Positive Deduction modulo Regular Theories *

Laurent Vigneron

CRIN-CNRS & INRIA Lorraine
B.P.239, 54506 Vandœuvre-lès-Nancy Cedex, France
E-mail: Vigneron@Loria.Fr

Abstract. We propose a *new technique* for dealing with an equational theory $\mathcal{E}$ in the clausal framework. This consists of the definition of two inference rules called *contextual superposition* and *extended superposition*, and of an *algorithm* for computing the only needed applications of these last inference rules only by examining the axioms of $\mathcal{E}$. We prove the refutational completeness of this technique for a class of theories $\mathcal{E}$ that include *all the regular theories*, i.e. any theory whose axioms preserve variables. This generalizes the results of Wertz [31] and Paul [17] who could not prove the refutational completeness of their superposition-based systems for any regular theory.
We also combine a *collection of strategies* that decrease the number of possible deductions, without loss of completeness: the superposition strategy, the positive ordering strategy, and a simplification strategy.
These results have been implemented in a theorem prover called `DATAC`, for the case of commutative, and associative and commutative theories. It is an interesting tool for comparing the efficiency of strategies, and practical results will follow.

1 Introduction

The paramodulation rule permits one to deal with the equality predicate without explicitly describing its properties of reflexivity, symmetry, transitivity and functional reflexivity. It is also based on a notion of replacement. Over time, several refinements have been added to this rule. Brand [6] has shown that only the reflexivity axiom $x \simeq x$ is needed. Peterson [18] has shown that paramodulations into variables are useless. Hsiang and Rusinowitch [10] have introduced ordering restrictions to the application of these rules, and have proved the completeness of the following *ordering strategy*: *each inference step has to be applied between maximal (w.r.t. an ordering) literals in clauses, and in each paramodulation step, a term cannot be replaced by a bigger one.*
Other refinements, such as the superposition strategy which applies replacements only in biggest sides of equations, and clausal simplifications which delete redundant clauses, have followed [4].

* This work was done during a fellowship at the University of Stony Brook (NY, USA), funded by the *Institut National de Recherche en Informatique et en Automatique* (France).

The complete Hsiang-Rusinowitch strategy and others are unfortunately often inefficient in the presence of clauses such as the associativity property of an operator f, $f(f(x,y),z) \simeq f(x,f(y,z))$, which produces the divergence of derivations by successive superpositions into the subterm $f(x,y)$. Other properties, such as the commutativity of an operator, induce problems with the superposition rule because they cannot be oriented.

The most established solution was proposed by Plotkin [20]. He proposed to define an equational theory $\mathcal{E}$, by extracting the above properties from the set of clauses, and to define a *unification algorithm modulo* $\mathcal{E}$. This result has been the basis of much work: Lankford and Ballantyne [14] for the particular case of associative and commutative theories, and Peterson and Stickel [19] for completion. When applying ordering strategies to theorem proving in equational theories, we need to add various additional techniques. The techniques usually proposed in equational deduction are:

1. either to add an inference rule applying replacements into axioms of $\mathcal{E}$, and therefore generating *extensions* of these axioms,
2. or to associate to each equation the set of its possible extensions, which may be used later by the superposition rule.

These extensions have been studied by Jouannaud and Kirchner [13], and Bachmair and Dershowitz [1]. Both techniques have been used by Wertz [31]; the second has been used by Paul [17] too.

We propose in this paper a *new technique* for dealing with these extensions in the clausal framework, by defining two inference rules called *contextual superposition* and *extended superposition*. We also define an *algorithm* for computing the only needed applications of these last inference rules, only by examining the axioms of $\mathcal{E}$ and generalizing the $\mathcal{E}$-redundant context notions of Jouannaud and Kirchner [13] defined for equational completion. Our inference system is defined by combining the superposition strategy and the positive ordering strategy; it is also compatible with the simplification strategy.

The positive strategy was initially proposed by Robinson [21] and has been transformed many times later. Our definition of this strategy, whose first version was presented in [25], is a much more attractive one. The usual condition is to apply superposition steps from a positive clause. Here, we mention that whenever we want to use a positive literal, it has to belong to a positive clause. A similar strategy has also been independently defined by Paul in [17].

Our positive strategy uses a particular case of the superposition calculus with selection, defined by Bachmair and Ganzinger in [3], for selecting negative literals. But in addition we define a new kind of selection on positive literals: a positive literal can be used in a deduction if it belongs to a positive clause and if it is maximal in this clause (for a given ordering).

We prove the refutational completeness of our inference system for all the equational theories $\mathcal{E}$ allowed by Wertz [31] and Paul [17], but in addition we prove it *for any regular theory* $\mathcal{E}$, i.e. any theory whose axioms preserve variables.

Moreover, our algorithm for detecting $\mathcal{E}$-redundant contexts permits a significant decreasing of the number of possible deductions.

These results have been implemented in a theorem prover called **DATAC**, for commutative, and associative and commutative theories. It is an interesting tool for comparing the efficiency of strategies, and practical results will follow.

The layout of this paper is the following: after introducing the basic notions in Sect. 2, we describe our inference rules in Sect. 3. Section 4 presents a procedure to compute useful contexts for extended equations. The proof of refutational completeness of the inference system is sketched in Sect. 5, but it is detailed in [30] (see also [26, 29]). Section 6 presents an example of trace with our system **DATAC**.

2 Notations and Definitions

Let us define some basic notions, based on the standard notations and definitions for term rewriting and unification given in [8, 12].

Let $\mathcal{E}$ be an equational theory, i.e. a set of equations. The congruence generated by this set $\mathcal{E}$ is called *$\mathcal{E}$-equality* and written $=_{\mathcal{E}}$. A *substitution* is a function replacing some variables by terms. A substitution σ is said to *$\mathcal{E}$-unify* two terms s and t if $s\sigma$ and $t\sigma$ are $\mathcal{E}$-equal, and if σ is a most general $\mathcal{E}$-unifier of s and t (see [12]). In this case, σ is a solution of the $\mathcal{E}$-unification problem $s =_{\mathcal{E}}^{?} t$.

An *atom* is an equality $l \simeq r$. A *clause* is denoted $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$, where $A_1, \ldots, A_n, B_1, \ldots, B_m$ are atoms; this means A_1 *and... and* A_n *implies* B_1 *or... or* B_m. A *literal* is an atom appearing in a clause. A *literal is positive* (resp. *negative*) if it appears on the right-hand side (resp. left-hand side) of $\rightarrow$. A *clause is positive* if it contains only positive literals, i.e. if the left-hand side of $\rightarrow$ is empty.

To express subterms and substitutions, we use *positions*. Envision a term represented as a tree; a position in a term is a node of this tree. The subterm at position p of a term t is written $t|_p$. A position is a sequence of integers: $f(t_1, \ldots, t_n)|_{i \cdot p} = t_i|_p$; the empty sequence (*empty position*) is denoted ϵ ($t|_\epsilon = t$). A position p in a term t is a *non-variable position* if $t|_p$ is not a variable. The set of all non-variable positions of a term t is denoted $\mathcal{FP}os(t)$. The term $s[t]_p$ represents the term s whose subterm at position p is t.

