LEO BACHMAIR AND HARALD GANZINGER

EQUATIONAL REASONING IN SATURATION-BASED THEOREM PROVING

KEY WORDS: equational reasoning, term rewriting, paramodulation, redundancy elimination

1. Introduction

Equational reasoning is fundamental in mathematics, logics, and many applications of formal methods in computer science. In this chapter we describe the theoretical concepts and results that form the basis of state-of-the-art automated theorem provers for first-order clause logic with equality. We mainly concentrate on refinements of paramodulation, such as the superposition calculus, that have yielded the most promising results to date in automated equational reasoning.

We begin with some preliminary material in section 2 and then explain, in section 3, why resolution with the congruence axioms is an impractical theorem proving method for equational logic. In section 4 we outline the main results about paramodulation—a more direct equational inference rule. This section also contains a description of the modification method, which can be used to demonstrate that the functional reflexivity axioms are redundant in the context of paramodulation. The modification method is still significant for its application in non-local theorem proving methods based on semantic tableaux or model elimination. In the later sections, we will concentrate on refinements of paramodulation, e.g., by rewrite techniques. In sections 5 and 6 we discuss completion and superposition and explain corresponding proof techniques for establishing refutational completeness: proof normalization and reduction of counterexamples for candidate models. In section 7 we present the basic theoretical results on superposition with subterm selection strategies (that are implicit in constrained deduction formalisms). We conclude with a summary and an outline of methods for superposition over built-in theories.

The highly optimized equational theorem proving methods outlined in this chapter today outperform the more traditional resolution-based or tableau-based methods. The recent automated proof of the Robbins conjecture by Mc-

Cune (1997) provides strong experimental evidence in this regard. This success required clever prover engineering and heuristics, but as we intend to demonstrate below, depended to an even larger degree on the insights provided by a long line of theoretical research on paramodulation-based calculi and related deductive methods.

2. Preliminaries

We assume the usual notions and notations about equational clause logic. *Clauses* are disjunctions of literals. The symbols $\vee$ and $\neg$ denote disjunction and negation, respectively. The *empty clause* is denoted by the symbol $\perp$. Since disjunction is associative and commutative, clauses may be viewed as multisets of literals. Clauses with at most one positive literal are called *Horn clauses*. Horn clauses of the form $\neg A_1 \vee \ldots \vee \neg A_k \vee B$ will also be written as sequents $A_1, \ldots, A_k \rightarrow B$. Satisfiability and logical consequence (denoted by the symbol $\models$) are defined in the usual way. For simplicity, we assume that equality (denoted by the symbol $\approx$) is the only predicate. We speak of an *equality interpretation* if $\approx$ is interpreted as a congruence relation. We say that a set of clauses N is (generally) *satisfiable* if it has a model; and *equationally satisfiable* if it has a model that is an equality interpretation. (The distinction will only matter when we discuss the relation between non-equational calculi, such as resolution, and equational calculi, such as paramodulation.) *Equality Herbrand interpretations* are essentially congruences on ground terms over the given signature.

Substitutions will be denoted by the letters σ and τ. The result of applying a substitution σ to an expression (e.g., a clause or term) E is denoted $E\sigma$. We write $E[s]$ to indicate that s is a subterm of E at some position and (ambiguously) write $E[t]$ for the result of replacing s by t at the indicated occurrence.

3. Resolution with Congruence Axioms

Saturation-based theorem proving in its modern form was invented by Robinson (1965b) when he introduced the resolution calculus:

(Binary) Resolution

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma}$$

where σ is the most general unifier of the atoms A and B.

(Positive) Factoring

$$\frac{C \vee A \vee B}{(C \vee A)\sigma}$$

where σ is the most general unifier of the atoms A and B.

Resolution is a refutationally complete theorem proving method: a contradiction (i.e., the empty clause) can be deduced from any unsatisfiable set of clauses. The search for a contradiction proceeds by saturating the given clause set, that is, systematically (and exhaustively) applying all inference rules.

Resolution on ground clauses is a version of the cut rule restricted to atomic formulas, whereas factoring is an instance of weakening.[1] In fact, the refutational completeness of resolution for propositional logic can be derived from the completeness of the sequent calculus; and Herbrand's theorem, which states that for any unsatisfiable set of non-ground clauses there is a finite set of ground instances that is propositionally unsatisfiable, establishes a link between propositional clauses and general clauses with variables. A key in the "lifting" argument is the existence (and uniqueness) of a most general unifier for any two unifiable atoms or terms.

But resolution is not primarily a method for deciding the unsatisfiability of propositional formulas (the procedure by Davis and Putnam (1960) is better suited for that purpose). Its main advantage over other early theorem proving methods, such as Gilmore's algorithm (1960), is that unification, as a selection mechanism for inferences, provides an effective way of interleaving the two processes: (i) the identification of suitable (ground) instances of clauses and (ii) a demonstration of their unsatisfiability.

The obvious extension of resolution to first-order logic with equality consists of explicitly adding *congruence axioms* that express the properties of equality. For instance, the set E of axioms

$$\begin{array}{rcl} & & x \approx x \\ x \approx y & \to & y \approx x \\ x \approx y\,,\, y \approx z & \to & x \approx z \end{array}$$

expresses that equality is reflexive, symmetric, and transitive, i.e., an equivalence relation. In addition, one needs functional substitutivity (or monotonicity) axioms of the form

$$x \approx y \quad \to \quad f(\ldots,x,\ldots) \approx f(\ldots,y,\ldots)$$

[1] Some textbooks (e.g., Gallier, 1986) take a different perspective and present resolution as a macro inference in the cut-free sequent calculus. However, when the clauses to be refuted are viewed as additional non-logical axioms, cuts can not be eliminated, but may be restricted to analytic cuts—the resolution inferences.

for all function symbols f in the given signature. A set of clauses N is *equationally* satisfiable if and only if the set consisting of N plus all the above axioms is satisfiable. Resolution can thus be used to determine whether a clause set is equationally unsatisfiable, but is extremely prolific when applied to the congruence axioms. For instance, there are infinitely many resolution inferences from just the transitivity and monotonicity axioms.

Various improvements have been proposed. The monotonicity axioms are not needed for Skolem functions, as they may be added to the original formulas, before any clause transformations (including skolemization) are applied. The symmetry and transitivity axioms may be replaced by a single *commutation axiom*

$$x \approx y\,,\, x \approx z \rightarrow y \approx z$$

as proposed by Brand (1975). We denote by $\mathcal{C}$ the set of the reflexivity, commutation, and all monotonicity axioms.

Other refinements involve *selection functions* that mark in every clause a possibly empty subset of (occurrences of) negative literals (the *selected* literals). The following multi-premise variant of resolution focusses on selected literals:

Resolution with selection

$$\frac{C_1 \vee A_1 \quad \ldots \quad C_n \vee A_n \quad D \vee \neg B_1 \vee \ldots \vee \neg B_n}{(C_1 \vee \ldots \vee C_n \vee D)\sigma}$$

where $n \geq 1$ and σ is the most general substitution for which $A_i\sigma = B_i\sigma$, for all i, such that (i) none of the *positive premises* $C_i \vee A_i$ contains a selected literal, and (ii) either the indicated literals $\neg B_i$ are exactly the selected literals in the last (or *negative*) premise, or else $n = 1$ and the negative premise contains no selected literal at all.

Hyper-resolution (Robinson, 1965a) is a particular instance of resolution with selection, where all negative literals in a clause are selected and, hence, all positive premises (called "electrons") and also the conclusion must be positive clauses (without negative literals). The refutational completeness of resolution with arbitrary selection functions is not difficult to show, cf. (Bachmair and Ganzinger, 1997b; Bachmair and Ganzinger, 1990).

The reflexivity axiom is the only congruence axiom that can be used as an electron in a hyper-resolution inference. But if it is the only electron we obtain only instances of reflexivity that can be ignored, such as

$$\frac{x \approx x \quad u \approx y \rightarrow f(\ldots,u,\ldots) \approx f(\ldots,z,\ldots)}{f(\ldots,x,\ldots) \approx f(\ldots,x,\ldots)}$$

In other words, a hyper-resolution inference is redundant if all its premises are congruence axioms. But hyper-resolution still provides only a very weak control on the proof search for equational problems. For example, even if all the input clauses (other than congruence axioms) are ground unit equations, one can usually not only derive an unbounded number of equational consequences by hyper-resolution, but derive the same equation in many different ways.

4. Paramodulation

Paramodulation was introduced by Robinson and Wos (1969) as an attempt to overcome the problems with resolution-based approaches to equality. The paramodulation inference is a clausal form of the Leibniz law for the replacement of equals by equals, combined with unification:

Paramodulation

$$\frac{C \vee s \approx t \quad D[u]}{(C \vee D[t])\sigma}$$

where σ is the most general unifier of s and u.

The conclusion results from replacing an indicated occurrence of $u\sigma = s\sigma$ in the second premise by the term $t\sigma$ and adding "side conditions" $C\sigma$. In descriptions of paramodulation calculi one usually identifies the symmetric equations $s \approx t$ and $t \approx s$, so that one may also replace $t\sigma$ by $s\sigma$. The paramodulation calculus P also includes the (positive) factoring rule, plus another inference rule that encodes resolution with the reflexivity axiom:

Reflexivity resolution

$$\frac{C \vee s \not\approx t}{C\sigma}$$

where σ is the most general unifier of s and t.

One of the aims of Robinson and Wos (1969) was to design a more "immediate" inference system that combined several resolution steps involving congruence axioms, but ignored the intermediate results, so as to achieve better "convergence" of the saturation process (meaning fewer derived clauses). Unfortunately, the claims for improved convergence are unfounded for unrestricted paramodulation. A related rule, called "demodulation" (Wos et al., 1967), is more effective, but incomplete. It is applied when the context C in the first premise is empty, $u = s\sigma$, and the subterm $t\sigma$ is in some sense simpler (e.g., shorter) than $s\sigma$, *and* has the side-effect that the second premise is deleted. Demodulation is an example of simplification, a concept that is of fundamental importance in automated theorem proving, but difficult to formalize.

4.1. *The Functional Reflexivity Equations*

The refutational completeness of paramodulation was initially only known for functionally reflexive clauses sets. A set of clauses N is called *functionally reflexive* if it contains, for each n-ary function symbol f, a corresponding instance $f(x_1,\ldots,x_n) \approx f(x_1,\ldots,x_n)$ of the reflexivity axiom, with pairwise different variables x_i. Let us denote this set of *functional reflexivity equations* by F.

THEOREM 1. (Robinson and Wos, 1969). *If N is a functionally reflexive set of clauses that is closed under P, then N is equationally satisfiable if and only if it does not contain the empty clause.*

Proof. Let M be $N \setminus \mathsf{F}$, the set N without the functional reflexivity equations. Observe that any clause that can be derived by hyper-resolution (with factoring) from $M \cup \mathsf{C}$ can also be derived by inferences in P from N. For instance, any resolution inference with a monotonicity axiom

$$\frac{C \vee s \approx t \quad x \approx y \rightarrow f(\ldots,x,\ldots) \approx f(\ldots,y,\ldots)}{C \vee f(\ldots,s,\ldots) \approx f(\ldots,t,\ldots)}$$

can be simulated by paramodulation into a functional reflexivity equation

$$\frac{C \vee s \approx t \quad f(\ldots,x,\ldots) \approx f(\ldots,x,\ldots)}{C \vee f(\ldots,s,\ldots) \approx f(\ldots,t,\ldots)} \; .$$

whereas a hyper-resolution

$$\frac{C \vee s \approx t \quad D \vee u \approx v \quad x \approx y \, , x \approx z \rightarrow y \approx z}{(C \vee D \vee t \approx v)\sigma}$$

with the commutation axiom (and most general unifier σ of s and u) is matched by a paramodulation inference

$$\frac{C \vee s \approx t \quad D \vee u \approx v}{(C \vee D \vee t \approx v)\sigma} \; .$$

Hyper-resolution inferences in which all the premises are clauses in M correspond to sequences of paramodulations inferences followed by reflexivity resolution steps.

The significance of the theorem, at first sight, seems to lie in the fact that no explicit inferences with the commutation and monotonicity axioms are required. But if we inspect the proof, we realize that paramodulation in the presence of the functional reflexivity equations is actually a *much* more prolific inference mechanism than hyper-resolution in the presence of the congruence axioms. Observe that hyper-resolution with the monotonicity axioms requires only paramodulations into subterms (variables) of functional reflexivity equations, whereas other hyper-resolutions need only paramodulations at the *top-most* position of terms. Thus, paramodulation inferences into subterms of input clauses actually represent *additional* inferences with no analogous hyper-resolution inferences! Moreover, the functional reflexivity equations are instances of, and hence subsumed by, the reflexivity axiom. If they were really needed, the paramodulation calculus would not be compatible with the eager elimination of subsumed clauses—another disadvantage compared to resolution.

