Theorem Proving modulo Regular Theories

Laurent Vigneron *

Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794, USA
Tel: 516-632-8473, Fax: 516-632-8334
E-mail: Laurent.Vigneron@cs.sunysb.edu

Abstract

We propose a new technique of automated deduction for proving properties by refutation in first-order logic. These proofs are done modulo a set of equations forming a regular theory E.

The inference system defined is based on the superposition strategy, a variant of the paramodulation strategy, and is compatible with simplification rules. It contains a positive ordered strategy to limit the number of possible deductions. Our inference rules avoid the combinatory of a direct use of equations of E by a calculus of replacement contexts and an algorithm for unification modulo E.

The efficiency of our technique is shown, for the case of associative and commutative theories, by an example of execution of our theorem prover `DATAC`.

Keywords

Automated Deduction, Equational Theories, Paramodulation, Superposition, Term Rewriting, Theorem Proving.

This paper has not already been accepted by and is not currently under review for a journal or another conference. Not will it be submitted for such during IJCAI's review period.

*Research supported by the *Institut National de Recherche en Informatique et en Automatique*, Rocquencourt (France)

1 Introduction

Theorem proving with the equality predicate has been considered as a difficult problem. The presence of some particular equations may cause the divergence of deductions. Our solution for dealing with such equations, regrouped in a set E, is to work in the E-congruence classes and to use equality, matching and unification procedures modulo E. This idea of building axioms within the unification procedures has been first initiated by Plotkin [15]. In the context of completion, it has been investigated by Peterson and Stickel [14], Bachmair and Dershowitz [1] and Jouannaud and Kirchner [9]. For theorem proving, it has been investigated by Lankford and Ballantyne [10].

We propose in this paper a procedure for theorem proving modulo equational theories, which generalizes the result of Rusinowitch and Vigneron [18, 19] about associative and commutative theories. Our procedure combines the superposition strategy, a variant of the paramodulation strategy, with a positive ordered strategy. The number of possible deductions is therefore seriously limited, and it guarantees the efficiency of the inference system.

Working modulo a theory E, an important problem is the treatment of extended clauses. We describe a procedure to define all the possible extensions, by computing contexts from the equations in E. This calculus has the advantage to be entirely independent fro the set of clauses to be used. Moreover, since these contexts are used in special inference rules, we never create explicitly extended clauses; it avoids the arrangement of a special control of these extended clauses, which often is very complex.

We prove the refutational completeness of our inference system, for any regular theory E, i.e. if E satisfies the following property:

> For each equation $(l \simeq r)$ in E, the set of variables of l has to be equal to the set of variables of r.

This result is an important improvement of works of Wertz [21] and Paul [13], since the property they require for E is much more restrictive.

The layout of this paper is the following: after introducing the main notations and definitions in Section 2, we describe our inference rules in Section 3. Section 4 presents a procedure to compute contexts for extended equations. The proof of refutational completeness of the inference system is sketched in Section 5, and we give an example of execution of the system `DATAC` for the case of associative and commutative theories in Section 6.

2 Notations and Definitions

Let us define some basic notions, based on the standard notations and definitions for term rewriting and unification given in [6, 8].

Let E be an equational theory, i.e. a set of equations. The congruence generated by this set E is called *E-equality* and written $=_E$.

A *substitution* is a function replacing some variables by terms. A substitution σ is said to *E-unify* two terms s and t if $s\sigma$ and $t\sigma$ are E-equal.

An *atom* is an equality $(l \simeq r)$, where l is assumed to be greater than or equal to r. A *clause* is denoted $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$, where $A_1, \ldots, A_n$, $B_1, \ldots, B_m$ are atoms; it 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 $\vdash$. A *clause is positive* if it contains only positive literals, i.e. if the left-hand side of $\vdash$ 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 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 ordered strategy. So, we assume we are given a *simplification ordering* $>$, defined on terms and atoms. For sake of completeness, it has to be *total* on ground E-congruence classes and *E-compatible*, i.e.

$$\forall s, t \text{ ground terms, if } s > t \text{ and } s \neq_E t, \text{ then } \forall s' =_E s,\ \forall t' =_E t,\ s' > t'$$

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.

A set of clauses is said to be *E-incoherent* if it has no model coherent with the theory E.

3 Inference Rules

We describe in this section a set of inference rules for applying deductions modulo an equational theory 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 ordered 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 Ordered 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 E-equal left-hand sides. This inference rule is essential for applying entirely the superposition strategy, and note that it is applied only on positive clauses.

