrightarrow^*_{AC} f(v, v')$, we have $f(u, u'') \downarrow_{AC\setminus R^e_C} f(v, v'')$.

Suppose $u' \approx u''$ is produced by $C' \vee u' \approx u''$ and $v' \approx v''$ is produced by $D' \vee v' \approx v''$. Both clauses are strictly smaller than C, so that by the induction hypothesis, they are true in I_C, whereas C' and D' are false in I. By AC-superposition we obtain the clause $C''' = C' \vee D' \vee f(u, u'') \approx f(v, v'')$. Since this inference can be AC-lifted, we may use saturation up to AC-redundancy, to infer that there exist ground instances $C_1, \ldots, C_k$ of N, such that $\{\{s\}\} \succ C_j$, for all j with $1 \le j \le k$, and C'' is true in every model of $AC \cup RA \cup F \cup T_{\prec s} \cup \{C_1, \ldots, C_k\}$. By induction hypothesis, I_C is a model of $AC \cup RA \cup F \cup T_{\prec s} \cup \{C_1, \ldots, C_k\}$, hence C'' is true in I_C. This implies that $f(u, u'') \approx f(v, v'')$ is true in I_C, and thus $f(u, u'') \downarrow_{AC\setminus R^e_C} f(v, v'')$.

(2) Suppose $C = \hat{C}\sigma$ is a ground instance of N, such that $x\sigma$ is AC-reducible by R^e_C, say $x\sigma \rightarrow_{AC\setminus R^e_C} u$. Let σ' be the same substitution as σ, except that $x\sigma' = u$, and let C' be the clause $\hat{C}\sigma'$. Since $C \succ C'$, we may use the induction hypothesis and Lemma 2 to infer that C' is true in I_C. Furthermore, since $x\sigma \approx u$ is true in I_C, a literal $L\sigma$ in C is true in I_C if and only if the corresponding literal $L\sigma'$ in C' is true in I_C. This implies that C is true in I_C.

If C is an instance $\hat{C}\sigma$, where $x\sigma$ is irreducible for all variables x in $\hat{C}$, then any inference with maximal premise C can be AC-lifted. We call C an *AC-reduced ground instance* of N in that case.

(3) Let $C = C' \vee s \not\approx t$ be an AC-reduced ground instance of a clause in which the literal corresponding to $s \not\approx t$ is selected. The assertion is obviously true if $s \approx t$ is false in I_C. So let us assume $s \downarrow_{AC \backslash R_C^e} t$.

(3.1) If $s \leftrightarrow_{AC}^* t$, then C' is a reflective AC-resolvent of C. Resolution inferences can be lifted, so that we may use saturation up to AC-redundancy, to infer that C' is true in I_C.

(3.2) If s and t are not AC-equivalent, then we have either $s \succ t$ or $t \succ s$. We discuss only one case, as the other is similar. Suppose $s \succ t$, in which case s is AC-reducible by R_C^e and can be written as $s[u']$, where $u' \leftrightarrow_{AC}^* u$, for some clause $D = D' \vee u \approx v$ with maximal literal $u \approx v$, and $s[v] \downarrow_{AC \backslash R_C^e} t$. The clause D is either productive or else is a ground instance of an extension of a productive clause. Since $C \succ D$, we use the induction hypothesis to infer that D' is false in I. Also, if D is productive, it is non-redundant and hence must be a ground instance of a clause with no selected literals. The clause $C'' = C' \vee D' \vee s[v] \not\approx t$ can be obtained from C and D by negative AC-superposition. By saturation up to redundancy, C'' must be true in I_C. Since D' and $s[v] \approx t$ are false in I_C, we may infer that C', and thus C, is true in I_C.

(4) Suppose C is false in I_C but non-productive. That is, C violates one of the other two conditions imposed on productive clauses.

(4.1) If condition (ii) is violated, then C is of the form $C' \vee s \approx t \vee s' \approx t'$, where $s \approx t$ is a maximal literal, $s \succ t \succ t'$, $s \leftrightarrow_{AC}^* s'$, and $t \downarrow_{AC \backslash R_C^e} t'$. We obtain the clause $C'' = C' \vee t \not\approx t' \vee s' \approx t'$ from C by AC-factoring. Using saturation up to redundancy, we may infer that C'' is true in I_C. Since $t \approx t'$ is true in I_C, this implies that $C' \vee s' \approx t'$, and therefore C, is true in I_C, which is a contradiction.