The technical reason for introducing the functional reflexivity equations is that they enable lifting. For resolution lifting is straightforward: if C' and D' are ground instances of C and D, respectively, then any resolvent G' that can be obtained from C' and D' is an instance of a corresponding resolvent G from C and D. That is, we may either first instantiate C and D and then resolve; or else first resolve C and D, and then instantiate the conclusion of the inference. This commutation property does not hold for paramodulation. Consider the inference

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

where the second premise is an instance of $f(x) \approx c$ via the substitution $[f(a)/x]$. There is no paramodulation inference from $a \approx b$ and $f(x) \approx c$ with a conclusion from which $f(f(b)) \approx c$ can be obtained by instantiation. The problem is that the ground inference replaces a subterm in the "substitution part" of the instantiated clause, $(f(x) \approx c)[f(a)/x]$. The functional reflexivity equations allow further instantiation of clauses. For example, we can get $f(a) \approx f(b)$ by paramodulation *into the variable* x of $f(x) \approx f(x)$. The additional inference, again into a variable,

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

shows that lifting is then possible.

The functional reflexivity equations are only useful for paramodulation into variables. Such inferences are not constrained by unification and are difficult to control. In most automated theorem provers they are simply ignored, though it was not clear whether this compromised the completeness of the paramodulation calculus.

In spite (or, perhaps, because) of the serious drawbacks of paramodulation, the paper by Robinson and Wos has stimulated a long line of subsequent research on equational reasoning mechanisms similar to paramodulation, but more practical.

The first key results were the proofs that (i) the functional reflexivity equations are not needed and that, more generally, (ii) paramodulation into variables is unnecessary. Further improvements were obtained in the following areas:

Constraining symmetry. In paramodulation the equality predicate is treated symmetrically, so that either side of an (instantiated) equations can be replaced by the other side. Term rewriting restricts replacements to one direction, when possible.

Literal selection. Selection functions provide better search control for resolution, but are even more desirable for paramodulation, as any positive equation of a clause may have to be paramodulated into any (positive or negative) equation of any other clause.

Term selection. Since all subterms in a clause are possible candidates for replacement via paramodulation, the inherent branching factor in the proof search may be huge. The problem is exacerbated by the instantiation of subterms that tends to produce larger and larger terms. Term selection strategies are an attempt to alleviate these problems.

Simplification. Simplification of formulas and elimination of redundant formulas are essential components of automated theorem provers, designed to constrain the proof search. In fact, in most successful automated provers simplification is the primary mode of computation, whereas expansive deduction rules are used only sparingly. Finding the right combination of deduction and simplification is a fundamental issue in automated deduction.

4.2. *Brand's Modification Method*

In 1975, Brand showed that paramodulation is refutationally complete even without the functional reflexivity axioms. His *modification method* transforms a given set of clauses N to a set N', such that $N \cup \mathcal{C}$ is satisfiable if and only if $N' \cup \{x \approx x\}$ is satisfiable; and, hence, eliminates all equality axioms except reflexivity. The redundancy of the functional reflexivity equations is obtained as a side effect.

The transformation proceeds in two steps. First, nested non-variable subterms are eliminated by introducing new auxiliary variables and corresponding defining equations. (The new variables may be viewed as names of subterms.) For example, the clause $x * i(x) \approx 1 \vee x \approx 0$ contains a nested subterm $i(x)$, that is eliminated by introducing a new variable y and replacing the clause by $i(x) \not\approx y \vee x * y \approx 1 \vee x \approx 0$. This subterm abstraction step is repeated for other nested subterms. The result of the transformation is a *flat* clause in which only variables occur as arguments of function symbols. It can be shown, by a rather elementary argument about canonical representations of substitutions (that will reappear in a refined form below), that one can dispense with the monotonicity axioms in testing satisfiability of flat clauses.

LEMMA 1. (Brand, 1975). *If N is a set of flat clauses, then $N \cup \mathsf{C}$ is satisfiable if and only if $N \cup \mathsf{E}$ is satisfiable.*

Proof. Let I be a Herbrand model for $N \cup \mathsf{E}$. Then $\approx_I$ is an equivalence relation on ground terms. Moreover, for any ground term t, let $\tilde{t}$ be some arbitrary, but fixed, representative in the equivalence class of t modulo $\approx_I$. We construct a new Herbrand interpretation $\tilde{I}$ (with the same domain as I) by taking $\approx_{\tilde{I}}$ to be $\approx_I$ and defining a function $f_{\tilde{I}}$ on ground terms by $f_{\tilde{I}}(t_1, \ldots, t_m) = f(\tilde{t}_1, \ldots, \tilde{t}_m)$, for every function symbol f. It can easily be proved that the new interpretation $\tilde{I}$ satisfies the congruence axioms C.

Let now σ be any ground substitution and define the substitution $\tilde{\sigma}$ by $x\tilde{\sigma} = \widetilde{x\sigma}$. Then

$$f_I(x_1\tilde{\sigma}, \ldots, x_k\tilde{\sigma}) = f(x_1\tilde{\sigma}, \ldots, x_k\tilde{\sigma}) = f_{\tilde{I}}(x_1\sigma, \ldots, x_k\sigma)$$

and $x\sigma \approx_I x\tilde{\sigma}$. Therefore, the values of a flat term u in the interpretations I and $\tilde{I}$ and under assignments $\tilde{\sigma}$ and σ, respectively, are equivalent. Thus, if A is a flat equation then $I, \tilde{\sigma} \models A$ iff $\tilde{I}, \sigma \models A$. Since I is a model of N, we have $I, \tilde{\sigma} \models C$ for all clauses C in N. Consequently, we also have $\tilde{I}, \sigma \models C$. In short, $\tilde{I}$ is a model of N, which completes the proof.

The key idea of the proof is to represent a substitution σ by a distinguished representative $\tilde{\sigma}$ from which the truth value of the σ-instance of a clause can be inferred. If no nested function symbols occur, the terms $\widetilde{f(\tilde{t})}$ and $\widetilde{f(t)}$ need not be equivalent.

The second transformation step eliminates the commutation axiom by splitting all positive equations via the introduction of so-called *link variables*. More specifically, a clause $C \vee s \approx t$ is replaced by two clauses $C \vee s \not\approx x \vee t \approx x$ and $C \vee s \approx x \vee t \not\approx x$, where x is a new variable. Any remaining positive equations in the new clauses are split in a similar way. The final clauses are called *ST-forms*.

For example, the above clause yields four transformed clauses $(i(x) \not\approx y \vee x*y \not\approx z \vee x \not\approx u \vee 1 \approx z \vee 0 \approx u)$, $(i(x) \not\approx y \vee x*y \not\approx z \vee x \approx u \vee 1 \approx z \vee 0 \not\approx u)$, $(i(x) \not\approx y \vee x*y \approx z \vee x \not\approx u \vee 1 \not\approx z \vee 0 \approx u)$, and $(i(x) \not\approx y \vee x*y \approx z \vee x \approx u \vee 1 \not\approx z \vee 0 \not\approx u)$.

The *ST*-forms of a clause are essentially obtained by resolution (modulo the symmetry of $\approx$) with the commutation law.

LEMMA 2. (Brand, 1975). *If N is the set of ST-forms of a set M of flat clauses, then M is equationally satisfiable if, and only if, $N \cup \{x \approx x\}$ is satisfiable.*

THEOREM 2. (Brand, 1975). *A set of equational clauses N that is closed under P is equationally satisfiable if, and only if, it does not contain the empty clause.*

Proof. By the preceding lemma, the set $N \cup \mathcal{C}$ is satisfiable if, and only if, the transformed set $N' \cup \{x \approx x\}$ is satisfiable. It can be shown that resolution inferences with clauses in N'—for a specific selection strategy—can be simulated by paramodulation with clauses in N. (On the ground level, the selection strategy simulates innermost rewriting.)

Brand's results also point out other redundancies in unrestricted paramodulation. Most of the paramodulation inferences needed to simulate resolution (with the specific selection strategy) take place at non-variable positions that were present in the initial set of clauses. Almost (but not quite) all paramodulations into variables or into terms introduced by unification in previous inference steps are unnecessary. For instance, the clauses $f(x) \approx z \rightarrow x \approx z$ and $y \approx z \rightarrow g(y) \approx z$ are *ST*-forms of $f(x) \approx x$ and $g(y) \approx y$, respectively. Resolution on the transformed clauses corresponds to paramodulation into the variable side of one of the equations. A variation of the modification method proposed by Moser and Steinbach (1997) largely avoids such inferences. Bachmair et al. (1997) prove the completeness of an even more optimized variant in the presence of ordering constraints for the auxiliary variables introduced by flattening and splitting.

Brand's modification method has been one of the more successful approaches to equality handling in non-local theorem proving methods, such as model elimination and semantic tableaux (Ibens and Letz, 1997). Other methods designed to improve the handling of equality in such provers (Moser et al., 1995; Degtyarev and Voronkov, 1996b; Degtyarev and Voronkov, 1996a) require subtle interactions between paramodulation-based and model elimination-based subcomponents and therefore are difficult to integrate into existing provers. Transformation methods typically flatten terms, so that the filtering effect of unification is much weaker on transformed clauses. These clauses also may become very long, so that many other heuristics for inference selection do not work well any more. Model elimination and semantic tableaux, in general, appear to be inherently limited in their capability for solving difficult equational problems.

5. Rewriting Techniques for Equational Reasoning

Brand's results demonstrated that the functional reflexivity equations are not needed, but left open whether paramodulation into variables is necessary. The difficulty is that paramodulation *from* variables and paramodulation *into* variables can not be clearly distinguished as separate resolution inferences on transformed clauses; and the former inferences are in general not redundant. We now turn to term rewriting methods, as pioneered by Knuth and Bendix (1970) with their completion procedure. Term rewriting provided important clues about the question of avoiding paramodulation into variables, and also contributed essential techniques for refining paramodulation into a practical inference system.

For instance, the problem of determining whether a ground equation $s \approx t$ logically follows[2] from a given set of ground equations E, can be decided by congruence closure techniques. But paramodulation from E and the negated goal $s \not\approx t$ may produce infinitely many new ground equations. For example, if E contains the equation $f(a) \approx a$, we obtain infinitely many equations of the form $f^n(a) \approx a$ by paramodulation. Term rewriting techniques prevent such unnecessary deductions.

5.1. *Knuth-Bendix Completion*

Paramodulation uses either side of an equation to replace a subterm in a clause by the other side. To Knuth and Bendix this was reminiscent of an ad-hoc "trial-and-error manner" in which certain algebraic problems had been traditionally treated. They were interested in the word problem for algebraic theories; deciding, for a given set of equations E and an equation $s \approx t$ (possibly containing variables), whether or not $E \models s \approx t$.[3] Their intention was to

[2] From now one we restrict ourselves to equality interpretations, and will simply speak of "satisfiability," "logical consequence," etc. when in fact we refer to the corresponding notions in equational logic.

[3] The definition of "word problems" unfortunately differs among authors.

solve word problems by using the equations in E as rewrite rules, and proving $s \approx t$ by rewriting both s and t to a common term u.

We shall explain the basic ideas of their approach. In the reformulation of the word problem as a refutation theorem proving problem, one deals with unit clauses only, positive (non-ground) unit equations in E and a ground disequation obtained by negating (and skolemizing) $s \approx t$ Semantically, this simplicity is mirrored by the existence of a unique minimal model for E—the quotient algebra $T_{\Sigma(X)}/E$ obtained by dividing the free Σ-algebra over (Skolem constants) X by the least congruence containing E.

Basic Concepts in Term Rewriting

A *rewrite rule* is an ordered pair of terms, written $l \Rightarrow r$. A *rewrite relation* is a binary relation that is closed under substitution and context application, i.e., if $s \Rightarrow t$, then $u[s\sigma] \Rightarrow u[t\sigma]$, for all substitutions σ and terms u. The rewrite relation $\Rightarrow_R$ *induced* by a set of rewrite rules R is the smallest rewrite relation containing R. We also say that a term u *rewrites* to another term v (by R), if $u = u[s\sigma]$ and $v = u[t\sigma]$, for some rewrite rule $s \Rightarrow t$ (in R) and some substitution σ. (The indicated position of $s\sigma$ in u is called a *redex*.) A term that can not be rewritten is said to be *irreducible*. A term t' is called a *normal form* of t (by R) if $t \overset{*}{\Rightarrow}_R t'$ and t' is irreducible by R. (We denote by $\Leftarrow$ the inverse, by $\Leftrightarrow$ the symmetric closure, and by $\overset{*}{\Leftrightarrow}$ the equivalence closure, respectively, of $\Rightarrow$. It is not hard to show that $\overset{*}{\Leftrightarrow}_R$ is the least congruence containing R.)

If a rewrite relation is well-founded, that is, admits no infinite sequences of rewrite steps, then every term has a normal form, and we may speak of a *reduction relation*. A *reduction ordering* is a well-founded strict partial ordering that is also a rewrite relation. A rewrite system is said to be *terminating* if it induces a well-founded rewrite relation, which is the case if and only if it is contained (as a subset) in some reduction ordering. For a detailed exposition of termination of rewrite systems, see (Dershowitz, 1987). We will usually use specific reduction orderings, called *lexicographic path orderings*, in examples.

Termination guarantees the existence of normal forms, but not their uniqueness. Normal forms are unique if a rewrite system is *confluent*, that is, whenever a term s can be rewritten to terms t and u ($s \overset{*}{\Rightarrow}_R t$ and $s \overset{*}{\Rightarrow}_R u$), then t and u are “joinable” for some term v ($t \overset{*}{\Rightarrow}_R v$ and $u \overset{*}{\Rightarrow}_R v$). We also say that the equation $t \approx u$ *converges*, or has a *rewrite proof* in R, and write $t \Downarrow_R u$.