Definition 2 (Equational *E*-Factoring)

$$\frac{\vdash (l_1 \simeq r_1), (l_2 \simeq r_2), R}{(r_1 \simeq r_2)\sigma \vdash (l_2 \simeq r_2)\sigma, R\sigma}$$

This rule applies if σ E-unifies l_1 and l_2, $(l_1 \simeq r_1)\sigma$ is greater than or equal to $(l_2 \simeq r_2)\sigma$ and each equation of $R\sigma$. Moreover, $l_1\sigma$ has to be greater than $r_1\sigma$.

The next rule avoids to add 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 (*E*-Reflexion)

$$\frac{(l \simeq r), L \vdash R}{L\sigma \vdash R\sigma}$$

This rule applies if σ E-unifies l and r, and if $(l \simeq r)\sigma$ is greater than or equal to 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 *left* and *right* rules, whether it is applied on the left-hand side or on the right-hand side of $\vdash$.

Definition 4 (Left *E*-Superposition)

$$\frac{\vdash (l_1 \simeq r_1), R_1 \qquad (l_2 \simeq r_2), L_2 \vdash R_2}{(l_2[r_1]_{p_2} \simeq r_2)\sigma, L_2\sigma \vdash R_1\sigma, R_2\sigma}$$

This rule applies if p_2 is a non-variable position in l_2, σ E-unifies $l_2|_{p_2}$ and l_1, $(l_1 \simeq r_1)\sigma$ is maximal for $>$ in its clause, and $(l_2 \simeq r_2)\sigma$ is greater than or equal to each atom of $L_2\sigma$. Moreover, $l_1\sigma$ has to be greater than $r_1\sigma$, and $l_2\sigma$ has to be greater than $r_2\sigma$.

Definition 5 (Right *E*-Superposition)

$$\frac{\vdash (l_1 \simeq r_1), R_1 \qquad \vdash (l_2 \simeq r_2), R_2}{\vdash (l_2[r_1]_{p_2} \simeq r_2)\sigma, R_1\sigma, R_2\sigma}$$

This rule applies if p_2 is a non-variable position in l_2, σ E-unifies $l_2|_{p_2}$ and l_1, $(l_1 \simeq r_1)\sigma$ is maximal for $>$ in its clause, and $(l_2 \simeq r_2)\sigma$ is maximal in its clause and greater than $(l_1 \simeq r_1)\sigma$. Moreover, $l_1\sigma$ has to be greater than $r_1\sigma$, and $l_2\sigma$ has to be greater than $r_2\sigma$.

For deduction modulo an equational theory E, these four inference rules are not complete, as shown in the next example.

Example 1 Let $E = \{\ (f(f(x,y),z) \simeq f(x,f(y,z)))\ \}$. The following clauses

$$\vdash (f(a,b) \simeq c) \qquad \vdash (f(a,f(b,d)) \simeq e) \qquad (f(c,d) \simeq e) \vdash$$

form an incoherent set with E, since: E permits to modify the parentheses in the left-hand side of the second clause, to obtain the term $f(f(a,b),d)$, and replacing $f(a,b)$ by c in this term (thanks to the first equation), we deduce $\vdash (f(c,d) \simeq e)$ which is in contradiction with the third clause.
However, there is no possible inference step between the 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 couple (e,p) is named *context*. In previous example, a contradiction can be derived using the context $(f(f(x,y),z),1)$, producing the extended equation $(f(f(a,b),z) \simeq f(c,z))$ from $(f(a,b) \simeq c)$.

By the Critical Pairs Lemma of Jouannaud and Kirchner [9], we know that contexts can be computed. We define in Section 4 a procedure to compute all the possible contexts for a given equational theory E. Given a term l, $\mathcal{C}ont(l)$ is the set of all contexts (e,p) for E such that $e|_p$ and l are E-unifiable. These contexts are used in three new inference rules.

The first two rules simulate replacements from an extended equation.

Definition 6 (Left Contextual *E*-Superposition)

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

The single difference with the Left E-Superposition is that we choose a context (e_1,p_1) in $\mathcal{C}ont(l_1)$, and σ E-unifies $e_1[l_1]_{p_1}$ and $l_2|_{p_2}$. The replacing term in the deduced clause is the extension of the right-hand side, $e_1[r_1]_{p_1}$.