To decrease the number of possible deductions, we use an ordering strategy. So, we assume we are given a *simplification ordering* $>$, defined on terms and atoms. For the sake of completeness, it has to be *total* on ground $\mathcal{E}$-congruence classes and *$\mathcal{E}$-compatible*, i.e.

$$\forall s, t \text{ ground terms, if } s > t \text{ and } s \neq_{\mathcal{E}} t, \text{ then } \forall s' =_{\mathcal{E}} s,\ \forall t' =_{\mathcal{E}} t,\ s' > t'$$

So, in our inference rules, we will use this ordering to orient equations and to check the maximality of an equation w.r.t. other equations. However terms may be incomparable; we will write that a term is maximal w.r.t. another term, if it

is not smaller than or equal to this second term.
Given an equality $l \simeq r$, we will assume l is maximal w.r.t. r.

3 Inference Rules

We describe in this section a set of inference rules for applying deductions modulo an equational theory $\mathcal{E}$. These rules are based on the *superposition strategy*, a variant of the paramodulation strategy; it applies replacements only in maximal sides of equations. This superposition strategy is combined with a positive ordering strategy to prune the search space. This strategy is described in the next definition, and needs a total simplification ordering for comparing terms.

Definition 1 (Positive Ordering Strategy).

- If an inference rule uses a positive literal in a clause, this clause has to be positive. In addition, the positive literal used has to be maximal in the clause.
- If an inference rule uses a negative literal in a clause, this literal has to be maximal w.r.t. the other negative literals of the clause.

The first inference rule is the Equational Factoring. Its purpose is to derive clauses in which two positive equations do not have $\mathcal{E}$-equal left-hand sides. This inference rule is essential for the completeness of the superposition strategy. Note that it is applied only on positive clauses.

Definition 2 (Equational Factoring). $\dfrac{\rightarrow l_1 \simeq r_1, l_2 \simeq r_2, R}{r_1\sigma \simeq r_2\sigma \rightarrow l_2\sigma \simeq r_2\sigma, R\sigma}$

where σ $\mathcal{E}$-unifies l_1 and l_2, $l_1\sigma \simeq r_1\sigma$ is maximal w.r.t. $l_2\sigma \simeq r_2\sigma$ and each equation of $R\sigma$. Moreover, $l_1\sigma$ has to be maximal w.r.t. $r_1\sigma$.

The next rule stands for avoiding the addition of the reflexivity axiom $x \simeq x$ of the equality predicate. It is the only rule which can derive the empty clause, symbolizing an incoherence in the initial set of clauses, since it is the only rule which deletes a literal.

Definition 3 (Reflexion). $\dfrac{l \simeq r, L \rightarrow R}{L\sigma \rightarrow R\sigma}$

where σ $\mathcal{E}$-unifies l and r, and $l\sigma \simeq r\sigma$ is maximal w.r.t. each equation of $L\sigma$.

The superposition rule applies the replacement of a term by an equal one, from a positive clause. It is decomposed into a *Left* and a *Right Superposition* rule, respectively defined by

$$\frac{\rightarrow l_1 \simeq r_1, R_1 \qquad l_2 \simeq r_2, L_2 \rightarrow R_2}{l_2[r_1]_{p_2}\sigma \simeq r_2\sigma, L_2\sigma \rightarrow R_1\sigma, R_2\sigma} \quad \text{and} \quad \frac{\rightarrow l_1 \simeq r_1, R_1 \qquad \rightarrow l_2 \simeq r_2, R_2}{\rightarrow l_2[r_1]_{p_2}\sigma \simeq r_2\sigma, R_1\sigma, R_2\sigma}$$

where σ is a $\mathcal{E}$-unifier of l_1 and the subterm at position p_2 of l_2. But, even with these two inference rules, the procedure of deduction is not complete, as shown in the next example.

Example 1. Let $\mathcal{E} = \{\ f(f(x_1, x_2), x_3) \simeq f(x_1, f(x_2, x_3))\ \}$. The following clauses

$$(1)\ \ \rightarrow f(a,b) \simeq c \qquad (2)\ \ \rightarrow f(a, f(b,d)) \simeq e \qquad (3)\ \ f(c,d) \simeq e \rightarrow$$

form an incoherent set with $\mathcal{E}$, since: $\mathcal{E}$ permits a modification of the parentheses in the left-hand side of the second clause, to obtain $\rightarrow f(f(a,b),d) \simeq e$, and replacing $f(a,b)$ by c in this term (thanks to (1)), we deduce $\rightarrow f(c,d) \simeq e$ which contradicts (3).
However, there is no possible inference step between the three initial clauses. $\diamond$

We solve this problem by applying superpositions from *extended equations*, i.e. from equations $e[l_1]_p \simeq e[r_1]_p$, where e is a term and p a non-variable position in e. Such a pair (e,p) is called a *context*. In the previous example, a contradiction can be derived using the context $(f(f(x_1, x_2), x_3), 1)$, producing the extended equation $f(f(a,b), x_3) \simeq f(c, x_3)$ from $f(a,b) \simeq c$.

By the Critical Pairs Lemma of Jouannaud and Kirchner [13], we know that contexts can be computed. We define in Sect. 4 a procedure to compute all the possible contexts for a given equational theory $\mathcal{E}$. Given a term l, $\mathcal{C}ont(l)$ is the set of all contexts (e,p) for $\mathcal{E}$ such that $e|_p$ and l are $\mathcal{E}$-unifiable. These contexts are used in three new inference rules.

The first two rules simulate replacements from an equation or an extended equation. Indeed, we assume that the context (l, ϵ) belongs to $\mathcal{C}ont(l)$. Left and right superposition rules are therefore particular cases of the next inference rules.

Definition 4 (Left Contextual Superposition).

$$\frac{\rightarrow l_1 \simeq r_1, R_1 \qquad\qquad l_2 \simeq r_2, L_2 \rightarrow R_2}{l_2[e_1[r_1]_{p_1}]_{p_2}\sigma \simeq r_2\sigma, L_2\sigma \rightarrow R_1\sigma, R_2\sigma}$$

where p_2 is a non-variable position in l_2, (e_1, p_1) is a context[2] in $\mathcal{C}ont(l_1)$, σ $\mathcal{E}$-unifies $l_2|_{p_2}$ and $e_1[l_1]_{p_1}$, $l_1\sigma \simeq r_1\sigma$ is maximal for $>$ in its clause, and $l_2\sigma \simeq r_2\sigma$ is maximal w.r.t. each atom of $L_2\sigma$. Moreover, $l_1\sigma$ has to be maximal w.r.t. $r_1\sigma$, and $l_2\sigma$ has to be maximal w.r.t. $r_2\sigma$. The replacing term in the deduced clause is the extension of the right-hand side, $e_1[r_1]_{p_1}$.

Definition 5 (Right Contextual Superposition).

$$\frac{\rightarrow l_1 \simeq r_1, R_1 \qquad\qquad \rightarrow l_2 \simeq r_2, R_2}{\rightarrow l_2[e_1[r_1]_{p_1}]_{p_2}\sigma \simeq r_2\sigma, R_1\sigma, R_2\sigma}$$

where the only difference with Left Contextual Superposition is that $l_2\sigma \simeq r_2\sigma$ is maximal in its clause and maximal w.r.t. $l_1\sigma \simeq r_1\sigma$.