A terminating, confluent rewrite system is called *convergent*. If R is a convergent rewrite system, then rewriting provides a decision procedure for the underlying equational theory: $s \overset{*}{\Leftrightarrow}_R t$ if and only if $s \approx t$ converges; that is, two terms are equivalent if, and only, if they reduce to the same normal form. Normalization (computation of normal forms by repeated reduction) is a don't care non-deterministic process for convergent rewrite systems.

The Completion Procedure
Knuth and Bendix (1970) proved that testing whether a terminating finite rewrite system is confluent can be done by applying normalization to certain (finitely many) equations. Consider the following restricted version of paramodulation for rewrite rules:

Superposition of rewrite rules

$$\frac{s \Rightarrow t \quad w[u] \Rightarrow v}{(w[t] \approx v)\sigma}$$

where σ is the most general unifier of s and u such that u is not a variable.

The equation $(w[t] \approx v)\sigma$ is called a a *critical pair* between the two premises; and the term $w[u]\sigma$, the *overlapped term*. (Note the restriction that u be a non-variable subterm: paramodulations into variables are not superpositions.)

LEMMA 3. (Critical Pair Lemma, Knuth and Bendix, 1970). *A terminating rewrite system R is confluent if and only if all critical pairs between rules in R converge.*

The lemma shows that confluence is decidable for finite, terminating rewrite systems, as only finitely many critical pairs need to be examined in that case. It also indicates that the reason for non-convergence can be found in a critical pair; an observation that provides a starting point for a completion procedure. Typically, completion processes critical pairs by first normalizing their terms and, in case of non-convergence, orienting the normalized equation into a rule so as to preserve the termination of the overall rewrite system. This process need not terminate, though, and may fail when equations such as $x+y \approx y+x$ are produced that can not be oriented.

The Knuth-Bendix completion procedure is bound to fail for equational theories with an undecidable word problem, of course, but the method can be refined so that it can always be used at least as a semi-decision procedure. Completion works reasonably well if the given equations contain no variables. In that case the procedure will always terminate and thus provide a decision procedure for the given equational theory. The efficiency of ground completion can often be improved by a technique, similar to variable abstraction, of introducing new constants to rename subterms. This provides for a more compact representation of terms (as multiple occurrences of the same subterm will be represented by the same constant), and can be used to simulate congruence closure techniques (Kapur, 1997).

Ordered Rewriting

Reduction orderings are closed under application of substitutions and therefore can not be total on terms with variables. (Two terms, each of which contains a variable not occurring in the other term, are not comparable in any reduction ordering.) But many reduction orderings are total on *ground* term. Such reduction orderings are called *complete*. They include any lexicographic path ordering based on a total (well-founded) precedence. Fortunately, in theorem proving applications one can often reason about non-ground formulas by considering the corresponding ground instances. Ordered rewriting techniques can be applied in such situations.

Let $\succ$ be a reduction ordering and E be a set of equations. The rewrite system $E^{\succ}$ is defined as the set of all instances $s\sigma \Rightarrow t\sigma$ of equations $s \approx t$ or $t \approx s$ in E, for which $s\sigma \succ t\sigma$. Rules in $E^{\succ}$ are also called *reductive instances* of equations in E. The rewrite relation $\Rightarrow_{E^{\succ}}$ represents *ordered rewriting* with respect to E and $\succ$.

For example, let $\succ$ be a lexicographic path ordering with precedence $+ \succ a \succ b \succ c$. Then $b+c \succ c+b \succ c$, and we may use the commutativity equation $x+y \approx y+x$ for ordered rewriting, $(b+c)+c \Rightarrow (c+b)+c \Rightarrow c+(c+b)$.

If $\succ$ is a complete reduction ordering, then $E^{\succ}$ contains all (non-trivial) ground instances of an equation $s \approx t$ in E, either as a rule $s\sigma \Rightarrow t\sigma$ or a rule $t\sigma \Rightarrow s\sigma$. (An instance is trivial if $s\sigma$ and $t\sigma$ are identical.) A rewrite system R is called *ground convergent* if the induced ground rewrite relation (that is, the rewrite relation $\Rightarrow_R$ restricted to pairs of ground terms) is terminating and confluent. A set of equations E is called *ground convergent with respect to* $\succ$ if $E^{\succ}$ is ground convergent.

Ordered rewriting leads to the following inference rule:

Superposition

$$\frac{s \approx t \quad w[u] \approx v}{(w[t] \approx v)\sigma}$$

where σ is the most general unifier of s and u such that $t\sigma \not\succeq s\sigma$, $v\sigma \not\succeq w\sigma$, and u is not a variable.

The equation $(w[t] \approx v)\sigma$ is called an *ordered critical pair* (with *overlapped term* $w[u]\sigma$) between the premises of the inference.

LEMMA 4. (Lankford, 1975). *Let $\succ$ be a complete reduction ordering. A set E of equations is ground convergent with respect to $\succ$ if, and only if, for* *all ordered critical pairs* $(w[t] \approx v)\sigma$ *(with overlapped term* $w[u]\sigma$*) between equations in* E *and for all ground substitutions* τ*, if* $w[u]\sigma\tau \succ w[t]\sigma\tau$ *and* $w[u]\sigma\tau \succ v\sigma\tau$*, then* $w[t]\sigma\tau \Downarrow_{E^{\succ}} v\sigma\tau$.

Note that the condition $t\sigma \not\succeq s\sigma$ imposed on superposition inferences represents an approximation to the condition $(w[u])\sigma\tau \succ (w[t])\sigma\tau$ and needed in the critical pair lemma (and similarly for the other condition). If the first condition is violated, the second condition can not be satisfied for any ground substitution τ. The difficulty in using the ordered critical pair lemma as the basis for a decision procedure for ground convergence lies in the fact that convergence has to be tested for a possibly infinite number of ground substitutions. In fact, in its general form the problem is undecidable. However, Comon et al. (1998) have recently proved the decidability of ground convergence with respect to lexicographic path orderings. They use ordering constraints (see section 7.3 below) to obtain a finite representation of infinitely many instances. Related techniques have been worked out for specific examples by Martin and Nipkow (1990).

Various forms of ordered rewriting have been used in the literature. Peterson (1983) used a restricted form of ordered rewriting in his proof that paramodulation into variables is not needed. Hsiang and Rusinowitch (1987) describe a special variant of ordered completion for refutational theorem proving applications. A general ordered completion method was presented by Bachmair et al. (1989).

Ordered Completion and Proof Orderings

The critical pair lemma indicates which equations need to be deduced by a completion procedure. For efficiency reasons simplification mechanisms have to be used extensively during the completion process, which complicates the task of proving the correctness of these procedures. Huet (1981) was the first to give a detailed correctness proof for a specific completion procedure. Bachmair et al. (1986) argued that completion should be described in more abstract terms, by inference rules. The completion process then corresponds to a derivation, a sequence of inference steps on sets of equations and rewrite rules. The presentation of the given equational theory, but not the theory itself, changes with each step.

In figure 1 we display the inference rules of (a version of) ordered completion. They specify a family of inference systems, parameterized by a reduction ordering $\succ$. We write $E;R \vdash E';R'$ if $E';R'$ can be obtained from $E;R$ by applying a completion inference. A (finite or countably infinite) sequence $(E_0;R_0) \vdash (E_1;R_1) \vdash \cdots$ is called a *derivation*. Usually, E_0 is the set of initial equations and $R_0 = \emptyset$. The *limit* of a derivation is the pair $E_\infty;R_\infty$, where

Deduction	$E;R \vdash E\cup\{s\approx t\};R$ if $s \Leftrightarrow_{E\cup R} w \Leftrightarrow_{E\cup R} t,\ s \not\succeq w,$ and $t \not\succeq w$
Orientation	$E\cup\{s\approx t\};R \vdash E;R\cup\{s\Rightarrow t\}$ if $s\succ t$
Deletion	$E\cup\{s\approx s\};R \vdash E;R$
Simplification	$E\cup\{s\approx t\};R \vdash E\cup\{u\approx t\};R$ if $s\Rightarrow_R u$
Composition	$E;R\cup\{s\Rightarrow t\} \vdash E;R\cup\{s\Rightarrow u\}$ if $r\Rightarrow_R u$
Collapse	$E;R\cup\{s[w]\Rightarrow t\} \vdash E\cup\{s[u]\approx t\};R$ if $w\Rightarrow_R u,$ and either $t\succ u$ or $w\neq s$

Figure 1. Ordered Completion Derivations $\vdash$

$E_\infty = \cup_i \cap_{j\geq i} E_i$ and $R_\infty = \cup_i \cap_{j\geq i} R_i$. The goal is to obtain a limit system that is ground convergent. To characterize successful derivations in this sense we first have to look at equational proofs.

An *(equational) proof* of $s\approx t$ with respect to $E;R$ is a sequence of proof steps

$$s = s_0\ \rho_0\ s_1\ \rho_1\ s_2 \ldots s_{n-1}\ \rho_{n-1}\ s_n = t,\ n\geq 0,$$

where each step $s_i\ \rho_i\ s_{i+1}$ is either (i) an equality step $w[u\sigma] \Leftrightarrow w[v\sigma]$, with $u\approx v$ in E, or (ii) a rewrite step $w[u\sigma] \Rightarrow w[v\sigma]$, with $u\Rightarrow v$ in $R\cup E^{\succ}$, or (iii) a reverse rewrite step $w[u\sigma] \Leftarrow w[v\sigma]$, with $v\Rightarrow u$ in $R\cup E^{\succ}$. An equation is in the equational theory generated by $E\cup R$ if and only if it has a proof in $E;R$. A proof of the form

$$s = s_0 \Rightarrow \ldots \Rightarrow s_m \Leftarrow \ldots \Leftarrow s_n = t.$$

is called a *rewrite proof*. A rewrite proof exists if and only if the equation converges with respect to $R\cup E^{\succ}$. In a ground convergent system all variable-free equational consequences have a rewrite proof.

It can easily be checked that the completion inference rules do not change the given equational theory itself, but only its presentation. That is, whenever $E;R \vdash (E',R')$ then the same equations are provable in $E;R$ as in $E';R'$. However, some of the proofs in $E';R'$ will be *simpler* than the corresponding proofs in $E;R$ with respect to an appropriately defined well-founded ordering on proofs (a *proof ordering*). The following ordering will do the job. We associate with each proof step a measure of its cost. More specifically, the cost of $s[u]\ \rho\ s[v]$ is

$$\begin{cases} (\{s[u]\}, u, \rho, s[v]), & \text{if } s[u] \succ s[v] \\ (\{s[v]\}, v, \rho, s[u]), & \text{if } s[v] \succ s[u] \\ (\{s[u], s[v]\}, \bot, \bot, \bot), & \text{otherwise} \end{cases}$$

The complexity of a proof is defined as the multiset of the costs of its proof steps. Cost comparisons are done lexicographically by (i) the multiset extension of the reduction ordering $\succ$, (ii) the *encompassment ordering* $\rhd$,[4] (iii) an ordering with $\Leftrightarrow > \Rightarrow$ and $\Leftrightarrow > \Leftarrow$, and (iv) the reduction ordering $\succ$, with $\bot$ minimal in any of these orderings. Let $\succ_\pi$ denote the corresponding multiset extension, which is a well-founded partial ordering on proofs.

Each completion step decreases the complexity of certain proofs. For example, orienting an equation $s \approx t$ into a rule $s \Rightarrow t$ is reflected by a proof transformation

$$(u[s\sigma] \Leftrightarrow u[t\sigma]) \Longrightarrow (u[s\sigma] \Rightarrow u[t\sigma])$$

that is simplifying: $(\{u[s\sigma]\}, s\sigma, \Leftrightarrow, s[t\sigma]) \succ_\pi (\{u[s\sigma]\}, s\sigma, \Rightarrow, s[t\sigma])$. The reader may check that the other inference rules are also reflected by simplifying proof transformations.

We now know that each derivation step simplifies some proofs. But what is required is that eventually all undesirable proofs (i.e., non-rewrite proofs) are simplified. Derivations that achieve this stronger proof normalization requirement can be characterized by a suitable notion of fairness. We call a derivation *fair* if each ordered critical pair $u \approx v$ in $E_\infty \cup R_\infty$ is an element of some set E_i. In short, a derivation is fair if all ordered critical pairs between persistent equations and rules are eventually computed. Fair derivations can be effectively enumerated by straightforward inference recording mechanisms, provided E_0 and $\succ$ are effectively given. Note that fairness depends only on deduction of critical pairs. Simplification steps are optional, though indispensable in practice.

THEOREM 3. (Bachmair et al., 1989). *Let $(E_0;R_0),(E_1;R_1),\ldots$ be a fair ordered completion derivation. Then these three properties are equivalent for all ground terms s and t: (i) $s \approx t$ is a consequence of E_0; (ii) $s \approx t$ has a rewrite proof in $E_\infty^\succ \cup R_\infty$; (iii) $s \approx t$ has a rewrite proof in $E_i^\succ \cup R_i$, for some i.*

[4] $s \unrhd t$ if, and only if, some subterm of s is an instance of t, and $\rhd$ denotes the strict part of this quasi-ordering.