(4.2) Otherwise, C must be a clause $C' \vee s \approx t$ with maximal literal $s \approx t$, where $s \succ t$ and s is AC-reducible by R_C^e. Then there exists a clause $D = D' \vee u \approx v$ with maximal literal $u \approx v$, such that s can be written as $s[u']$ and $u' \leftrightarrow_{AC}^* u$. Moreover, D is either productive or is a ground instance of an extension of a productive clause. By the induction hypothesis, the clause D' is false in I and if D is productive it is a non-redundant instance of a clause with no selected literals. Thus, we may obtain from C and D by AC-superposition the clause $C'' = C' \vee D' \vee s[v] \approx t$, which by saturation has to be true in I_C. Since D' is false in I_C, either C' or $s[v] \approx t$ must be true in I_C. If C' is true in I_C, so is C. If $s[v] \downarrow_{AC \backslash R_C^e} t$, then we have $s = s[u'] \rightarrow_{AC \backslash R_C^e} s[v] \downarrow_{AC \backslash R_C^e} t$, which indicates that $s \approx t$, and hence C, is true in I_C—a contradiction.

(5) A productive clause $C = C' \vee A$ is false in I_C, hence is non-redundant and, by part (3), cannot be an instance of a clause with selected literals. The definition of productive clauses also ensures that C' is false in the AC-rewrite closure of $(R_C \cup E_C)^e$, and hence false in I.

The lemma indicates that under the given assumptions I is an equality model of $AC \cup N$. (Any non-productive ground instance C of N is true in I_C, while non-productive ground instances C are true in the AC-rewrite closure of $(R_C \cup E_C)^e$. Furthermore, I is a model of $T_{\preceq t}$, for all ground terms t; and hence is a model of T.)

The proof of the lemma also shows that certain inferences with extended clauses are unnecessary. For instance, AC-superpositions on a proper subterm of an extended clause, i.e., inferences of the form

$$\frac{C,\ s \approx t \quad D,\ f(x, u[s']) \approx f(x, v)}{C\sigma,\ D\sigma,\ f(x, u[t])\sigma \approx f(x, v)\sigma}$$

where σ is a most general AC-unifier of s and s', and the second premise extends the clause $D \vee u[s'] \approx v$, are not needed.

As an immediate corollary of the above lemmas we obtain:

Theorem 6. *Let N be a set of clauses that is saturated up to AC-redundancy with respect to the associative-commutative superposition calculus $\mathsf{S}^{\succ}_{AC}$. Furthermore, suppose N contains an extension of every non-redundant reductive clause in N. Then $N \cup AC$ is equality unsatisfiable if and only if it contains the empty clause.*

Proof. If $N \cup AC$ contains the empty clause it is unsatisfiable. If $N \cup AC$ does not contain the empty clause, let I be the interpretation constructed from the set of all ground instances of $N \cup RA$. By Lemmas 3 and 5, I is an equality model of $N \cup AC$.

This theorem shows that associative-commutative superposition calculi provide a basis for refutationally complete theorem provers. Such a theorem prover has to saturate a given set of input clauses up to AC-redundancy. The empty clause will be generated if the input set, plus AC, is equality unsatisfiable. Saturation of clause sets can be achieved by *fair* application of inference rules. These aspects of the saturation process have been discussed elsewhere, e.g., Bachmair and Ganzinger (1994) or Bachmair, Ganzinger and Waldmann (1994), and apply directly to the present case.

5 Summary

We have presented an associative-commutative superposition calculus and proved its refutational completeness. We have tried to keep the exposition clear and simple, and therefore have not discussed various possible improvements to the calculus, most of which can be derived from redundancy. Some are relatively minor, such as the redundancy of certain inferences involving an extended clause, while others are more important, e.g., the use of redex orderings to achieve a similar effect as critical pair criteria. The main difference of our calculus with other associative-commutative paramodulation calculi is that the associativity and commutativity axioms, but not all instances of the transitivity axiom, are built into our notion of redundancy; whereas other researchers have opted for transitivity and compromised on associativity and commutativity (Wertz 1992, Nieuwenhuis and Rubio 1994). From a more practical perspective our approach is preferrable, as enough instances of transitivity are provided to cover full associative-commutative rewriting, while otherwise associative-commutative rewriting actually needs to be restricted. In addition, our completeness proof is simpler than previous proofs.

We believe that the approach we have outlined for associativity and commutativity can be applied to other equational theories, such associativity and commutativity with identity (Baird, Peterson and Wilkerson 1989, Jouannaud and

Marché 1992); for some work in this direction see Wertz (1992). Recently, associative-commutative calculi have been combined with the basic strategy, or more generally, with constraint-based deduction, and completeness results have been proved by Nieuwenhuis and Rubio (1994) and by Vigneron (1994).

References

L. Bachmair, N. Dershowitz and D. Plaisted, 1989. Completion without failure. In H. Ait-Kaci, M. Nivat, editors, *Resolution of Equations in Algebraic Structures, vol. 2*, pp. 1–30. Academic Press.

L. Bachmair and H. Ganzinger, 1990. On Restrictions of Ordered Paramodulation with Simplification. In M. Stickel, editor, *Proc. 10th Int. Conf. on Automated Deduction, Kaiserslautern*, Lecture Notes in Computer Science, vol. 449, pp. 427–441, Berlin, Springer-Verlag.

L. Bachmair and H. Ganzinger, 1993. Rewrite Techniques for Transitive Relations. Technical Report MPI-I-93-249, Max-Planck-Institut für Informatik, Saarbrücken. Short version in Proc. LICS'94, pp. 384–393, 1994.