The next inference rule simulates a superposition between two extended equations, at the top of their maximum side.

Definition 6 (Extended Superposition).

$$\frac{\rightarrow l_1 \simeq r_1, R_1 \qquad\qquad \rightarrow l_2 \simeq r_2, R_2}{\rightarrow e_1[r_1]_{p_1}\sigma \simeq e_2[r_2]_{p_2}\sigma, R_1\sigma, R_2\sigma}$$

[2] (e_1, p_1) may be an empty context, i.e. $p_1 = \epsilon$.

where, given a non-empty context (e_1, p_1) in $\mathcal{C}ont(l_1)$ and a non-empty context (e_2, p_2) in $\mathcal{C}ont(l_2)$, σ $\mathcal{E}$-unifies $e_1[l_1]_{p_1}$ and $e_2[l_2]_{p_2}$. Each equation has to be maximal in its clause, and their left-hand side has to be maximal w.r.t. their right-hand side.

3.1 About the Superposition Strategy

The principle of the superposition strategy is *to apply replacements only into maximal sides of equations*, and has been extensively used for term rewriting and completion. But, the completeness of inference systems representing this strategy has been a longstanding open problem. For completeness, either some deductions using non-maximal literals [24], or some replacements into minimal sides of equations [3], were needed. Bachmair and Ganzinger [2] have proved the completeness in the empty theory of the entire superposition strategy by adding two *Equational Factoring* rules (one for negative and one for positive literals). Defining a particular ordering for comparing negative and positive literals, Nieuwenhuis and Rubio [16] have proved that the rule on negative literals is useless.

In our inference rules, we never need to compare such literals: we always compare literals of the same sign. So for us, *specifying a special ordering on literals is useless.*

3.2 Other Predicate Symbols

Our five inference rules have been defined for deduction in first-order logic with a unique predicate, the equality predicate. This restriction has been decided only for simplifying notations, but it is easy to adapt the inference rules to the presence of other predicate symbols. And we have to add a Factoring rule (applied only on positive clauses) and a Resolution rule (applied with a positive clause) for dealing with the non-equational literals (see [26]). The new system of deduction remains complete if the equality symbol is minimal in precedence.

Now that we have defined all the inference rules, let us show how to compute extended equations with contexts.

4 Extended Equations

An extension of an equation $l \simeq r$ is an equation $e[l]_p \simeq e[r]_p$, also called an *extended equation* of $l \simeq r$, where e is a term, p a non variable position in this term. The subterm at position p in e is $\mathcal{E}$-unifiable with l, the maximum side of the equation. The couple (e, p) is called the *context* of this extension.

The set of all possible contexts for a theory $\mathcal{E}$, written $\mathcal{C}_\mathcal{E}$, is defined by $\bigcup_{k \geq 0} \mathcal{C}ont_k$, where the sets $\mathcal{C}ont_k$ are inductively defined by:

$$\boxed{\begin{array}{rcl} \mathcal{C}ont_0 & = & \{\ (e,p)\ \mid\ \exists e \simeq e' \text{ or } e' \simeq e \in \mathcal{E},\ p \in \mathcal{FP}os(e) \text{ and } p \neq \epsilon\ \} \\ \mathcal{C}ont_{k+1} & = & \{\ (e_1[e_2]_{p_1}, p_1 \cdot p_2)\ \mid\ (e_1,p_1) \in \mathcal{C}ont_0,\ (e_2,p_2) \in \mathcal{C}ont_k, \\ & & \qquad\qquad \text{and } e_1|_{p_1} \text{ and } e_2 \text{ are } \mathcal{E}\text{-unifiable}\ \} \end{array}}$$

Then, given an equation $l \simeq r$ where l is maximal w.r.t. r, the set of all possible contexts which can extend $l \simeq r$ is defined by:

$$\mathcal{C}ont(l) \;=\; \{\, (e,p) \in \mathcal{C}_{\mathcal{E}} \;\mid\; e|_p \text{ and } l \text{ are } \mathcal{E}\text{-unifiable} \,\} \;\cup\; \{\, (l,\epsilon) \,\}$$

We have added (l,ϵ) for avoiding the definition of special inference rules, applying superpositions without context.

Let $l \rightarrow r$ be a ground rewrite rule. Let $\mathcal{C}_l$ be the set of all ground instances of contexts of $\mathcal{C}ont(l)$. We define the relation $\longrightarrow_{\mathcal{C}_l,\mathcal{E}}$ by:

$$t_1 \longrightarrow_{\mathcal{C}_l,\mathcal{E}} t_2 \;\text{ if }\; \exists (e_l,p_l) \in \mathcal{C}_l,\; \exists q \in \mathcal{FP}os(t_1),\; t_1|_q =_{\mathcal{E}} e_l,\; t_2 = t_1[e_l[r]_{p_l}]_q$$

This relation $\longrightarrow_{\mathcal{C}_l,\mathcal{E}}$ satisfies a property called *$\mathcal{E}$-closure* if: whenever a term t is reducible into a term t_1 by the relation $\longrightarrow_{\mathcal{C}_l,\mathcal{E}}$, then for each term t_2, $\mathcal{E}$-equal to t, t_2 and t_1 are reducible by $\longrightarrow_{\mathcal{C}_l,\mathcal{E}}$ into two $\mathcal{E}$-equal terms.
A set of contexts $\mathcal{C}$ is said to be *$\mathcal{E}$-covering* if, for any ground term l, the relation $\longrightarrow_{\mathcal{C}_l,\mathcal{E}}$ satisfies the property of $\mathcal{E}$-closure, where $\mathcal{C}_l = \mathcal{C} \cap \mathcal{C}ont(l)$.

Proposition 7. *Let $\mathcal{E}$ be an equational theory. The set of contexts $\mathcal{C}_{\mathcal{E}}$ is $\mathcal{E}$-covering.*

This Proposition means that the role of the equations of $\mathcal{E}$ is entirely simulated by superpositions with contexts of $\mathcal{C}_{\mathcal{E}}$. Its proof is similar to the proof of the Critical Pairs Lemma of Jouannaud and Kirchner [13] (see [29]), and consists of simple case analyses.

However, the definition of $\mathcal{C}_{\mathcal{E}}$ is very general, and for efficiency we combine it with a procedure deleting redundant contexts. Before describing this procedure, let us introduce some definitions.

Definition 8. A context (e_1,p_1) is **redundant at a position p** w.r.t. a set of contexts $\mathcal{C}$, if p is a non-variable position in e_1 and $p_1 = p{\cdot}q$ (where $q \neq \epsilon$), and if there is a context (e_2,p_2) in $\mathcal{C}$ and a substitution σ such that:

1. $e_2[\cdot]_{p_2}\sigma =_{\mathcal{E}} (e_1|_p)[\cdot]_q$ for guaranteeing the equivalence of terms $e_2\sigma$ and $e_1|_p$,
2. $(e_2|_{p_2})\sigma =_{\mathcal{E}} e_1|_{p_1}$ for guaranteeing the equivalence of subterms where replacements will apply.

where the symbol $\cdot$ is a new constant. (e_1,p_1) is said to be redundant at p w.r.t. $\mathcal{C}$, by $(e_2\sigma,p_2)$. If p is ϵ, the context (e_1,p_1) is said to be top-redundant.