Since fair derivations can be effectively constructed, the theorem in particular asserts the refutational completeness of ordered completion, and of superposition with simplification for equations.

We have now shown that for unit equations paramodulation into variables is not required. Moreover, paramodulation inferences can be constrained by ordering restrictions so that one only replaces subterms of the maximal term of an equation by some other term that is not larger in the reduction ordering. The reduction ordering is a parameter that can be freely chosen, provided it is completable. We have also seen that superposition is compatible with the powerful simplification inferences of ordered completion that can be eagerly applied in any order without impairing refutational completeness. A rather liberal notion of fairness allows one to cover a large spectrum of concrete completion procedures. The concept of proof transformations admits, in particular, simple proofs of admissibility of most, if not all, criteria for identifying redundant critical pairs that have been proposed in the literature. A detailed treatment of ordered completion can be found in (Bachmair, 1991).

5.2. *Superposition for Horn Clauses*

The results about ordered completion and superposition for unit clauses were obtained by proof-theoretic arguments that rely on the simple structure of equational proofs and are directly related to the canonical models (initial algebra semantics) of equational theories. Theories presented by equational Horn clauses have similar semantic properties and the corresponding proof theory is only modestly more complex.

The canonical model of a set P of non-equational Horn clauses can be obtained by a construction, familiar from logic programming, based on computing the least fixed point $\bigcup_n T_P^n(\emptyset)$ of the T_P-operator

$$T_P : I \mapsto \{A \mid (B_1, \ldots, B_n \to A) \in g(P), \text{ and } B_i \in I\},$$

where $g(P)$ denotes the set of ground instances of P. An instance $B_1, \ldots, B_n \to A$ of a clause in P *produces* the atom A for the canonical model, if, for some $n \geq 0$, all the B_i are in $T_P^n(\emptyset)$, but A is not. The T_P-operator also induces a well-founded partial ordering on ground atoms:

$$A \succ A' \text{ if } \min\{n \mid A \in T_P^n(\emptyset)\} > \min\{n \mid A' \in T_P^n(\emptyset)\}.$$

Productive instances $B_1, \ldots, B_n \to A$ of clauses in P may be viewed as *reductive* conditional rewrite rules, as $A \succ B_i$, for all i.

In the equational case, one needs to take into account the congruence properties of equality. An equational Horn clause $u_1 \approx v_1, \ldots, u_k \approx v_k \to s \approx t$ is called *reductive* for $s \Rightarrow t$ (with respect to a reduction ordering $\succ$), if s is the strictly maximal term in the clause; that is, $s \succ t$ and $s \succ u_i$ and $s \succ v_i$. We also write $u_1 \approx v_1, \ldots, u_k \approx v_k \rightarrow s \Rightarrow t$ to indicate that a clause is reductive for $s \Rightarrow t$. If R is a set of reductive clauses we recursively define a rewrite relation $\Rightarrow_R$ as follows: $u \Rightarrow_R v$ if there exist a reductive clause $C \rightarrow s \Rightarrow t$ in R and a substitution σ, such that $u = u[s\sigma]$ and $v = u[t\sigma]$, and $u'\sigma \Downarrow_R v'\sigma$ for all equations $u' \approx v'$ in C. (The recursion is well-founded as the rules in R are reductive and the union of any reduction ordering with the strict subterm ordering is a well-founded ordering.) The relation $\Rightarrow_R$ is terminating and, for finite R, decidable.

Superposition of reductive rewrite rules

$$\frac{C \rightarrow s \Rightarrow t \quad D \rightarrow w[u] \Rightarrow v}{(C, D \rightarrow w[t] \approx v)\sigma}$$

where σ is the most general unifier of s and u, and u is not a variable.

The clause $(C, D \rightarrow w[t] \approx v)\sigma$ is called a *conditional critical pair* between the premises of the inference. The critical pair is said to *converge* in R if for all substitutions τ for which all equations in $C\sigma\tau$ and $D\sigma\tau$ converge in R, the equation $(w[t] \approx v)\sigma\tau$ converges, too, that is, $w[t]\sigma\tau \Downarrow_R v\sigma\tau$.

LEMMA 5. (Jouannaud and Waldmann, 1986). *A reductive conditional rewrite system is confluent if and only if all its critical pairs converge.*

This lemma provides a confluence criterion that slightly improves an earlier result by Kaplan (1987), but is undecidable in general. Jouannaud and Waldmann (1986) also proved that if R is confluent and reductive, then an equation $s \approx t$ is true in the equational theory of R if, and only if, $s \Downarrow_R t$.

THEOREM 4. (Jouannaud and Waldmann, 1986). *Let $\succ$ be a reduction ordering and N be a finite set of reductive equational Horn clauses. If all critical pairs from N converge, then the word problem for N is decidable.*

Additional inferences may be necessary to eliminate non-reductive clauses with the maximal term in a condition. Let us illustrate the situation with an example. Let $\succ$ be a lexicographic path ordering with precedence $f \succ g \succ a \succ b \succ c$ and consider the clauses

$$\begin{array}{rcl} f(f(x)) \approx x & \rightarrow & f(x) \approx g(x) \\ a \approx c & \rightarrow & f(c) \approx c \\ & \rightarrow & a \approx b \\ & \rightarrow & b \approx c \end{array}$$

The last three clauses are reductive and produce no critical pairs. Thus, the associated rewrite relation is convergent. The first clause is non-reductive, as the maximal term $f(f(x))$ occurs in the condition. This clause is not satisfied in the equational theory generated by the other, reductive clauses. If we substitute c for x we obtain an instance where the condition, but not the consequent, has a rewrite proof. The existence of a rewrite proof for a condition indicates that there is a "negative superposition" inference,

$$\frac{a \approx c \to f(c) \approx c \quad f(f(x)) \approx x \to f(x) \approx g(x)}{a \approx c, f(c) \approx c \to f(c) \approx g(c)} .$$

The conclusion of this inference is not reductive, but can be reduced by a similar inference,

$$\frac{a \approx c \to f(c) \approx c \quad a \approx c, f(c) \approx c \to f(c) \approx g(c)}{a \approx c, a \approx c, c \approx c \to f(c) \approx g(c)}$$

that produces a reductive clause. The expanded set of reductive clauses is no longer convergent, though. There is a conditional critical pair,

$$\frac{a \approx c \to f(c) \approx c \quad a \approx c, a \approx c, c \approx c \to f(c) \approx g(c)}{a \approx c, a \approx c, a \approx c,\ c \approx c \to g(c) \approx c} ,$$

that produces another reductive clause. At this point, the subset of reductive clauses is convergent, and the two non-reductive clauses (the first clause and the conclusion of the first inference) are true in the theory generated by the reductive clauses.

Note that negative superposition inferences are characterized by conditions similar to superposition of reductive rewrite rules: (i) the first premise is a reductive clause, (ii) replacement takes place in the maximal literal of the second premise, and (iii) the term to be replaced is not a variable and is replaced by a smaller term. One may weaken these conditions and repeatedly superpose on all conditions (including non-maximal ones) and remove any trivial conditions of the form $s \approx s$, until an unconditional equation is obtained. Such unit strategies for equational Horn clauses have been considered by Paul (1986). In the limit one obtains a clause set, the equational theory of which is represented by the subset of unit equations. But this strategy will only terminate in trivial cases, whereas strategies designed to produce reductive clauses will terminate more often.

Superposition calculi and completion procedures for Horn clauses based on these ideas have been presented and proved complete by several authors, including (Rusinowitch, 1987; Kounalis and Rusinowitch, 1987; Bachmair, 1991; Ganzinger, 1991). The approach in (Ganzinger, 1991) extends the method of proof orderings to Horn clauses. We do not go into details since these results are largely subsumed by the results about superposition for full clauses discussed next.

6. Superposition for First-Order Clauses

For general clauses the semantic situation is more complex than for Horn clauses, as minimal models need not be unique in the presence of disjunctions of positive literals. But proof-theoretic methods are difficult to apply to paramodulation-like calculi, as the corresponding derivations reveal some intricate non-local structure (Bachmair, 1989). Most refutational completeness results for such calculi have been obtained by semantic methods that systematically explore Herbrand interpretations of given clauses. We will describe one such method that is based on the idea of reducing counterexamples for "candidate models." This approach also induces a powerful, yet natural, concept of redundancy for clauses and inferences that covers virtually all known simplification and redundancy elimination techniques.

The inference rules of the superposition calculus S (with selection) for general clauses are shown in figure 2. These rules are formulated in terms of two parameters: a completable reduction ordering and a selection function, see also section 3.[5] We speak of *superposition without selection* if no literals are selected in any clause. Note that only clauses without selected literals may occur as positive premises in superposition inferences. We again identify any equation or disequation, respectively, with its symmetric variant.

The well-founded ordering $\succ$ on literals and terms has to be *admissible* in the sense that (i) the restriction of $\succ$ to ground literals is a total ordering, (ii) the restriction of $\succ$ to terms is a complete reduction ordering, and (iii) if L and L' are ground literals, then $L \succ L'$, whenever (iii.1) $\max(L) \succ \max(L')$, or (iii.2) $\max(L) = \max(L')$ and the literal L is negative, whereas L' is positive. Here $\max(L)$ denotes the maximal term (according to $\succ$) of the literal L. Condition (iii) in essence states that replacing the maximal term in a literal by a smaller term yields a smaller literal, while sub-condition (iii.2) indicates that clauses in which the maximal term occurs negatively are larger than clauses with the same maximal term, but occurring only in positive literals. Thus, the comparison of two literals in an admissible ordering depends primarily on the respective maximal terms, and hence on the underlying term ordering. Addi-

[5] For simplicity we do not consider multi-premise inferences, such as hyper-paramodulation, and assume that at most one negative literal is selected in any clause.

Positive superposition

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

where σ is the mgu of s and s' such that (i)–(vi).

Negative superposition

$$\frac{C \vee s \approx t \quad D \vee u[s'] \not\approx v}{(C \vee D \vee u[t] \not\approx v)\sigma}$$

where σ is the mgu of s and s' such that (i)–(iii),(iv'),(v).

Reflexivity resolution

$$\frac{D \vee u \not\approx v}{D\sigma}$$

where σ is the mgu of u and v such that (iv').

Ordered factoring

$$\frac{C \vee u \approx v \vee s \approx t}{(C \vee s \approx t)\sigma}$$

where σ is the mgu of $s \approx t$ and $u \approx v$ such that (iii').

Equality factoring

$$\frac{C \vee u \approx v \vee s \approx t}{(C \vee v \not\approx t \vee u \approx t)\sigma}$$

where σ is the mgu of s and u such that (iii') and (vi).

The restrictions are defined as follows:

(i) $t\sigma \not\succeq s\sigma$ — (ii) $v\sigma \not\succeq u\sigma$ — (iii) $(s \approx t)\sigma$ is strictly maximal with respect to $C\sigma$, and C contains no selected literal — (iv) $(u \approx v)\sigma$ is strictly maximal with respect to $D\sigma$, and D contains no selected literal — (v) s' is not a variable — (vi) $(s \approx t)\sigma \not\succeq (u \approx v)\sigma$

(iii') $(s \approx t)\sigma$ is maximal with respect to $C\sigma$, and C contains no selected literal — (iv') $u \not\approx v$ is selected, or else nothing is selected in this premise and $(u \not\approx v)\sigma$ is maximal with respect to $D\sigma$

Figure 2. *Superposition with Selection for General Clauses* S

tional information (e.g., about the respective minimal terms) is needed when two literals are of the same polarity with identical maximal terms.

The multiset extension of an ordering on literals yields an ordering on clauses. We speak of an *admissible clause ordering* if the underlying ordering on literals is admissible (and usually use the same symbol $\succ$ for both orderings). Admissible clause orderings are well-founded and total on ground clauses. We shall consider only admissible orderings from now on.[6]

Several variations of superposition have been proposed in the literature. For instance, one may replace equality factoring by an inference rule called *merging paramodulation*, that allows one to replace subterms in the smaller term t of a maximal equations $s \approx t$, but only under similar restrictions as for equality factoring. *Ordered paramodulation* does not impose the ordering constraints (ii) and (vi), and hence permits replacement of subterms in the smaller side of equations. (Equality factoring is not needed for this variant.)

THEOREM 5. (Hsiang and Rusinowitch, 1991). *Ordered paramodulation is refutationally complete.*

The original proof of this theorem in (Hsiang and Rusinowitch, 1991) employs a proof technique based on transfinite semantic trees, and covers only specific selection strategies such as the positive strategy or the positive unit strategy in the case of Horn clauses.

6.1. *Candidate Models and Counterexamples*

Candidate models and reduction of counterexamples are the key concepts in establishing the refutational completeness of saturation-based refutation theorem proving methods. We explain the basic ideas for the case of ground clauses and discuss issues related to lifting later.

Let Π be a clausal inference system and $\Pi(N)$ be the subset of inferences in Π with premises in N. If π is an inference of the form

$$\frac{C_1 \quad \dots \quad C_n \quad C}{D}$$

with $n \geq 0$, we call the (rightmost) premise C the *main premise*, and the remaining premises C_i the *side premises*, of the inference. The conclusion D of the inference will also be denoted by $\Gamma(\pi)$.