L. Bachmair and H. Ganzinger, 1994. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, Vol. 4, No. 3, pp. 217–247. Revised version of Technical Report MPI-I-91-208, 1991.

L. Bachmair, H. Ganzinger, Chr. Lynch and W. Snyder, 1992. Basic Paramodulation and Superposition. In D. Kapur, editor, *Automated Deduction — CADE'11*, Lecture Notes in Computer Science, vol. 607, pp. 462–476, Berlin, Springer-Verlag.

Leo Bachmair, Harald Ganzinger and Uwe Waldmann, 1994. Refutational Theorem Proving for Hierarchic First-Order Theories. *Applicable Algebra in Engineering, Communication and Computing*, Vol. 5, No. 3/4, pp. 193–212. Earlier version: Theorem Proving for Hierarchic First-Order Theories, in Giorgio Levi and Hélène Kirchner, editors, *Algebraic and Logic Programming, Third International Conference*, LNCS 632, pages 420–434, Volterra, Italy, September 2–4, 1992, Springer-Verlag.

Timothy Baird, Gerald Peterson and Ralph Wilkerson, 1989. Complete sets of reductions modulo associativity, commutativity and identity. In *Proc. 3rd Int. Conf. on Rewriting Techniques and Applications*, Lecture Notes in Computer Science, vol. 355, pp. 29–44, Berlin, Springer-Verlag.

N. Dershowitz and J.-P. Jouannaud, 1990. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science B: Formal Methods and Semantics*, chapter 6, pp. 243–309. North-Holland, Amsterdam.

J. Hsiang and M. Rusinowitch, 1991. Proving refutational completeness of theorem proving strategies: The transfinite semantic Tree method. *Journal of the ACM*, Vol. 38, No. 3, pp. 559–587.

Jieh Hsiang and Michael Rusinowitch, 1987. On Word Problems in Equational Theories. In *Proc. 14th ICALP*, Lecture Notes in Computer Science, vol. 267, pp. 54–71, Berlin, Springer-Verlag.

Jean-Pierre Jouannaud and Claude Marché, 1992. Termination and completion modulo associativity, commutativity and identity. *Theoretical Computer Science*, Vol. 104, pp. 29–51.

Paliath Narendran and Michaël Rusinowitch, 1991. Any Ground Associative-Commutative Theory has a Finite Canonical System. In Ronald V. Book, editor, *Proc. 4th Rewriting Techniques and Applications 91*, Como, Italy, Springer-Verlag.

R. Nieuwenhuis and A. Rubio, 1992. Basic superposition is complete. In *ESOP'92*, Lecture Notes in Computer Science, vol. 582, pp. 371–389, Berlin, Springer-Verlag.

R. Nieuwenhuis and A. Rubio, 1994. AC-superposition with constraints: No AC-unifiers needed. In *Proc. 12th International Conference on Automated Deduction*, Lecture Notes in Computer Science, vol. 814, pp. 545–559, Berlin, Springer-Verlag.

John Pais and G.E. Peterson, 1991. Using Forcing to Prove Completeness of Resolution and Paramodulation. *Journal of Symbolic Computation*, Vol. 11, pp. 3–19.

E. Paul, 1992. A general refutational completeness result for an inference procedure based on associative-commutative unification. *Journal of Symbolic Computation*, Vol. 14, pp. 577–618.

G. Peterson and M. Stickel, 1981. Complete sets of reductions for some equational theories. *Journal of the ACM*, Vol. 28, pp. 233–264.

A. Rubio and R. Nieuwenhuis, 1993. A precedence-based total AC-compatible ordering. In *Proc. 5th Int. Conf. on Rewriting Techniques and Applications*, Lecture Notes in Computer Science, vol. 690, pp. 374–388, Berlin, Springer-Verlag.

M. Rusinowitch, 1991. Theorem proving with resolution and superposition: An extension of the Knuth and Bendix completion procedure as a complete set of inference rules. *J. Symbolic Computation*, Vol. 11, pp. 21–49.

M. Rusinowitch and L. Vigneron, 1991. Automated deduction with associative-commutative operators. In *Proc. Int. Workshop on Fundamentals of Artificial Intelligence Research*, Lecture Notes in Artificial Intelligence, vol. 535, pp. 185–199, Berlin, Springer-Verlag.

L. Vigneron, 1994. Associative-commutative dedution with constraints. In *Proc. 12th International Conference on Automated Deduction*, Lecture Notes in Computer Science, vol. 814, pp. 530–544, Berlin, Springer-Verlag.

U. Wertz, 1992. First-Order Theorem Proving Modulo Equations. Technical Report MPI-I-92-216, Max-Planck-Institut für Informatik, Saarbrücken.

H. Zhang, 1988. *Reduction, superposition and induction: Automated reasoning in an equational logic.* PhD thesis, Rensselaer Polytechnic Institute, Schenectady, New York.