Definition 9. Let (e_1,p_1) be a context. Let e'_1 be a ground term and σ a ground substitution such that e'_1 is $\mathcal{E}$-equal to $e_1\sigma$. The representation e'_1 of the context (e_1,p_1) is said to be ***$\mathcal{E}$*-redundant at a position p** w.r.t. a set of contexts $\mathcal{C}$, if there is a term e_2 $\mathcal{E}$-equal to $e'_1|_p$ and a non-variable position p_2 in e_2, s.t. :

1. (e_2,p_2) is top-redundant w.r.t. $\mathcal{C}$, by a context (e_3,p_3),
2. $(e_1\sigma,p_1)$ is top-redundant by $(e'_1[e_3]_p, p{\cdot}p_3)$.

Note that the position p may be the empty position.

A context (e_1, p_1) is **$\mathcal{E}$-redundant** w.r.t. a set of contexts $\mathcal{C}$ if,

1. either (e_1, p_1) is top-redundant w.r.t. $\mathcal{C}$,
2. or, for each term e'_1, $\mathcal{E}$-equal to a ground instance $e_1\sigma$ of e_1,
   (a) either there is a non-variable position p'_1 in e'_1 such that (e'_1, p'_1) is top-redundant by (e_1, p_1),
   (b) or the representation e'_1 of the context (e_1, p_1) is $\mathcal{E}$-redundant at a position p' w.r.t. $\mathcal{C}$.

Fig. 1. Redundancy criteria of a context in $\mathcal{E}$.

Proposition 10. *Let $\mathcal{E}$ be an equational theory. The set of contexts $\mathcal{C}_{\mathcal{E}}$, constructed with the $\mathcal{E}$-redundancy criteria described in Fig. 1, is $\mathcal{E}$-covering.*

Proof. *Uselessness of redundant contexts is easily derived from the algorithm described in Fig. 1 as follows:*

1. *If there is a context (e_2, p_2) in $\mathcal{C}$ such that any replacement with the context (e_1, p_1) is an instance of a replacement with this context (e_2, p_2), then to use (e_2, p_2) instead of (e_1, p_1) generates the same result, or a more general one.*
2. *Let us study the terms in which the context (e_1, p_1) could be applied. A first remark is there is no need to use this context with terms that are instances of e_1; the replacement can be applied directly at the position p_1. We can generalize this remark: (e_1, p_1) is useless if all terms in which it could be applied can be treated without context or with another context of $\mathcal{C}$. But to test this for each term $\mathcal{E}$-equal to e_1 is not sufficient, because a term $\mathcal{E}$-equal to an instance of e_1 may not be an instance of a term $\mathcal{E}$-equal to e_1. So, the context (e_1, p_1) is $\mathcal{E}$-redundant if, for each term e'_1 $\mathcal{E}$-equal to a ground instance $e_1\sigma$ of e_1,*
   (a) either a term $\mathcal{E}$-equal to $e_1|_{p_1}$ appears at a position p'_1 of e'_1, i.e. the replacement can be directly done at this position; in addition, we have to check that the result is identical to the one obtained with the context (e_1, p_1),
   (b) or a context of $\mathcal{C}$ can be applied at a position p' of e'_1, producing the same result as applying the context (e_1, p_1) to the top of e'_1.

In practice, to check the second point does not consist of studying all the ground instances of e_1, but of enumerating the different forms that can have these instances. And we can note that if the context (e_1, p_1) is redundant at a non-empty position p by a context (e_2, p_2), then all its representations $e'_1 =_{\mathcal{E}} e_1\sigma$ such that

$$\exists p' \in \mathcal{FP}os(e'_1),\ e'_1|_{p'} =_{\mathcal{E}} (e_1|_p)\sigma \text{ and } e'_1[\cdot]_{p'} =_{\mathcal{E}} e_1[\cdot]_p\sigma$$

are $\mathcal{E}$-redundant at the position p' by the context (e_2, p_2). □

A simple algorithm for constructing the contexts with the redundancy criteria of Fig. 1 is, for each context newly created, to verify it is not $\mathcal{E}$-redundant w.r.t. the set $\mathcal{C}$ of the contexts already constructed; then, we delete from $\mathcal{C}$ the contexts $\mathcal{E}$-redundant by the addition of this new context. Moreover, it would be interesting, when applying an inference rule involving a context, to check the non-redundancy of the instance of this context used, and even to check the non-redundancy of the representation of its term in the clause where the replacement is going to apply.

Let us give two examples of the construction of contexts.

Example 2 (Associativity and Commutativity). If $\mathcal{E}$ represents properties of associativity and commutativity of an operator f,

$$\mathcal{E} \;=\; \{\; f(f(x_1,x_2),x_3) \simeq f(x_1,f(x_2,x_3)),\;\; f(x_1,x_2) \simeq f(x_2,x_1)\;\}$$

$\mathcal{C}ont_0$ contains two contexts, $(f(f(x_1,x_2),x_3),1)$ and $(f(x_1,f(x_2,x_3)),2)$, but the second one is top-redundant by the first one. $(f(f(f(x_1,x_2),x_3),x_4),1{\cdot}1)$, the unique context of $\mathcal{C}ont_1$, is top-redundant by $(f(f(x_1,x_2),x_3),1)$ too. Hence, $\mathcal{C}_{AC} = \{\;(f(f(x_1,x_2),x_3),1)\;\}$, which means that the only possible extension of an equation $l \simeq r$ is $f(l,x_3) \simeq f(r,x_3)$. ◇

Example 3 (Associativity). If $\mathcal{E}$ contains the property of associativity of f,

$$\mathcal{E} \;=\; \{\; f(f(x_1,x_2),x_3) \simeq f(x_1,f(x_2,x_3))\;\}$$

the non-redundant contexts are

$$\begin{aligned}
\mathcal{C}ont_0 &= \{\;(f(f(x_1,x_2),x_3),1),\;(f(x_1,f(x_2,x_3)),2)\;\}\\
\mathcal{C}ont_1 &= \{\;(f(f(x_1,f(x_2,x_3)),x_4),1{\cdot}2)\;\}
\end{aligned}$$

So, there are three useful extensions of an equation $l \simeq r$ where f is the top-symbol of l: to add a new variable, either on the right, $f(l,x_3) \simeq f(r,x_3)$, or on the left, $f(x_1,l) \simeq f(x_1,r)$, or on both sides, $f(f(x_1,l),x_4) \simeq f(f(x_1,r),x_4)$. ◇

4.1 Refining the Construction of Contexts

In the construction of the sets of contexts $\mathcal{C}ont_k$, we have used the notion of $\mathcal{E}$-unifiability. For instance, for building a context $(e_1[e_2[e_3]_{p_2}]_{p_1}, p_1{\cdot}p_2{\cdot}p_3)$, we have assumed that $e_2|_{p_2}$ and e_3 are $\mathcal{E}$-unifiable, and that $e_1|_{p_1}$ and $e_2[e_3]_{p_2}$ are $\mathcal{E}$-unifiable. But, in this last test, we have lost the information that $e_2|_{p_2}$ and e_3 have to be $\mathcal{E}$-unifiable. There may be no substitution satisfying both conditions, and therefore the context may be useless.