[6] In the theorem proving literature one can also find slightly different (often implicit) definitions of admissibility of clause orderings (or quasi-orderings). In particular, the sub-condition (iii.2) is sometimes not required. The differences are not of importance conceptually, but are critical in justifying certain technical optimizations.

Let now I be a mapping, called a *model functor*, that assigns to each set N of ground clauses that does not contain the empty clause, an equality Herbrand interpretation I_N, called a *candidate model*. If I_N is a model of N, then N is evidently satisfiable. If, on the other hand, some clause C in N is false in I_N (a *counterexample* for I_N), then N must contain a *minimal* such *counterexample* with respect to $\succ$. We say that Π has the *reduction property for counterexamples* (with respect to I) if, for all clause sets N and all minimal counterexamples C for I_N, there exists an inference in Π with main premise C, side premises that are true in N, and a conclusion D that is a smaller counterexample for I_N than C, that is, $C \succ D$. Inference systems with this property are refutationally complete:

PROPOSITION 1. *If Π has the reduction property for counterexamples and N is closed under Π, that is, $\Gamma(\Pi(N)) \subseteq N$, then N is either satisfiable, or else contains the empty clause.*

Let us next consider the problem of constructing, for a given inference system Π, a model functor I such that Π has the reduction property for counterexamples with respect to I. In particular, we shall examine the superposition calculus from this perspective (but for simplicity discuss only the calculus without selection). For that purpose we only need to consider inferences on ground clauses that are *monotone* in the sense that the conclusion is smaller than the main premise.

We describe equality Herbrand interpretations by convergent (with respect to $\succ$) ground rewrite systems.[7] This allows us to characterize provability, and truth, via rewriting: a (ground) clause is true in such an interpretation if and only if, whenever all negative equations have rewrite proofs, then some positive equation has a rewrite proof. The concept of reductivity of clauses will provide us with a way of extracting suitable rewrite rules from clauses. The idea is to define the candidate model I_N for N as the least model generated by "some maximal" convergent subset of reductive clauses in N which will be called productive.

A clause $C \vee s \approx t$ is called *reductive* (for $s \Rightarrow t$) if $s \succ t$ and $s \approx t$ is strictly maximal with respect to C. For example, any ground instance of the clause $x * i(x) \approx 1 \vee x \approx 0$ is reductive for $x * i(x) \Rightarrow 1$ in most reduction orderings $\succ$. A reductive clause may be viewed as a reductive conditional rewrite rule with both positive and negative conditions, such as $x \not\approx 0 \rightarrow x * i(x) \Rightarrow 1$.

[7] If R is a convergent ground rewrite system, the *equality Herbrand interpretation (induced by) R* is the set of ground equations which converge in R. From now on we shall usually identify any ground rewrite system with the interpretation it induces. Moreover, given an admissible ordering $\succ$ we need not distinguish between a ground equation and the rewrite rule it represents when oriented according to $\succ$.

We extend conditional rewriting to reductive rules with negative conditions by considering $u \not\approx v$ to be true if $u \approx v$ does *not* have a rewrite proof. Unfortunately, if the maximal term in a reductive clause occurs in more than one literal (which all have to be positive), it is not clear how to (recursively) evaluate these (negative) conditions. For example, suppose the two clauses $C = f(a) \approx a \vee f(a) \approx b$ and $D = a \approx b$ with $a \neq b$ are reductive for $f(a) \Rightarrow a$ and $a \Rightarrow b$, respectively. Then the rule $f(a) \Rightarrow a$ is applicable provided the negative condition $f(a) \not\approx b$ is true, that is, $f(a) \approx b$ has no rewrite proof. But since we have another (unconditional) rule $a \Rightarrow b$, there is a rewrite proof of $f(a) \approx b$, whenever the rule $f(a) \Rightarrow a$ is applicable! To avoid such paradoxical situations we derive alternative clauses to potentially problematic reductive clauses. For instance, if we apply equality factoring to C we obtain another clause $C' = a \not\approx b \vee f(a) \approx b$ that can be used to rewrite $f(a)$ to b. In the presence of D, the clause C' is equivalent to C, and the latter clause can be discarded.

If N is a set of ground clauses and C a ground clause, we denote by N_C the subset of all clauses D in N with $C \succ D$. If C is greater than all clauses in N, then evidently $N = N_C$. Let us denote by $\top$ a special clause that is assumed to be greater in $\succ$ than any regular clause. Then every set N of regular clauses can be written as a set N_C; for instance, $N = N_\top$. We define a mapping I that assigns to each clause set N_C a (convergent) rewrite system I_{N_C}. The definition is by induction over the clause ordering $\succ$ in that I_{N_C} is defined in terms of sets I_{N_D}, where $C \succ D$. More specifically, the set I_{N_C} is defined as the set of all ground rewrite rules $s \approx t$ for which there exists a clause $D = D' \vee s \approx t$ in N with $C \succ D$ such that (i) D is reductive for $s \Rightarrow t$, (ii) D is a counterexample for I_{N_D}, (iii) s is irreducible by I_{N_D}, and (iv) D' is a counterexample in $I_{N_D} \cup \{s \approx t\}$. If such a clause D exists, we also say that it is *productive*, and *produces* the equation (or rule) $s \approx t$ for I_{N_C}.[8]

A productive clause D is false in I_D, but true in $I_D \cup \{s \approx t\}$ and in I_C, for any $C \succ D$. Condition (iii) ensures that the sets I_C are left-reduced rewrite systems and, therefore, convergent. A productive clause D may be thought of as a reductive conditional rewrite rule with positive and negative conditions from $\neg D'$ (all of which are true in I_D). The maximal term s may occur in another (necessarily positive) equation $s \approx u$ in D', but by condition (iv) the truth value of any such equation must not depend on the truth value of $s \approx t$. (In other words, there are no side effects from designating $s \approx t$ to be true.) If C

[8] Since $D \prec C$, the sets I_{N_D} are well-defined by the induction hypothesis. The definition of I is also independent of the choice of N and C: if $N_C = N'_{C'}$, then $I_{N_C} = I_{N'_{C'}}$. We sometimes omit the index N and write I_C instead of I_{N_C}, if N is clear from the context.

and D are two clauses with $C \preceq D$, then $I_C \subseteq I_D$. The definition of the mapping I, and the properties of the clause ordering, ensure that if D is a productive clause in N, and C, D', and D'' are other clauses with $D'' \succ D' \succ D \succeq C$, then C is a counterexample for $I_{D'}$ if and only if it is a counterexample for $I_{D''}$.

Let us illustrate the above definition with an example. Suppose we have constants $a \succ b \succ c \succ d$. The following table lists clauses in descending order, along with any rewrite rules that are produced.

	C	produces	remarks
1	$a \not\approx b \vee a \approx d$		true in I_C
2	$a \not\approx d \vee b \approx c$		false in I_C and I
3	$c \not\approx d \vee a \approx c \vee b \approx c$	$a \approx c$	reductive and false in I_C
4	$c \approx d$	$c \approx d$	

All but the second clause are true in the candidate model $I = \{c \approx d, a \approx c\}$. There is a superposition inference between the third (productive) clause and this counterexample, which yields a smaller counterexample $c \not\approx d \vee c \not\approx d \vee b \approx c \vee b \approx c$.

When a counterexample $C[s]$ is reduced via some smaller clause $D \vee s \approx t$, then both $C[t]$ and $D \vee C[t]$ are smaller counterexamples than C. If the maximal terms s in a counterexample C is not reducible in this way, then it must occur more than once and a smaller counterexample can be obtained by (equality) factoring. In sum, we get:

THEOREM 6. *Let N be a set of ground clauses not containing the empty clause and C be the minimal counterexample in N for I_N. Then there exists an inference in S from C such that*

(i) the conclusion is a smaller counterexample for I_N than C; and

(ii) in case of a superposition inference, C is the main premise and the side premise is a productive clause.

The theorem indicates that the superposition calculus has the reduction property for counterexamples, and hence is refutationally complete for ground clauses. It can also be seen that all essential inferences have the minimal counterexample as their main premise, with side premises that are true in the candidate model. We next present a general notion of redundancy by specifying sufficient criteria characterizing inessential inferences and clauses that can not be part of essential inferences.

6.2. *Redundancy and Saturation*

A ground clause C is called *redundant* with respect to a set N of ground clauses if there exist clauses $C_1,\ldots,C_k$ in N such that $C_1,\ldots,C_k \models C$ and $C \succ C_i$. (Note that C need not be an element of N.) Let $\mathcal{R}(N)$ denote the set of redundant clauses with respect to N. A clause C that is redundant in N is entailed by $N_C \setminus \mathcal{R}(N)$, the set of all non-redundant clauses in N smaller than C; and any model of $N \setminus \mathcal{R}(N)$ is also a model of N. A redundant clause obviously can not be the minimal counterexample in N for any interpretation.

Observe that the side premises of an inference that reduces a counterexample must not be counterexamples themselves, whereas the conclusion should be a counterexample smaller than the main premise. This suggests the following definition. An inference with main premise C, side premises $C_1,\ldots,C_n$, and conclusion D is called *redundant* (with respect to N), either if $D \succeq C$, or else if there exist clauses $D_1,\ldots,D_k$ in N_C such that $D_1,\ldots,D_k,C_1,\ldots,C_n \models D$. By $\mathcal{R}_\Pi(N)$ we denote the set of redundant inferences in Π with respect to N. Note that an inference is redundant with respect to any set N for which the conclusion is in $N \cup \mathcal{R}(N)$.

Finally, we call N *saturated up to redundancy* with respect to Π whenever $\Pi(N \setminus \mathcal{R}(N)) \subseteq \mathcal{R}_\Pi(N)$, that is, if all inferences from non-redundant premises are redundant with respect to N.

THEOREM 7. *Let Π be sound and have the reduction property and let N be saturated up to redundancy with respect to Π. Then N is unsatisfiable if and only if it contains the empty clause.*

Proof. Suppose N does not contain the empty clause and let M be the set $N \setminus \mathcal{R}(N)$. Consider the interpretation I_M. If I_M is not a model of M, then M contains a minimal counterexample C for I_M. Since Π has the reduction property there is an inference from M with main premise C, side premises C_i, and a conclusion D, such that D is a smaller counterexample for I_M than C and the clauses C_i are true in I_M. By saturation, this inference is redundant. Thus, there are clauses D_j in N_C, such that D is entailed by all clauses C_i and D_j. We may also assume that the clauses D_j are non-redundant and true in I_M (for C is the minimal counterexample for I_M). But this implies that D is true in I_M, which is a contradiction. In sum, I_M is a model of M, and also of N.

Let us also point out that redundancy is preserved if one either adds clauses or deletes redundant clauses: (i) if $N \subseteq N'$ then $\mathcal{R}(N) \subseteq \mathcal{R}(N')$ and $\mathcal{R}_\Pi(N) \subseteq \mathcal{R}_\Pi(N')$ and (ii) if $N' \subseteq \mathcal{R}(N)$ then $\mathcal{R}(N) \subseteq \mathcal{R}(N \setminus N')$ and $\mathcal{R}_\Pi(N) \subseteq \mathcal{R}_\Pi(N \setminus N')$.

Deduction	$N \vdash N \cup M$	if $N \models M$
Deletion	$N \cup M \vdash N$	if $M \subseteq \mathcal{R}(N)$

Figure 3. *Theorem Proving Derivations* $\vdash$

6.3. *Theorem Proving Processes*

We next formalize the process of saturating a clause set up to redundancy. First, we define a one-step derivation relation $\vdash$ on clause sets as shown in figure 3. We have $N \vdash N'$ if N' is obtained from N either by adding logical consequences of the current clauses or by deleting (any number of) redundant clauses. In an actual theorem prover, some sound deduction system will be used to implement steps of the first kind, whereas steps of the second kind capture redundancy elimination techniques.

A (finite or countably infinite) sequence $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ is called a *(theorem proving) derivation*. The set $N_\infty = \bigcup_i \bigcap_{j \geq i} N_j$ of all *persisting clauses* is called the *limit* of the derivation. The sets of clauses N_i represent the successive states in the theorem proving process; the set N_∞ its result (which, in the case of an infinite derivation, is only obtained in the limit).

LEMMA 6. *If $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ is a derivation, then*

(i) $\mathcal{R}(N_i) \subseteq \mathcal{R}(\bigcup_j N_j) \subseteq \mathcal{R}(N_\infty)$ *and* $\mathcal{R}_\Pi(N_i) \subseteq \mathcal{R}_\Pi(\bigcup_j N_j) \subseteq \mathcal{R}_\Pi(N_\infty)$, *for all* $i \geq 0$, *and*

(ii) the set N_∞ is satisfiable if and only if N_0 is satisfiable.

The lemma states that redundancy and (un)satisfiability are preserved throughout the theorem proving process, and carry over to the limit set N_∞. Thus, if a clause or inference is redundant at some point in a derivation, it will be redundant at any later point. This implies that redundancy elimination criteria can be implemented in a don't-care nondeterministic fashion.