Definition 7 (Right Contextual *E*-Superposition)

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

The single difference with the Right E-Superposition is that we choose a context (e_1, p_1) *in* $\mathcal{C}ont(l_1)$*, and* σ *E-unifies* $e_1[l_1]_{p_1}$ *and* $l_2|_{p_2}$*. The replacing term in the deduced clause is the extension of the right-hand side,* $e_1[r_1]_{p_1}$*.*

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

Definition 8 (Extended *E*-Superposition)

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

This rule applies if, given a context (e_1, p_1) *in* $\mathcal{C}ont(l_1)$ *and a context* (e_2, p_2) *in* $\mathcal{C}ont(l_2)$*,* σ *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 greater than their right-hand side.*

Remark : The principle of the superposition strategy, *to apply replacements only into maximal sides of equations*, is very old, since it has been used for term rewriting and completion. But, the completeness of inference systems representing this strategy has been a long standing open problem. For completeness, either some deductions using non-maximal literals [17], or some replacements into minimal sides of equations [3], were needed. Nieuwenhuis and Rubio [12] proved the completeness of the entire superposition strategy by adding the *Equational Factoring* rule, and defining a particular ordering for comparing negative and positive literals.
However, in our inference rules, we never need to compare such literals: we always compare literals of the same sign. So, specifying a special ordering on literals is useless.

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

4 Extended Equations

An extension of an equation $(l \simeq r)$ is an equation $(e[l]_p \simeq e[r]_p)$, also called *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 E-unifiable with l, 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 E, written $\mathcal{C}_E$, is defined by $\bigcup_{k\geq 0} \mathcal{C}ont_k$, where sets $\mathcal{C}ont_k$ are inductively defined by

$$\begin{array}{l} \mathcal{C}ont_0 \;=\; \{\ (e,p) \quad | \quad \exists (e \simeq e') \text{ or } (e' \simeq e) \in E,\ p \text{ is a non-variable} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{position in } e \text{ and } p \neq \epsilon\ \} \\ \mathcal{C}ont_{k+1} \;=\; \{\ (e_1[e_2]_{p_1}, p_1{\cdot}p_2) \quad | \quad (e_1,p_1) \in \mathcal{C}ont_0,\ (e_2,p_2) \in \mathcal{C}ont_k, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{and } e_1|_{p_1} \text{ and } e_2 \text{ are } E\text{-unifiable}\ \} \end{array}$$

Then, given an equation $(l \simeq r)$ where l is greater than r, the set of all possible contexts which can extend $(l \simeq r)$ is defined by:

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

The definition of $\mathcal{C}_E$ 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 9 *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. *$e_2[\cdot]_{p_2}\sigma =_E (e_1|_p)[\cdot]_q$ to guaranty the equivalence of terms $e_2\sigma$ and $e_1|_p$,*
2. *$(e_2|_{p_2})\sigma =_E e_1|_{p_1}$ to guaranty the equivalence of subterms where replacements will apply.*

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

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.

The main notions being defined, let us describe the redundancy criteria of a context in the theory E.

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

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

Lemma 11 *Let E be an equational theory. The set of contexts $\mathcal{C}_E$, constructed with the E-redundancy criteria, is sufficient to express the whole theory E.*

The proof of this Lemma is done in a similar way to the proof of the Critical Pairs Lemma of Jouannaud and Kirchner [9]. By lack of place, it is not detailed in this paper, but it is in [20].

Let us give two examples of construction of contexts.

Example 2 (Associativity and Commutativity) If E represents properties of associativity and commutativity of an operator f,

$$E \;=\; \{\; (f(f(x,y),z) \simeq f(x,f(y,z))),\; (f(x,y) \simeq f(y,x)) \;\}$$

$\mathcal{C}ont_0$ contains the two contexts

$$(f(f(x,y),z),1) \text{ and } (f(x,f(y,z)),2),$$

but the second one is top-redundant by the first one. The unique context of $\mathcal{C}ont_1$, $(f(f(f(x,y),z),v),1{\cdot}1)$, is top-redundant by $(f(f(x,y),z),1)$ too. Hence, $\mathcal{C}_{AC} = \{\; (f(f(x,y),z),1) \;\}$, which means that the only possible extension of an equation $(l \simeq r)$ is $(f(l,z) \simeq f(r,z))$. $\diamond$

Example 3 (Associativity) If E contains the property of associativity of an operator f,

$$E \;=\; \{\; (f(f(x,y),z) \simeq f(x,f(y,z))) \;\}$$

the non-redundant contexts are

$$\begin{aligned}
\mathcal{C}ont_0 &= \{\; (f(f(x,y),z),1),\; (f(x,f(y,z)),2) \;\} \\
\mathcal{C}ont_1 &= \{\; (f(f(x,f(y,z)),v),1{\cdot}2) \;\} \\
\mathcal{C}ont_2 &= \emptyset
\end{aligned}$$

This means that to extend an equation $(l \simeq r)$ consists in adding a new variable, either on the right, $(f(l,z) \simeq f(r,z))$, or on the left, $(f(x,l) \simeq f(x,r))$, or on both sides, $(f(f(x,l),v) \simeq f(f(x,r),v))$. $\diamond$

Remark : In the construction of contexts as in their use in inference rules, we always have used a test of E-unifiability between terms. However, we can add a third element to each context: the conjunction of the E-unification problems encountered to construct it. A first consequence is that a context is created only if its unification problems admit at least one solution. As second consequence, to apply 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.
Therefore, less contexts are constructed, less inferences rules with contexts are applicable and their unification problems have less solutions.
This case is completely developed in [20].