A simple way for avoiding such cases, is to add a third element to each context: *the conjunction of the $\mathcal{E}$-unification constraints* encountered to construct it. In the previous example, the context would be:

$$(e_1[e_2[e_3]_{p_2}]_{p_1},\; p_1{\cdot}p_2{\cdot}p_3,\; \{e_2|_{p_2} =^?_{\mathcal{E}} e_3 \;\wedge\; e_1|_{p_1} =^?_{\mathcal{E}} e_2[e_3]_{p_2}\})$$

Hence, a context is created only if its unification problems admit at least one solution. As a second consequence, for applying an inference rule using a context, we solve the specific unification problem of this rule, but in conjunction with the unification problems of the context.

With this additional parameter, less contexts are constructed, less inference rules are applicable and their unification problems have less solutions.

However, even with this optimization, there is an infinite number of contexts for a lot of theories, as shown in the next example. This can be dealt with using the algorithm building contexts with incrementality.

Example 4 (Distributivity). Let $\mathcal{E} = \{ f(x_1, g(x_2, x_3)) \simeq g(f(x_1, x_2), f(x_1, x_3)) \}$. $\mathcal{C}ont_0$ contains the three contexts

$$(f(x_1, g(x_2, x_3)), 2),\ (g(f(x_1, x_2), f(x_1, x_3)), 1),\ (g(f(x_1, x_2), f(x_1, x_3)), 2)$$

The context $(f(x_1, f(x_2, g(x_3, x_4))), 2{\cdot}2)$ belongs to $\mathcal{C}ont_1$, and so on... We can build an infinite sequence of contexts of the form:

$$(f(x_1, f(x_2, \ldots f(x_n, g(x_{n+1}, x_{n+2})))), 2^n)$$

These contexts are all useful: they permit to recover the subterm $g(x_{n+1}, x_{n+2})$, where the replacement occurs, from the representation:

$$g(f(x_1, f(x_2, \ldots f(x_n, x_{n+1}))), f(x_1, f(x_2, \ldots f(x_n, x_{n+2})))) \qquad \diamond$$

5 Refutational Completeness

A set of clauses S is said to be $\mathcal{E}$*-incoherent* if there is no model such that $S \cup \mathcal{E}$ is valid in this model. Let us define two properties of a theory $\mathcal{E}$:

- **(P1)** *Regularity.* For any equation $e_1 \simeq e_2$ in $\mathcal{E}$, each variable of e_1 is a variable of e_2, and vice-versa.
- **(P2)** For any ground term s that is $\mathcal{E}$-equal to one of its strict subterms $s|_p$ $(p \neq \epsilon)$, for any ground term t, $s[t]_p$ has to be $\mathcal{E}$-equal to t.

Let *INF* be the set of the five inference rules described in Sect. 3. Let us state the theorem of refutational completeness of *INF*.

Theorem 11 (Completeness). *Let $\mathcal{E}$ be an equational theory satisfying at least one of the properties (P1) and (P2), and let S be a set of clauses. If S is $\mathcal{E}$-incoherent, INF will always derive a contradiction from S.*

This theorem states that our inference system is refutationally complete if $\mathcal{E}$ satisfies (P1) or (P2). This result is an important improvement of previous works of Wertz [31] and Paul [17], since they proved the completeness of their systems only if $\mathcal{E}$ satisfies (P2). Moreover, our inference system limits the number of possible deductions much more than Wertz' and Paul's systems, thanks to the positive ordering strategy and the notion of $\mathcal{E}$-redundant contexts.

Note that many theories satisfy (P1) but not (P2). For instance, if $\mathcal{E}$ is $\{f(x, 0) \simeq x\}$, $\mathcal{E}$ is regular but: given the ground term $f(0, 0)$, it is $\mathcal{E}$-equal to 0; however, for a constant a, $f(0, a)$ is not $\mathcal{E}$-equal to a. (P2) is not satisfied.

There are also some particular theories $\mathcal{E}$ that satisfy (P2) but not (P1). For instance[3], if $\mathcal{E}$ is $\{f(x,y) \simeq x\}$, f is the only functional symbol and a is the only constant, the term $f(a,a)$ is $\mathcal{E}$-equal to a, and for any ground term t, $f(a,t) =_{\mathcal{E}} t$. Indeed, since f and a are the only symbols, any ground term (a, $f(a,a)$, $f(f(a,a),a)$,...) is $\mathcal{E}$-equal to a.

We prove the Theorem of Completeness by the transfinite semantic tree method of Hsiang and Rusinowitch [10], extended to deduction modulo an equational theory in [26, 29]. Let us give a sketch of this proof, as it is rather similar to the proof for the particular case of associative and commutative theories [26] (see [29, 30] for the detailed proofs).

Proof. Let $\mathcal{E}$ be a theory satisfying (P1) or (P2). Let S be an $\mathcal{E}$-incoherent set of clauses. Let us describe the main steps of the proof of refutational completeness. It is realized in the ground case, because each deduction step with ground instances clauses can be lifted to the general case.

Given a total $\mathcal{E}$-compatible ordering on ground atoms, we sort them by increasing order, and we construct the transfinite semantic tree $\mathcal{T}$ (in the empty theory). An interpretation *is a node of this tree.*

As S is $\mathcal{E}$-incoherent, each branch of the semantic tree $\mathcal{T}$ has a node that falsifies either a ground instance of a clause of S, or a trivial equation $t \simeq t'$ where $t =_{\mathcal{E}} t'$. Such nodes are called failure nodes*. The maximal subtree of $\mathcal{T}$ which does not contain a failure node is called the* maximal consistent tree*, and written $MCT(S)$.*

Our inference system INF is refutationally complete if it is always able to derive a contradiction (the empty clause) from S. Let $INF^*(S)$ be the set of all clauses deduced by INF from S, in any number of steps. For proving that $INF^*(S)$ contains the empty clause, we show that the maximal consistent tree for $INF^*(S)$, $MCT(INF^*(S))$, is reduced to an empty tree.