A derivation $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ is called *fair* (with respect to Π) if every inference in Π with non-redundant premises in N_∞ is redundant in $\bigcup_j N_j$, i.e., $\Pi(N_\infty \setminus \mathcal{R}(N_\infty)) \subseteq \mathcal{R}_\Pi(\bigcup_j N_j)$. Fairness essentially requires that no inference in Π from non-redundant persisting formulas be delayed indefinitely.

LEMMA 7. *A derivation is fair with respect to* Π *if the conclusion of every non-redundant inference in* Π *from non-redundant formulas in* N_∞ *is an element of, or is redundant in,* $\bigcup_j N_j$, *i.e.,*

$$\Gamma(\Pi(N_\infty \setminus \mathcal{R}(N_\infty) \setminus \mathcal{R}_\Pi(N_\infty \setminus \mathcal{R}(N_\infty))) \subseteq (\bigcup_j N_j \cup \mathcal{R}(\bigcup_j N_j)).$$

In other words, a fair derivation can be constructed by exhaustively applying inferences to persisting clauses.

LEMMA 8. *If a derivation is fair with respect to* Π *then its limit* N_∞ *is saturated up to redundancy with respect to* Π.

THEOREM 8. *If* Π *has the reduction property for counterexamples and* $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ *is a fair theorem proving derivation with respect to* Π, *then* N_0 *is unsatisfiable if and only if* N_∞ *(and, hence,* $\bigcup_j N_j$*) contains the empty clause.*

6.4. *Lifting*

The notions of redundancy, derivation and saturation need to be generalized for general clauses with variables. If M is a set of clauses, we denote by $g(M)$ the set of all its ground instances. Furthermore, we say that a clause C is *redundant* with respect to M if $g(C) \subseteq \mathcal{R}(g(M))$; that is, all ground instances of C are redundant with respect to $g(M)$. Similarly, an inference π in Π is called redundant with respect to M if $g(\pi) \subseteq \mathcal{R}_\Pi(g(M))$, where $g(\pi)$ consists of all *ground instances* of π, that is, all ground inferences in Π obtained from π by uniformly instantiating all premises and the conclusion with the same substitution. Note that instantiating the premises and the conclusion of an inference in Π need not necessarily yield a ground inference in Π, as the side conditions, (in particular, the ordering constraints) need not be satisfied for all instantiated formulas.

Theorem proving derivations can now be defined for general clauses, based on this generalized notion of redundancy. For any such derivation $M_0 \vdash M_1 \vdash M_2 \vdash \cdots$ the corresponding sequence of sets of ground clauses $g(M_0), g(M_1), \ldots$ is also a theorem proving derivation $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ where N_i denotes the set $g(M_i)$. We also need to show that the ground derivation is fair, provided the general derivation is fair. Two conditions are required. First, any persistent ground clause in N_∞ should be an instance of a persistent clause in M_∞. This is indeed the case, as the elimination of clauses is based on the presence of smaller clauses in a well-founded ordering.[9] Sec-

[9] There is a subtle issue related to the fact that a ground clause may be an instance of several non-ground clauses. One might want to admit a more liberal notion of redundancy on the non-ground level and allow clause deletion based on non-strict subsumption. But then a ground instance $C = D\sigma = D'\sigma'$ may persist, even though neither D nor D' persists. This might happen, for instance, if D is an element of all sets M_{4i} and D' and element of all sets M_{4i+2} (but not vice versa). This problem can be solved by using additional syntax such as clause numberings (then non-strict "forward subsumption" can also be covered by redundancy), or by more restrictive definitions of fairness on the non-ground level.

ondly, non-redundant ground inferences in Π must be *liftable* in the sense that they can be obtained as ground instances of general inferences in Π.

LEMMA 9. *Non-redundant superposition inferences are liftable.*

Proof. Let M be a set of clauses. The problematic ground inferences are superpositions from $g(M)$ with a main premise $C[s]$ and side premise $D \vee s \approx t$ such that $C[s]$ is an instance of some non-ground clause $C'[x]$ in M by a substitution σ, where $x\sigma = u[s]$. That is, the subterm replacement occurs at or below the position of a variable x in C'. Let $\tau = \sigma[u[t]/x]$ be a "more reduced" substitution than σ. Then $C'\tau$ is also a clause in $g(M)$, but smaller than C, and $\{C'\tau, D \vee s \approx t\} \models D \vee C[t]$.[10] That is, the ground inference is redundant in $g(M)$.

Summarizing the above results we obtain:

THEOREM 9. (Bachmair and Ganzinger, 1990). *(i) Let N be a set of ground clauses that is saturated up to redundancy with respect to $\mathsf{S}^{\succ}$. Then N is unsatisfiable if and only if N contains the empty clause. (ii) If M is the limit of a fair theorem proving derivation with respect to $\mathsf{S}^{\succ}$ then $g(M)$ is saturated up to redundancy with respect to $\mathsf{S}^{\succ}$.*

Note that redundancy can only be an approximation of the concept of non-essential clauses and inferences, so that some inferences are neither redundant nor essential. The reason is that essential inferences need not stay essential, and non-essential inferences may become essential, during a theorem proving derivation. Redundancy, on the other hand, is a property that it is preserved by each derivation step.

Ideas related to the notion of candidate model can be found in Brand's original proof of theorem 1 and in the work of Zhang (1988). The definitions presented here are almost identical to (Bachmair and Ganzinger, 1990), though there the proofs are technically more difficult in that they deal with counterexample reduction and redundancy simultaneously. A later paper by Pais and Peterson (1991) contains a similar proof of the refutational completeness of superposition, but without any mention of redundancy.

[10] Observe that this reasoning makes essential use of the symmetry of equality (when C' has more than one occurrence of x) and, therefore, cannot be applied to chaining calculi for non-symmetric transitive relations.

6.5. *Simplification*

An important application of redundancy is the use of logical equivalences for simplification of clauses. For instance, if $N, C' \models C$ and $N, C \models C'$ and $C \succ C'$, then there is a derivation

$$N \cup \{C\} \vdash N \cup \{C, C'\} \vdash N \cup \{C'\}$$

consisting of a deduction step (C' is a logical consequence of $N \cup \{C\}$) followed by a deletion step (C is rendered redundant by C'). This suggest a derived inference rule for theorem proving derivations:

Simplification

$$N \cup \{C\} \vdash N \cup \{C'\} \qquad \text{if } N, C' \models C \text{ and } N, C \models C' \text{ and } C \succ C'.$$

For example, reduction by oriented instances of equations

$$N \cup \{s \approx t, C[s\sigma]\} \vdash N \cup \{s \approx t, C[t\sigma]\}$$

is a simplification step provided $s\sigma \succ t\sigma$ and $C \succ (s \approx t)\sigma$. More sophisticated reduction techniques employ general reductive clauses as conditional rewrite rules.

Subsumption and tautology deletion are standard redundancy elimination techniques:

Tautology elimination

$$N \cup \{C\} \vdash N \quad \text{if } \models C$$

Subsumption

$$N \cup \{C, D\} \vdash N \cup \{C\} \quad \text{if } C\sigma \subset D$$

Both represent deletion steps.

6.6. *Strict Superposition*

If we delete equality factoring from the superposition calculus, we obtain the strict superposition calculus SS. This calculus is not compatible with tautology elimination (Bachmair and Ganzinger, 1990) and hence can not have the reduction property for counterexamples. But if we relax the notion of counterexamples, and slightly modify the proof techniques by using a suitable notion of "direct provability" rather than (semantic) validity, we may establish the completeness of strict superposition in a similar way as for superposition.

A *direct rewrite proof* of an equation $s \approx t$ (with respect to a rewrite system R) is a rewrite proof that uses only equations $u \approx v$ that are smaller than or equal to $s \approx t$.

Equality factoring has been designed to reduce counterexamples in which the maximal term is positive and occurs more than once. We dispense with equality factoring, but allow *weak counterexamples* (for a rewrite system R), that is, clauses for which all negative equations have a rewrite proof, but no positive equation has a *direct* rewrite proof. If we replace "counterexample" by "weak counterexample" in the definition of I, condition (iv) can be removed.

THEOREM 10. (Bachmair and Ganzinger, 1997a). *Let N be a set of ground clauses not containing the empty clause and C be the minimal weak counterexample in N for I_N. Then there exists an inference in* SS *from C such that*

(i) the conclusion is a smaller weak counterexample for I_N than C; and

(ii) in case of a superposition inference, C is the main premise and the side premise is a productive clause.

The theorem indicates that the superposition calculus has the *reduction property for weak counterexamples*, and hence is refutationally complete for ground clauses. The notion of redundancy based on weak counterexamples is also weaker than the standard notion and does not cover all cases of tautology deletion. For instance, the tautology $a \approx b \rightarrow a \approx b$ with $a \succ b$ is a weak counterexample for the rewrite system $\{a \approx c, c \approx b\}$ if $c \succ b$. Most other simplification techniques are, by and large, covered by weak redundancy (Bachmair and Ganzinger, 1997a). Unfortunately, not all superposition inferences at or below variables are redundant when redundancy is based on weak counterexamples. Therefore, lifting requires the techniques of basic superposition to be discussed in the next section.

A weaker variant of strict superposition was introduced by Zhang and Kapur (1988). The completeness of that proof calculus had been an open problem that is settled by the above result.

7. Subterm Selection Strategies

Consider a superposition inference between two Robbins equations:

$$\frac{n(n(n(x')+y')+n(x'+y')) \approx y' \quad n(\underline{n(n(x)+y)}+n(x+y)) \approx y}{n(y'+n(\boxed{n(x')+y'}+\boxed{n(x'+y')})) \approx \boxed{n(x'+y')}}$$

(McCune, 1997). The underlined subterm is replaced after unification, while the boxed subterms are introduced by instantiation of variables in the second premise. The first premise can again be superposed on (either of) the boxed subterms, producing yet larger subterms that allow yet more similar superpositions, and so on. When the "substitution parts" of a clause grow very quickly, the proof search may easily get out of hand. Subterm selection strategies are designed to better control such explosive growth. The *basic restriction* selects positions in a clause in order to block further superpositions on the corresponding subterms. This restriction complements, on the term level, the effect of literal selection functions and represents a robust answer to a related research problem posed by Wos (1988). It has been a key part in the solution of the Robbins problem (McCune, 1997).

Brand's modification method provides some insight for subterm selection: when paramodulation is used to simulate resolution on transformed clauses, it very rarely replaces subterms introduced by unification in previous inference steps. Also, there are rewriting strategies, such as innermost rewriting, that considerably restrict the number of subterm position eligible for replacements. Some of these techniques have been applied to deduction in the context of narrowing, see Hullot (1980).

7.1. *Marked Clauses*

The basic restriction can be formalized by explicitly *marking* subterm positions in literals and clauses. Any position at or below a marked position is called a *substitution position*. The subterms at substitution positions form the *substitution part* of a clause (or literal). We shall prohibit superposition inferences with subterms at substitution positions. Since superposition into variables is known to be redundant, we generally require that all variables belong to the substitution part of a clause. In examples we represent the substitution part by enclosing its maximal subterms in boxes (though for convenience we usually omit boxes around variables as these are in the substitution part by default), such as in $f(\boxed{fx}) \approx h(x) \vee g(f(x)) \approx \boxed{a}$. A position p in a literal L of a clause C is called an *induced substitution position* if the subterm of L at p also occurs at a substitution position in a literal L' in C, with $L' \succeq L$. For example, if the first literal in the above clause is greater than the second (in the given ordering), then the second (un-boxed) occurrence of $f(x)$ is at an induced substitution position. We shall see that basic restrictions can be propagated to induced substitution positions. (Note that substitution positions, by definition, are also induced substitution positions.) Marking schemes have no semantic significance; they are a syntactic mechanism for refining inference rules.

A central notion in reasoning about basic inference systems is that of a reduced clause. We say that a literal $L[s]_p$ is *order-reducible* (at position p) by an equation $s \approx t$, if $s \succ t$ and $L \succ s \approx t$. Thus, a negative literal is order-reducible by $s \approx t$ if it contains s as a subterm. A positive equation $u \approx v$ is always order-reducible if s is a subterm of (the smaller term) v or a proper subterm of (the larger term) u. If s is identical to u, the two equations need to be compared to determine order-reducibility. A literal is *order-reducible by* R if it is order-reducible by some equation in R. A clause is order-reducible if one of its literals is order-reducible. Finally, a (marked) clause C is said to be *reduced* with respect to R if it is not order-reducible at any substitution position.

For example, if we use an admissible extension[11] of a lexicographic path ordering with precedence $f \succ a \succ b \succ c$, then $(f(\boxed{a}) \approx b \ \vee \ f(\boxed{f(a)}) \approx a)$ is reduced with respect to the system $\{fa \approx a\}$, but $\boxed{f(a)} \not\approx c$ is not.

7.2. *Basic Superposition*

The inference rules of the *basic superposition* calculus B are the same as for superposition S (cf. figure 2), but applied to marked clauses and with the variable restriction (v) sharpened to

(v') s' occurs not at an *induced* substitution position.

It is to be understood that in forming the disjunction $C \vee D$ of two marked clauses C and D, all markings are inherited. Similarly, we assume that in any instance $C\sigma$ the same positions are marked as in C. New subterm positions created by the application of the substitution σ are below a variable position in C, and hence automatically belong to the substitution part of $C\sigma$ (as expected). Finally, if t is a marked term and L a marked literal, the literal $L[t]$ is assumed to be marked like t for positions within the indicated occurrence of t in $L[t]$, and like L for other positions.