The first step is to choose a branch in $MCT(INF^*(S))$ that is consistent with the theory $\mathcal{E}$. This is done by adding new special failure nodes: distant failure nodes *and* quasi-failure nodes*.*

- *Let K be a failure node at the level of an atom $u \simeq w$ s.t. $u > w$, w is reducible and $u \simeq w$ is falsified by K. If there is an irreducible atom $u \simeq v$, smaller than $u \simeq w$ and s.t. K satisfies $w \simeq v$ (therefore K falsifies $u \simeq v$), the restriction of K to the level of $u \simeq v$ is a* distant failure node*.*
  This distant failure node permits to avoid a branch where there is a failure node falsifying an equation in which only the smallest side is reducible (condition of the superposition strategy).
- *Let K be an interpretation defined on atoms $A_1, \ldots, A_n$. Let A_{n+1} be an irreducible equation $u_1 \simeq v$ s.t. $u_1 > v$. K has two extensions: L, satisfying $u_1 \simeq v$, and R, falsifying $u_1 \simeq v$. R is a* quasi-failure node *if there is a term u_2, $\mathcal{E}$-equal to u_1, s.t. $u_2 \simeq v$ is reducible by an equation $l \simeq r$ into $u_2[r] \simeq v$,*

[3] This example has been suggested to me by Wayne Snyder.

and K satisfies $u_2[r] \simeq v$.
This quasi-failure node avoids to have $u_1 \simeq v$ satisfied and $u_2 \simeq v$ falsified in the same branch; this would be inconsistent with $\mathcal{E}$.

In the proof of consistency with $\mathcal{E}$ of this branch, we encounter a major problem; we have to prove that the following case cannot happen in the chosen branch: two $\mathcal{E}$-equal atoms $u_1 \simeq v$ and $u_2 \simeq v$ are interpreted differently, $u_1 \simeq v$ is reducible in u_1 by $l_1 \simeq r_1$, and $u_2 \simeq v$ is reducible in u_2 by $l_2 \simeq r_2$. For the case of associative and commutative theories [26], we show that the branch falsifies a ground instance of a clause of $INF^*(S)$, produced by an extended superposition between $l_1 \simeq r_1$ and $l_2 \simeq r_2$. But, for a general theory $\mathcal{E}$, it is not so easy. The terms $u_1[l_1]$ and l_1 may be $\mathcal{E}$-equal, and in such a situation, we have to prove that $u_1[r_1] \simeq r_1$ is valid in the chosen branch.
Wertz [31] and Paul [17] have decided to only consider theories $\mathcal{E}$ such that, whenever $u_1[l_1]$ and l_1 are $\mathcal{E}$-equal, $u_1[r_1]$ and r_1 are $\mathcal{E}$-equal too (Property (P2)).
In addition, studying the transformation of $u_1[l_1]$ into l_1 by $\mathcal{E}$-equality steps, we prove that $u_1[r_1] \simeq r_1$ is always valid if the theory $\mathcal{E}$ is regular (Property (P1)).

The last step of the proof is to show that the branch is empty. This implies that $MCT(INF^*(S))$ is empty, and also that the empty clause belongs to $INF^*(S)$. A study of the leaves of this branch, i.e. of failure nodes, distant failure nodes and/or quasi-failure nodes cutting it, shows that this branch falsifies a clause of $INF^*(S)$, deduced from clauses falsified by the leaves.
The final solution is that the branch is empty, and therefore the empty clause belongs to $INF^*(S)$.

The compatibility with the positive strategy is a consequence of the following property: if a (distant) failure node along the chosen branch, occuring at the level of an atom A_i, falsifies A_i, then it falsifies a positive clause of $INF^*(S)$. The proof of this property is done by induction on the failure and distant failure nodes, as in [23] for the deduction in the empty theory. □

Our inference system *INF* is compatible with the *simplification strategy*, if the derivations are fair, i.e. do not infinitely forget a possible deduction. This strategy has for purpose the deletion of redundant clauses. Let us give some examples of simplification rules:

- *Simplification* (also called *Demodulation*): it consists of applying a term rewriting step, using a procedure of matching modulo $\mathcal{E}$.
- *Clausal Simplification*: if there is a clause $\rightarrow A$ (resp. $A \rightarrow$), then each clause of the form $A', L \rightarrow R$ (resp. $L \rightarrow A', R$), where A' is $\mathcal{E}$-equal to an instance of A, is replaced by $L \rightarrow R$.
- *Trivial Reflexion*: a clause of the form $l \simeq r, L \rightarrow R$, where l is $\mathcal{E}$-equal to r, is replaced by $L \rightarrow R$.
- *Tautology Deletion*: each clause of the form $L \rightarrow l \simeq r, R$ where $l =_{\mathcal{E}} r$, or $A, L \rightarrow A', R$ where $A =_{\mathcal{E}} A'$, is deleted.

INF is also compatible with the subsumption: if a clause C_1 subsumes a clause C_2 thanks to a substitution σ, i.e. each literal of $C_1\sigma$ is $\mathcal{E}$-equal to a literal of C_2, the clause C_2 is deleted.

6 Implementation

The inference system described in this paper is implemented in the system **DATAC** for the case where $\mathcal{E}$ represents properties of commutativity, or associativity and commutativity (AC), of operators.

DATAC is a theorem prover written in CAML Light (18000 lines), a functional language of the ML family; it has a graphical interface written in Tcl/Tk, X11 Toolkit based on the language Tcl. It runs on SUN, HP and IBM PC workstations.
It uses an AC-unification algorithm based on the algorithm of Stickel [27] and the technique for solving Diophantine equations of Fortenbacher [9]. The algorithm for AC-matching is inspired by the algorithm of Hullot [11]. The ordering for comparing terms is the APO of Bachmair and Plaisted [5] with the improvements of Delor and Puel [7].

Let us detail an example of execution in modular lattices, where $\cdot$ denotes the function *meet*, $+$ the function *join*, 1 the greatest element and 0 the least element. The predicate symbol *Comp* denotes the complementarity of two elements (*Comp* is commutative). The equational theory $\mathcal{E}$ is the following:

$$\mathcal{E} \;=\; \left\{ \begin{array}{rcl} (x_1 + x_2) + x_3 & \simeq & x_1 + (x_2 + x_3) \\ x_1 + x_2 & \simeq & x_2 + x_1 \\ (x_1 \cdot x_2) \cdot x_3 & \simeq & x_1 \cdot (x_2 \cdot x_3) \\ x_1 \cdot x_2 & \simeq & x_2 \cdot x_1 \end{array} \right\}$$

There are only two useful contexts for this theory $\mathcal{E}$ (cf. Example 2):

$$\mathcal{C}_{\mathcal{E}} \;=\; \{\ ((x_1 + x_2) + x_3, 1),\ ((x_1 \cdot x_2) \cdot x_3, 1)\ \}$$

The initial set of clauses is:

(1) $\rightarrow\ x_1 \cdot x_1 \simeq x_1$ *(2)* $\rightarrow\ x_1 + x_1 \simeq x_1$
(3) $\rightarrow\ x_1 \cdot (x_1 + x_2) \simeq x_1$ *(4)* $\rightarrow\ x_1 + (x_1 \cdot x_2) \simeq x_1$
(5) $\rightarrow\ x_1 \cdot 0 \simeq 0$ *(6)* $\rightarrow\ x_1 + 0 \simeq x_1$
(7) $\rightarrow\ x_1 \cdot 1 \simeq x_1$ *(8)* $\rightarrow\ x_1 + 1 \simeq 1$
(9) $x_1 \cdot x_2 \simeq x_1\ \rightarrow\ x_2 \cdot (x_1 + x_3) \simeq x_1 + (x_3 \cdot x_2)$
(10) $Comp(x_1, x_2)\ \rightarrow\ x_1 \cdot x_2 \simeq 0$ *(11)* $Comp(x_1, x_2)\ \rightarrow\ x_1 + x_2 \simeq 1$
(12) $x_1 + x_2 \simeq 1,\ x_1 \cdot x_2 \simeq 0\ \rightarrow\ Comp(x_1, x_2)$

The property we want to prove is:

> *For all elements a and b, let c_1 be the complement of $a \cdot b$ and let c_2 be the complement of $a + b$; then $c_2 + (c_1 \cdot b)$ is the complement of a.*

For this purpose, we add three new clauses that represent the negation of this property (A, B, C_1 and C_2 are new constants):

(13) $\rightarrow\ Comp(C_1, A \cdot B)$ (14) $\rightarrow\ Comp(C_2, A + B)$
(15) $Comp(A, C_2 + (C_1 \cdot B))\ \rightarrow$

The theorem prover `DATAC` is run with these 15 initial clauses, and with the precedence ordering $\cdot > + > B > A > C_1 > C_2 > 1 > 0$ on functional operators, and $Comp > \simeq$ on predicate operators. Deductions are applied thanks to the inference rules defined in Sect. 3, combined with a resolution rule (for dealing with the predicate $Comp$). These deduction rules combine the positive ordering strategy with the superposition strategy. When a contextual superposition uses an empty context, we simply call it a superposition.
Note that we are going to use a flattened representation under AC operators, i.e. a term $C_1 \cdot (A \cdot B)$ will be written $C_1 \cdot A \cdot B$.
`DATAC` automatically derives a contradiction, the empty clause written $\Box$, in the following way:

`Resolution between 10 and 13`
(16) $\rightarrow\ A \cdot B \cdot C_1 \simeq 0$

`Resolution between 11 and 13`
(17) $\rightarrow\ (A \cdot B) + C_1 \simeq 1$

`Resolution between 10 and 14`
(18) $\rightarrow\ (A + B) \cdot C_2 \simeq 0$

`Resolution between 11 and 14`
(19) $\rightarrow\ A + B + C_2 \simeq 1$

`Left Contextual Superposition from 1 into 9`
(32) $x_1 \cdot x_2 \simeq x_1 \cdot x_2\ \rightarrow\ x_1 \cdot ((x_1 \cdot x_2) + x_3) \simeq (x_1 \cdot x_2) + (x_3 \cdot x_1)$

`Trivial Reflexion in 32`
(32) $\rightarrow\ x_1 \cdot ((x_1 \cdot x_2) + x_3) \simeq (x_1 \cdot x_2) + (x_3 \cdot x_1)$

`Left Contextual Superposition from 3 into 9`
(63) $x_1 \cdot x_3 \simeq x_1 \cdot x_3\ \rightarrow\ (x_1 + x_2) \cdot ((x_1 \cdot x_3) + x_4) \simeq (x_1 \cdot x_3) + (x_4 \cdot (x_1 + x_2))$

`Trivial Reflexion in 63`
(63) $\rightarrow\ (x_1 + x_2) \cdot ((x_1 \cdot x_3) + x_4) \simeq (x_1 \cdot x_3) + (x_4 \cdot (x_1 + x_2))$

`Right Superposition from 17 into 32`
(131) $\rightarrow\ (B \cdot A) + (C_1 \cdot B) \simeq B \cdot 1$

`Simplification from 7 into 131`
(131) $\rightarrow\ (B \cdot A) + (C_1 \cdot B) \simeq B$

`Extended Superposition between 3 and 63`
(197) $\rightarrow\ ((x_2 \cdot x_3) + (x_4 \cdot (x_2 + x_1))) \cdot x_1 \simeq x_1 \cdot ((x_2 \cdot x_3) + x_4)$

`Extended Superposition between 4 and 131`
(267) $\rightarrow\ A + (C_1 \cdot B) \simeq B + A$

`Right Superposition from 18 into 197`
(397) $\rightarrow\ A \cdot ((B \cdot x_1) + C_2) \simeq ((B \cdot x_1) + 0) \cdot A$

`Simplification from 6 into 397`
(397) $\rightarrow\ A \cdot ((B \cdot x_1) + C_2) \simeq B \cdot x_1 \cdot A$

`Left Superposition from 397 into 12`

(1214) $(B \cdot x_1) + C_2 + A \simeq 1,\ B \cdot x_1 \cdot A \simeq 0\ \rightarrow\ Comp((B \cdot x_1) + C_2, A)$

`Left Superposition from 16 into 1214`

(2541) $(B \cdot C_1) + C_2 + A \simeq 1,\ 0 \simeq 0\ \rightarrow\ Comp((B \cdot C_1) + C_2, A)$

`Trivial Reflexion in 2541`

(2541) $(B \cdot C_1) + C_2 + A \simeq 1\ \rightarrow\ Comp((B \cdot C_1) + C_2, A)$

`Clausal Simplification in 2541 thanks to 15`

(2541) $(B \cdot C_1) + C_2 + A \simeq 1\ \rightarrow$

`Simplification from 267 into 2541`

(2541) $B + A + C_2 \simeq 1\ \rightarrow$

`Clausal Simplification in 2541 thanks to 19`

(2541) $\square$

The following table compares our positive ordering strategy with the classical ordering strategy [10], which requires only that deductions have to apply between maximal literals of clauses. For this comparison, we applied two linear completion steps on the 12 initial clauses of the previous example. A step of linear completion consists of applying all possible deductions between the initial clauses, but none with one of the deduced clauses. The second step for the ordering strategy was stopped because of a lack of memory while solving a tricky AC-unification problem.
The last column of this table presents statistics for the example traced above. For this example, we have used a simplified version of the AC-unification algorithm that permits not to compute all the minimal solutions and not to solve tricky problems. A consequence is the loss of the completeness of the strategy, but the main advantage is that we avoid problems of memory size.

Linear Completion	First step		Second step		Example
	Ordering	Positive	Ordering	Positive	
Initial Clauses	12	12	53	19	15
Generated Clauses	111	63	>3336	240	2526
Final Clauses	53	19	>1508	46	258
Resolutions	0	0	0	0	4
Superpositions	14	12	>554	59	783
Cont. Superpositions	20	14	>74	24	125
Ext. Superpositions	9	6	>67	12	748
Deductions	43	32	>695	95	1660
Simplifications	132	109	>4410	413	5086
Deletions	151	133	$\gg$1881	324	3407

These statistics give an idea of the advantage of the positive strategy, but the proportions cannot be generalized. Indeed, the positive strategy may be less powerful if some initial clauses have several negative literals. In addition, if the positive strategy reduces the width of the search space, it increases the depth of the proofs (depth 5 for previous example, while depth 4 with the ordering strategy).

7 Conclusion

In this paper, we have defined an inference system for automated deduction modulo equational theories. This system combines the superposition strategy with a positive ordering strategy to prune the search space. Moreover, we have described a procedure for computing contexts, from the theory $\mathcal{E}$ only, i.e. without the use of the initial set of clauses.
Our system has been proved refutationally complete for a large class of equational theories, including all the regular theories. This and our algorithm for constructing non-redundant contexts are important improvements of previous results of Wertz [31] and Paul [17]. One of our further works is to implement this algorithm and to study theories where there is an infinity of non-redundant contexts.