The negative basic superposition inference

$$\frac{Q(\boxed{ga}) \vee f(hz,z) \approx j(z,\boxed{a}) \qquad \neg P(f(x,gy)) \vee k(x,gy) \approx \boxed{hy}}{\neg P(j(\boxed{gy},\boxed{a})) \vee Q(\boxed{ga}) \vee k(\boxed{hgy},\underline{gy}) \approx \boxed{hy}}$$

illustrates the inheritance of markings from premises to the conclusion. New subterms introduced by instantiation also belong to the substitution part. The

[11] Let us mention that in defining (admissible) orderings, we may take advantage of the information provided by marking when comparing terms or literals.

underlined occurrence of $g(y)$ is at an induced substitution position, under certain assumptions about the ordering. All boxed (and possibly all underlined) subterms are blocked from subsequent inferences.

In investigating the reduction property for B we consider candidate models for sets of ground clauses as defined before, though clauses and (produced) equations will be marked.

A set of ground clauses N is said to be *reduced* if any clause C in N is reduced by I_C. Observe that if C is reduced by I_C it is also reduced by I_N. A reduced subset M of N is said to be *maximal* if it contains all clauses in N that are reduced with respect to I_M.

PROPOSITION 2. *Any set of ground clauses contains a maximal reduced subset.*

Proof. Define $M = M(N)$ as the set of all clauses C in N that are reduced with respect to $I_{M(N_C)}$, where $M(N_C)$ denotes, by induction hypothesis, the maximal reduced subset of N_C.

The following theorem applies to both the strict and non-strict version of basic superposition. As for non-basic superposition, in the strict calculus equality factoring is not needed, at the cost of having to reduce weak counterexamples.

THEOREM 11. *Let N be a reduced set of ground clauses not containing the empty clause and C be the minimal [weak] counterexample in N for I_N. Then there exists an inference in B from C such that*

(i) the conclusion is a smaller [weak] counterexample for I_N than C, and is also reduced with respect to I_N; and

(ii) if the inference is a superposition inference then C is its negative premise, and the positive premise is a productive clause.

The theorem states that for *reduced sets of clauses* B has the reduction property for [weak] counterexamples. The smaller counterexample is again a reduced clause, which is important for effective saturation.

A set of ground clauses N is called *schematic* if it consists of all ground instances $C\sigma$ of clauses C in some set K (of non-ground clauses) in which only variable positions are marked.[12] Schematic sets N of clauses allow one to reduce the substitution part of any clause in N to obtain an equivalent (with respect to I_N) reduced clause in N. The reduction property of B implies its refutational completeness for schematic sets of clauses.

[12] There may be multiple occurrences of the same clause in N, but with different markings.

THEOREM 12. (Bachmair et al., 1992;Nieuwenhuis and Rubio, 1992a). *Let N be a set of ground clauses that is closed with respect to* B. *Moreover, assume that N contains a schematic subset N' such that every clause in $N \setminus N'$ is a logical consequence of N'. Then N either contains the empty clause, or else is satisfiable.*

This theorem can be applied whenever N is obtained as the closure of some initial schematic clause set (as represented by an arbitrary set of non-ground clauses) under B. The lifting of this completeness result to the non-ground case is not difficult. Positions in the substitution part of a non-ground clause become substitution positions in ground instances and hence are blocked from further ground inferences.

7.3. *Basic Superposition for Clauses with Constraints*

A generalization of the clause marking schema leads us to clauses with constraints. In logic programming, constraints have been proposed by Jaffar and Lassez (1987) as a logical means for handling primitive theories, though in theorem proving one encounters both object- and meta-level constraints, see (Huet, 1972; Kirchner et al., 1990).

Marked ground clauses can be represented by constrained (non-ground) clauses with simple equational constraints. For example, the constrained clause $(x \not\approx c) \cdot (x = f(a))$ represents the ground clause $f(a) \not\approx c$ (with the constraint defining a ground substitution). In general, a constrained clause represents all ground clauses obtained from its clause part by instantiation with a substitution solving its constraints.

Constraints serve two different purposes in theorem proving. Equality constraints abstractly define ground substitutions for a clause, and thus also determine the substitution part of a clause that may be blocked from inferences. Ordering constraints, on the other hand, allow for a better approximation of the side conditions of inferences. In essence, we may move ordering constraints from the meta- to the object level and carry them along throughout the derivation process. For example, in the inference

$$\frac{x+y \approx y+x \quad u+f(z) \approx f(z)+z}{(f(z)+u \approx f(z)+z) \cdot (u \succ f(z))}$$

the combined equality and ordering constraint

$$x = u \ \wedge \ y = f(z) \ \wedge \ u+f(z) \succ f(z)+u \ \wedge \ u+f(z) \succ f(z)+z$$

when projected to the variables u and z appearing in the conclusion, is equivalent to $u \succ f(z)$. The unit clause

$$(f(z)+u \approx f(z)+z) \cdot (u \succ f(z))$$

represents all ground clauses of the form $f(\boxed{t}) + \boxed{s} \approx f(\boxed{t}) + \boxed{t}$, where s and t are ground terms with $s \succ f(t)$, and the root positions of substituted terms are marked as indicated. In standard superposition, the satisfiability of the constraint is verified once, and the unconstrained equation $f(z) + u \approx f(z) + z$ is generated. There are additional superpositions of the latter unconstrained equation with commutativity, whereas all inferences of the constrained equation with commutativity result in unsatisfiable ordering constraints (because the left-hand side of the equation is already constrained to be maximal, with the larger term s as the second argument) and hence are redundant.

Formally, a *constrained clause* is a pair $C \cdot \gamma$ of a clause C and a constraint γ. *Constraints* are built from boolean connectives and atomic constraints of the form $E = E'$, $E \succeq E'$, or $E \succ E'$, where E and E' are expressions (terms, literals, or clauses). A ground substitution σ is a *solution* of an atomic constraint if the indicated relation holds between $E\sigma$ and $E'\sigma$, where $=$ is interpreted as syntactic equality and $\succ$ refers to the given ordering. Non-atomic constraints are evaluated on the basis of the truth values of their atomic constraints. A clause $C\sigma$ is said to be a *(ground) instance* of a constrained clause $C \cdot \gamma$ if σ is a solution of the constraint γ and all positions that correspond to variable positions in C are marked.

If a constraint γ is equivalent to $\top$, then $C \cdot \gamma$ can be identified with C and represents the set of all ground instances $C\sigma$. We also speak of an *unconstrained clause* in this case. A clause with an unsatisfiable constraint represents a tautology. A constrained clause $C \cdot \gamma$ is called a *contradiction* if C is the empty clause, but γ is satisfiable. For theorem proving applications it is usually not necessary, though, to test the satisfiability of arbitrary constraints in order to determine whether a constrained clause is a contradiction. Ordering constraints are kept strictly separate from the equality constraints, and may always be removed without affecting the soundness of the process. Thus, to detect contradictions, we only need to be able to decide the satisfiability of equality constraints, which is straightforward for the syntactic identity relation. (For many orderings, such as lexicographic path orderings, the satisfiability of arbitrary constraints is also decidable, see (Comon, 1990; Nieuwenhuis, 1993).)

Figure 4 shows a version of positive basic superposition with constraints. Other inference rules are extended to constrained clauses in a similar way. Note that the conclusion inherits all constraints from the premises and is further constrained by an equality constraint $s = s'$ (expressing the unifiability of the two terms) and by ordering constraints. The only meta-level constraint remaining is that s' should not be a variable.

Positive basic superposition with constraints

$$\frac{(C \vee s \approx t) \cdot \gamma \quad (D \vee w[s'] \approx v) \cdot \delta}{(C \vee D \vee w[t] \approx v) \cdot (\gamma \wedge \delta \wedge s = s' \wedge \omega)}$$

where s' is not a variable and where ω is the ordering constraint

$$(s \approx t) \succ C \ \wedge \ (w \approx v) \succ D \ \wedge \ s \succ t \ \wedge \ w \succ v \ \wedge \ (w \approx v) \succ (s \approx t).$$

Figure 4. *Basic superposition for constrained clauses* B

LEMMA 10. (Lifting Lemma for Basic Superposition with Constraints). *Let N be a set of constrained clauses. Any inference in* B *from ground instances of N is the ground instance of some inference in* B *from N.*

THEOREM 13. (Nieuwenhuis and Rubio, 1992b). *Let N be the closure under* B *of a set of unconstrained clauses. Then N is satisfiable if and only if it contains no contradiction.*

Proof. Let N be the closure of a set M of unconstrained clauses under B. The set of ground instances K of N satisfies the requirements of theorem 12, as the ground instances of M form a schematic subset. The result then follows by lifting.

A related result with ordering constraints at the meta-level was obtained in Bachmair et al., 1992. Nieuwenhuis and Rubio (1992b, 1995) also discuss extensions of the theorem to sets of clauses with certain kinds of initial constraints. Basically, one can also admit initial constraints if their solutions are closed under reduction. Earlier work on the subject includes Degtyarev (1979) who has stated the completeness of a certain strategy of basic paramodulation.

The issue of redundancy is more complicated in the context of basic superposition, because the reduction property for counterexamples holds only for reduced sets of clauses. A clause set K, by definition, is reduced with respect to (reduced) rewrite system I_K. A suitable notion of redundancy that is compatible with deduction and deletion (of redundant clauses), essentially singles out clauses that can *not* be a minimal reduced counterexample with respect to *any* reduced rewrite system. As a consequence, the usual simplification methods (reduction, subsumption) need to be somewhat restricted. On the other hand, a clause can be eliminated if its substitution part is reducible by another equation in the system. We refer to (Bachmair et al., 1995; Nieuwenhuis and Rubio, 1995) for a detailed exposition of the subject.

In (Bachmair et al., 1995) there is also a discussion of variable abstraction techniques that allow one to expand the substitution part of clauses. Redex orderings can be used to mark positions in the conclusion of a superposition inference that precede (in the redex ordering) the position at which the inference takes place. If basic paramodulation into the smaller sides of positive equations is permitted, then the replacement term in a superposition inference can be put in the substitution part also.

We already mentioned that monotonicity axioms are not needed for Skolem functions. The completeness of basic superposition also confirms a conjecture by (McCune, 1990), that superposition below a Skolem function is not required: since Skolem functions occur only with variable arguments in initial clauses, any term below a Skolem function in a derived clause is in the substitution part.

8. Summary

We have described superposition and its variants, calculi that in one form or another are implemented in virtually all state-of-the-art theorem provers for first-order clause logic with equality. Superposition represents the combination of two lines of research that started about thirty years ago. It is a refinement of the paramodulation calculus that was introduced by Robinson and Wos (1969), but also essentially draws on term rewriting techniques that ultimately derive from work by Knuth and Bendix (1970). Theoretical research on this topic has been complemented by extensive experimental and implementation work that has produced a number of powerful equational reasoning systems. A notable success has been the proof of a long-standing open problem, the Robbins conjecture, by (McCune, 1997).

The performance of superposition- and completion-based provers often depends on additional improvements that we have not been able to include in this survey. For instance, certain standard axioms or theories can be built directly into an inference mechanism. An example is the associative-commutative (AC) variant of completion that was initially proposed by Peterson and Stickel (1981) and later generalized by Jouannaud and Kirchner (1986) and Bachmair and Dershowitz (1989). Similar techniques have been applied to ordered paramodulation (Paul, 1992; Rusinowitch and Vigneron, 1991) and to superposition (Wertz, 1992; Bachmair and Ganzinger, 1995). In order to be able to apply these completeness results, ne needs to be able to construct a complete *AC-compatible* reduction ordering for the given signature. (Narendran and Rusinowitch, 1996) and (Rubio and Nieuwenhuis, 1993) have presented constructions which show that such orderings in fact do exist for all signatures, even if they contain more than one AC-symbol.

Typically, it is the unification algorithm that needs to be generalized to accommodate the additional axioms. The drawback is that most general unifiers need not be unique any more, and that each unification problem may generate a (potentially large) number of unifiers. In the case of associativity and commutativity, there are doubly exponentially many unifiers in the size of the terms to be unified (Kapur and Narendran, 1992). Fortunately, the explicit computation of unifiers can be avoided, if constraints are used to specify the requisite unification problems for an inference. Then it suffices to check whether the constraints are satisfiable, which in the case of associativity and commutativity is "only" an NP-complete problem, see (Kapur and Narendran, 1992). For a detailed description of associative-commutative superposition calculi with equality constraints see Nieuwenhuis and Rubio (1994) and Vigneron (1994). Constraints can also be used to represent implicit or explicit types of variables such that type inference is integrated into proof search, cf. Weidenbach (this handbook, I.2.9).

Other recent work focuses on developing specialized (basic and non-basic) completion and superposition calculi for specific commutative algebraic theories, such as groups and rings. The motivation is that the richer algebraic structure may allow for stronger simplification techniques and more optimized deduction rules; see the papers by Marché (1994), Stuber (this handbook, III.1.2), and (Waldmann, 1997) for work in this direction.