Our technique of deduction modulo some equations has shown its interest in our theorem prover `DATAC`, for the case of associative and commutative theories. However, for testing it on other theories, we need to study orderings for comparing terms and unification algorithms, since there are very few in the literature. This lack of orderings may be solved by term rewriting techniques as in [7, 22]. Unification algorithms may be solved by term rewriting techniques too, for dealing with parts of these theories such as in [15].
However, it seems that one of the most interesting ways for dealing with these problems of $\mathcal{E}$-unification is to use symbolic constraints, as in [28].

Acknowlegments: I would like to thank Prof. Anita Wasilewska of Stony Brook for the numerous discussions we had on the history of the bases of this paper.

I would like to dedicate this paper to the memory of my colleague Valentin Antimirov of INRIA Lorraine (France), with whom I had very interesting discussions while preparing a first version of this paper.

References

1. L. Bachmair and N. Dershowitz. Completion for Rewriting Modulo a Congruence. *Theoretical Computer Science*, 67(2-3):173–202, October 1989.
2. L. Bachmair and H. Ganzinger. On Restrictions of Ordered Paramodulation with Simplification. In M. E. Stickel, editor, *Proceedings 10th International Conference on Automated Deduction, Kaiserslautern (Germany)*, volume 449 of *Lecture Notes in Computer Science*, pages 427–441. Springer-Verlag, July 1990.
3. L. Bachmair and H. Ganzinger. Rewrite-based Equational Theorem Proving with Selection and Simplification. *Journal of Logic and Computation*, 4(3):1–31, 1994.
4. L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic Paramodulation. *Information and Computation*, 121(2):172–192, 1995.

5. L. Bachmair and D. Plaisted. Associative Path Orderings. In *Proceedings 1st Conference on Rewriting Techniques and Applications, Dijon (France)*, volume 202 of *Lecture Notes in Computer Science*. Springer-Verlag, 1985.
6. D. Brand. Proving Theorems with the Modification Method. *SIAM Journal of Computing*, 4:412–430, 1975.
7. C. Delor and L. Puel. Extension of the Associative Path Ordering to a Chain of Associative Symbols. In C. Kirchner, editor, *Proceedings 5th Conference on Rewriting Techniques and Applications, Montreal (Canada)*, volume 690 of *Lecture Notes in Computer Science*, pages 389–404. Springer-Verlag, 1993.
8. N. Dershowitz and J.-P. Jouannaud. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. Elsevier Science Publishers B. V. (North-Holland), 1990.
9. A. Fortenbacher. *Effizientes Rechnen in AC-Gleichungstheorien*. PhD thesis, Universität Karlsruhe (Germany), February 1989.
10. J. Hsiang and M. Rusinowitch. Proving Refutational Completeness of Theorem Proving Strategies: The Transfinite Semantic Tree Method. *Journal of the ACM*, 38(3):559–587, July 1991.
11. J.-M. Hullot. *Compilation de Formes Canoniques dans les Théories équationelles*. Thèse de Doctorat de Troisième Cycle, Université de Paris Sud, Orsay (France), 1980.
12. J.-P. Jouannaud and C. Kirchner. Solving Equations in Abstract Algebras: a Rule-based Survey of Unification. In Jean-Louis Lassez and G. Plotkin, editors, *Computational Logic. Essays in honor of Alan Robinson*, chapter 8, pages 257–321. MIT Press, Cambridge (MA, USA), 1991.
13. J.-P. Jouannaud and H. Kirchner. Completion of a Set of Rules Modulo a Set of Equations. *SIAM Journal of Computing*, 15(4):1155–1194, 1986. Preliminary version in Proceedings 11th ACM Symposium on Principles of Programming Languages, Salt Lake City (USA), 1984.
14. D. S. Lankford and A. Ballantyne. Decision Procedures for Simple Equational Theories with Associative Commutative Axioms: Complete Sets of Associative Commutative Reductions. Technical report, Univ. of Texas at Austin, Dept. of Mathematics and Computer Science, 1977.
15. C. Marché. *Réécriture modulo une théorie présentée par un système convergent et décidabilité du problème du mot dans certaines classes de théories équationnelles*. Thèse de Doctorat d'Université, Université de Paris-Sud, Orsay (France), October 1993.
16. R. Nieuwenhuis and A. Rubio. Basic Superposition is Complete. In B. Krieg-Brückner, editor, *Proceedings of ESOP'92*, volume 582 of *Lecture Notes in Computer Science*, pages 371–389. Springer-Verlag, 1992.
17. E. Paul (E-mail: etienne.paul@issy.cnet.fr). E-Semantic Tree. Unpublished paper, 70 pages, 1994.
18. G. E. Peterson. A Technique for Establishing Completeness Results in Theorem Proving with Equality. *SIAM Journal of Computing*, 12(1):82–100, 1983.
19. G. E. Peterson and M. E. Stickel. Complete Sets of Reductions for Some Equational Theories. *Journal of the ACM*, 28:233–264, 1981.
20. G. Plotkin. Building-in Equational Theories. *Machine Intelligence*, 7:73–90, 1972.
21. J. A. Robinson. A Machine-oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12:23–41, 1965.
22. A. Rubio and R. Nieuwenhuis. A Precedence-Based Total AC-Compatible Ordering. In C. Kirchner, editor, *Proceedings 5th Conference on Rewriting Techniques*

and Applications, Montreal (Canada), volume 690 of *Lecture Notes in Computer Science*, pages 374–388. Springer-Verlag, 1993.

23. M. Rusinowitch. *Démonstration automatique — Techniques de réécriture.* InterEditions, 1989.
24. M. Rusinowitch. Theorem-proving with Resolution and Superposition. *Journal of Symbolic Computation*, 11:21–49, 1991.
25. M. Rusinowitch and L. Vigneron. Associative Commutative Deduction. In E. Domenjoud and Claude Kirchner, editors, *Proceedings of the 1st CCL Workshop, Le Val d'Ajol (France)*, October 1992.
26. M. Rusinowitch and L. Vigneron. Automated Deduction with Associative-Commutative Operators. *Applicable Algebra in Engineering, Communication and Computing*, 6(1):23–56, January 1995.
27. M. E. Stickel. A Unification Algorithm for Associative-Commutative Functions. *Journal of the ACM*, 28:423–434, 1981.
28. L. Vigneron. Associative-Commutative Deduction with Constraints. In A. Bundy, editor, *Proceedings 12th International Conference on Automated Deduction, Nancy (France)*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 530–544. Springer-Verlag, June 1994.
29. L. Vigneron. *Automated Deduction with Symbolic Constraints in Equational Theories.* PhD Thesis, Université Henri Poincaré - Nancy 1, November 1994. Available as Research Report *CRIN 94-T-266* (in French).
30. L. Vigneron. Theorem Proving modulo Regular Theories. Technical report 95-1, Department of Computer Science, SUNY at Stony Brook, Stony Brook, January 1995.
31. U. Wertz. First-Order Theorem Proving Modulo Equations. Technical Report MPI-I-92-216, Max Planck Institut für Informatik, April 1992.