Leo Bachmair, Department of Computer Science, SUNY, Stony Brook, NY.
Supported in part by the National Science Foundation under grants INT-9314412 and CCR-9510072.
Harald Ganzinger, MPI Informatik, Saarbrücken, Germany.
Supported in part by German Science Foundation (Deutsche Forschungsgemeinschaft) under grants Ga 261/7-1 and Ga 261/8-1, and by the ESPRIT Basic Research Working Group 22457 (Construction of Computational Logics II).

References

Bachmair, L.: 1989, 'Proof normalization for resolution and paramodulation'. In: *Proc. 3rd Int Conf. Rewriting Techniques and Applications*, Vol. 355 of *Lecture Notes in Computer Science*. Berlin, pp. 15–28.

Bachmair, L.: 1991, *Canonical equational proofs*. Boston: Birkhäuser.

Bachmair, L. and N. Dershowitz: 1989, 'Completion for rewriting modulo a congruence'. *Theoretical Computer Science* **67**, 173–201.

Bachmair, L., N. Dershowitz, and J. Hsiang: 1986, 'Orderings for Equational Proofs'. In: *Proceedings of the First IEEE Symposium on Logic in Computer Science (Cambridge, MA)*. pp. 346–357.

Bachmair, L., N. Dershowitz, and D. Plaisted: 1989, 'Completion without failure'. In: H. Aït-Kaci and M. Nivat (eds.): *Resolution of Equations in Algebraic Structures, vol. 2*. pp. 1–30.

Bachmair, L. and H. Ganzinger: 1990, 'On Restrictions of Ordered Paramodulation with Simplification'. In: M. Stickel (ed.): *Proc. 10th Int. Conf. on Automated Deduction, Kaiserslautern*, Vol. 449 of *Lecture Notes in Computer Science*. Berlin, pp. 427–441.

Bachmair, L. and H. Ganzinger: 1995, 'Associative-Commutative Superposition'. In: N. Dershowitz and N. Lindenstrauss (eds.): *Proc. 4th Int'l Workshop on Conditional and Typed Rewrite Systems,* Jerusalem, Vol. 968 of *Lecture Notes in Computer Science*. Berlin, pp. 1–14.

Bachmair, L. and H. Ganzinger: 1997a, 'Strict Basic Superposition and Chaining'. Research Report MPI-I-97-2-011, Max-Planck-Institut für Informatik, Saarbrücken.

Bachmair, L. and H. Ganzinger: 1997b, 'A Theory of Resolution'. Research Report MPI-I-97-2-005, Max-Planck-Institut für Informatik, Saarbrücken. To appear in the Handbook of Automated Reasoning.

Bachmair, L., H. Ganzinger, C. Lynch, and W. Snyder: 1992, 'Basic Paramodulation and Superposition'. In: D. Kapur (ed.): *Automated Deduction — CADE'11*, Vol. 607 of *Lecture Notes in Computer Science*. Berlin, pp. 462–476.

Bachmair, L., H. Ganzinger, C. Lynch, and W. Snyder: 1995, 'Basic Paramodulation'. *Information and Computation* **121**(2), 172–192.

Bachmair, L., H. Ganzinger, and A. Voronkov: 1997, 'Elimination of Equality via Transformation with Ordering Constraints'. Research Report MPI-I-97-2-012, Max-Planck-Institut für Informatik, Saarbrücken.

Brand, D.: 1975, 'Proving theorems with the modification method'. *SIAM J. Computing* **4**, 412–430.

Comon, H.: 1990, 'Solving symbolic ordering constraints'. *International Journal on Foundations of Computer Science* **1**(4).

Comon, H., P. Narendran, R. Nieuwenhuis, and M. Rusinowitch: 1998, 'Decision Problems in Ordered Rewriting'. In: *Proc. 13th IEEE Conference on Logic in Computer Science*. To appear.

Davis, M. and H. Putnam: 1960, 'A Computing Procedure for Quantification Theory'. *J. Association for Computing Machinery* **7**, 201–215.

Degtyarev, A.: 1979, 'The Strategy of Monotone Paramodulation (in Russian)'. In: *Fifth Soviet All-Union Conference on Mathematical Logic*. Novosibirsk, p. 39.

Degtyarev, A. and A. Voronkov: 1996a, 'Equality Elimination for the Tableau Method'. In: J. Calmet and C. Limongelli (eds.): *Design and Implementation of*

Symbolic Computation Systems. International Symposium, DISCO'96, Vol. 1128 of *Lecture Notes in Computer Science*. Karlsruhe, Germany, pp. 46–60.

Degtyarev, A. and A. Voronkov: 1996b, 'What You Always Wanted to Know About Rigid E-Unification'. In: J. Alferes, L. Pereira, and E. Orlowska (eds.): *Logics in Artificial Intelligence. European Workshop, JELIA'96*, Vol. 1126 of *Lecture Notes in Artificial Intelligence*. Évora, Portugal, pp. 50–69.

Dershowitz, N.: 1987, 'Termination of Rewriting'. *J. Symbolic Computation* **3**(1), 69–115.

Gallier, J. H.: 1986, *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Harper and Row.

Ganzinger, H.: 1991, 'A completion procedure for conditional equations'. *J. Symbolic Computation* **11**, 51–81.

Gilmore, P. C.: 1960, 'A proof method for quantification theory'. *IBM J. Res. Develop.* **4**, 28–35.

Hsiang, J. and M. Rusinowitch: 1987, 'On Word Problems in Equational Theories'. In: *Proc. 14th ICALP*, Vol. 267 of *Lecture Notes in Computer Science*. Berlin, pp. 54–71.

Hsiang, J. and M. Rusinowitch: 1991, 'Proving refutational completeness of theorem proving strategies: The transfinite semantic tree method'. *J. Association for Computing Machinery* **38**(3), 559–587.

Huet, G.: 1972, 'Constrained Resolution: A Complete Method for Higher-Order Logic'. Ph.D. thesis, Case Western Reserve University.

Huet, G.: 1981, 'A complete proof of correctness of the Knuth and Bendix completion algorithm'. *J. Computer and System Sciences* **23**, 11–21.

Hullot, J.-M.: 1980, 'Canonical Forms and Unification'. In: *Proc. 5th Conf. on Automated Deduction, Les Arcs*, Vol. 87 of *Lecture Notes in Computer Science*.

Ibens, O. and R. Letz: 1997, 'Subgoal alternation in model elimination'. In: D. Galmiche (ed.): *Automated Reasoning with Analytic Tableaux and Related Methods*, Vol. 1227 of *Lecture Notes in Artificial Intelligence*. pp. 201–215.

Jaffar, J. and J.-L. Lassez: 1987, 'Constraint logic programming'. In: M. J. O'Donnell (ed.): *Proc. 14th Annual ACM Symposium on Principles of Programming Languages*. Munich, FRG, pp. 111–119.

Jouannaud, J.-P. and H. Kirchner: 1986, 'Completion of a set of rules modulo a set of equations'. *SIAM J. Computing* **15**, 1155–1194.

Jouannaud, J.-P. and B. Waldmann: 1986, 'Reductive Conditional Term Rewriting Systems'. In: *Proc. Third IFIP Working Conference on Formal Description of Programming Concepts*. Ebberup, Denmark, pp. 223–244.

Kaplan, S.: 1987, 'Simplifying Conditional Term Rewriting Systems: Unification, Termination and Confluence'. *J. Symbolic Computation* **4**(3).

Kapur, D.: 1997, 'Shostak's congruence closure as completion'. In: H. Comon (ed.): *Rewriting Techniques and Applications*, Vol. 1232 of *Lecture Notes in Computer Science*. Sitges, Spain, pp. 23–37.

Kapur, D. and P. Narendran: 1992, 'Complexity of Unification Problems with Associative-Commutative Operators'. *J. Automated Reasoning* **9**(2), 261–288.

Kirchner, C., H. Kirchner, and M. Rusinowitch: 1990, 'Deduction with symbolic constraints'. *Revue Française d'Intelligence Artificielle* **4**(3), 9–52. Special issue on automatic deduction.

Knuth, D. and P. Bendix: 1970, 'Simple word problems in universal algebras'. In: J. Leech (ed.): *Computational Problems in Abstract Algebra*. Oxford: Pergamon Press, pp. 263–297.

Kounalis, E. and M. Rusinowitch: 1987, 'On word problems in Horn logic'. In: B. Caviness (ed.): *Proc. First Int. Workshop on Conditional Term Rewriting*, Vol. 204 of *Lecture Notes in Computer Science*. Berlin, pp. 390–399.

Lankford, D.: 1975, 'Canonical inference'. Technical Report ATP-32, Dept. of Mathematics and Computer Science, University of Texas, Austin.

Marché, C.: 1994, 'Normalized rewriting and normalized completion'. In: *Proc. 9th IEEE Symposium on Logic in Computer Science*. pp. 394–405.

Martin, U. and T. Nipkow: 1990, 'Ordered Rewriting and Confluence'. In: M. Stickel (ed.): *Proc. 10th Int. Conf. on Automated Deduction, Kaiserslautern*, Vol. 449 of *Lecture Notes in Computer Science*. Berlin, pp. 366–380.

McCune, W.: 1990, 'Skolem Functions and Equality in Automated Deduction'. In: T. Dietterich and W. Swartout (eds.): *Proceedings of the 8th National Conference on Artificial Intelligence*. pp. 246–251.

McCune, W.: 1997, 'Solution of the Robbins Problem'. *J. Automated Reasoning* **19**(3), 263–276.

Moser, M., C. Lynch, and J. Steinbach: 1995, 'Model Elimination with Basic Ordered Paramodulation'. Technical Report AR-95-11, Fakultät für Informatik, Technische Universität München, München.

Moser, M. and J. Steinbach: 1997, 'STE-Modification Revisited'. Technical Report AR-97-03, Technische Universität München.

Narendran, P. and M. Rusinowitch: 1996, 'Any Ground Associative-Commutative Theory has a Finite Canonical System'. *J. Automated Reasoning* **17**(1), 131–143.

Nieuwenhuis, R.: 1993, 'Simple LPO constraint solving methods'. *Information Processing Letters* **47**(2), 65–69.

Nieuwenhuis, R. and A. Rubio: 1992a, 'Basic superposition is complete'. In: *ESOP'92*, Vol. 582 of *Lecture Notes in Computer Science*. Berlin, pp. 371–389.

Nieuwenhuis, R. and A. Rubio: 1992b, 'Theorem proving with ordering constrained clauses'. In: *Automated Deduction — CADE'11*, Vol. 607 of *Lecture Notes in Computer Science*. Berlin, pp. 477–491.

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

Nieuwenhuis, R. and A. Rubio: 1995, 'Theorem Proving with Ordering and Equality Constrained Clauses'. *J. Symbolic Computation* **19**(4), 321–352.

Pais, J. and G. Peterson: 1991, 'Using Forcing to Prove Completeness of Resolution and Paramodulation'. *J. Symbolic Computation* **11**, 3–19.

Paul, E.: 1986, 'On solving the equality problem in theories defined by Horn clauses'. *Theoretical Computer Science* **44**, 127–153.

Paul, E.: 1992, 'A general refutational completeness result for an inference procedure based on associative-commutative unification'. *J. Symbolic Computation* **14**, 577–618.

Peterson, G.: 1983, 'A technique for establishing completeness results in theorem proving with equality'. *SIAM J. Computing* **12**, 82–100.

Peterson, G. and M. Stickel: 1981, 'Complete sets of reductions for some equational theories'. *J. Association for Computing Machinery* **28**, 233–264.

Robinson, G. and L. T. Wos: 1969, 'Paramodulation and theorem proving in first order theories with equality'. In: B. Meltzer and D. Michie (eds.): *Machine Intelligence 4*. New York: American Elsevier, pp. 133–150.

Robinson, J. A.: 1965a, 'Automatic Deduction with hyper-resolution'. *Int. J. of Comp. Math.* **1**, 227–234.

Robinson, J. A.: 1965b, 'A machine-oriented logic based on the resolution principle'. *J. Association for Computing Machinery* **12**, 23–41.

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

Rusinowitch, M.: 1987, 'Démonstration Automatique par des Techniques de Réécriture'. Thèse d'Etat, Univ. Nancy, France.

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

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

Waldmann, U.: 1997, 'Cancellative Abelian Monoids in Refutational Theorem Proving.'. Ph.D. thesis, Fachbereich Informatik, Universität des Saarlandes, Germany.

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

Wos, L.: 1988, *Automated Reasoning: 33 Basic Research Problems*. Englewood Cliffs, New Jersey: Prentice-Hall.

Wos, L. T., G. A. Robinson, D. F. Carson, and L. Shalla: 1967, 'The concept of demodulation in theorem proving'. *J. Association for Computing Machinery* **14**, 698–709.

Zhang, H.: 1988, 'Reduction, superposition and induction: Automated reasoning in an equational logic'. Ph.D. thesis, Rensselaer Polytechnic Institute, Schenectady, NY, USA.

Zhang, H. and D. Kapur: 1988, 'First-order theorem proving using conditional rewrite rules'. In: E. Lusk and R. Overbeek (eds.): *Proc. 9th Int. Conf. on Automated Deduction*, Vol. 310 of *Lecture Notes in Computer Science*. Berlin, pp. 1–20.