5 Refutational Completeness

Let *INF* be the set of the 7 inference rules described in Section 3. Let us state the theorem of refutational completeness of *INF*.

Theorem 12 (Completeness) *Let E be a regular equational theory, and let S be a set of clauses. If S is E-incoherent, INF always will derive a contradiction from S.*

This theorem states that our inference system is refutationally complete for every regular equational theory, i.e. every theory whose equations preserve variables. This result is an important improvement of previous works of Wertz [21] and Paul [13], since they require the following property for E:

> Given ground terms s and t, such that t is a strict subterm of s $(s = s[t])$. If $s[t]$ and t are E-equal, then for each term t', $s[t']$ has to be E-equal to t'.

This condition implies the regularity of E, but many regular theories do not satisfy this condition. For instance, if E is $\{(f(x,0) \simeq x)\}$, E is regular but: given the ground term $f(0,0)$, it is E-equal to 0; however, for a constant a, $f(0,a)$ is not E-equal to a. The condition of Wertz and Paul is not satisfied. Moreover, our inference system limits the number of possible deductions much more than Wertz' and Paul's systems, thanks to the positive strategy.

We prove the Theorem of Completeness by the transfinite semantic tree method of Hsiang and Rusinowitch [16, 7], extended to deduction modulo an equational theory in [20]. Let us give a sketch of this proof.

Sketch of Proof: *Let E be a regular theory. Let S be a set of clauses, incoherent for E. We construct the transfinite semantic tree for the empty theory. By E-incoherence of S, each branch of this tree falsifies either an instance of a clause of S, or a trivial equation $(t \simeq t')$ where $t =_E t'$. We say that each branch is cut by a clause at the level where it falsifies it. To derive a contradiction from S means to reduce the semantic tree into an empty tree, thanks to clauses deduced by INF.*

We prove that INF always derives a contradiction in the following way: we assume the semantic tree cannot be reduced into an empty tree; then, we select a branch in the remaining subtree and we show this branch has to be empty. It implies that the semantic tree has to be empty too. □

Our inference system *INF* is compatible with simplification rules, whose purpose is to delete redundant clauses, if the derivations done are fair, i.e. do not forget definitively a possible deduction. Some examples of simplification rules are:

- E-Simplification: it consists in applying a term rewriting step, using a procedure of matching modulo E.
- Clausal E-Simplification: if there is a clause $\vdash A$ (resp. $A \vdash$), then each clause of the form $A', L \vdash R$ (resp. $L \vdash A', R$), where A' is E-equal to an instance of A, is replaced by $L \vdash R$.
- Tautology Deletion: each clause of the form $L \vdash (t \simeq t'), R$ where $t =_E t'$, or $A, L \vdash A', R$ where $A =_E A'$, is deleted.

6 Implementation

The inference system described in this paper is implemented in the system `DATAC` for the case where E represents the properties of associativity and commutativity, or only commutativity, of operators.

`DATAC` is a theorem prover written in CAML Light, a functional language of the ML family. It runs on SUN, HP and IBM PC workstations.

Let us detail an example of execution with `DATAC`. The purpose of this example is to prove a property of the operator *star* $(^*)$ in Kleene algebras

$(K, +, \cdot, 0, 1, ^*)$, where the operator $+$ is associative and commutative. The equational theory E is therefore:

$$\{\ ((x_1 + x_2) + x_3 \simeq x_1 + (x_2 + x_3)),\ (x_1 + x_2 \simeq x_2 + x_1)\ \}$$

and, as shown in Example 2, the unique context for E is: $((x_1 + x_2) + x_3, 1)$. Kleene algebras are expressed by the following clauses:

- $(K, +, 0)$ is an idempotent commutative monoïd: ($+$ is AC)

  (1) $\vdash\ (x_1 + 0 \simeq x_1)$
  (2) $\vdash\ (x_1 + x_1 \simeq x_1)$

- $(K, \cdot, 1)$ is a monoïd:

  (3) $\vdash\ (x_1 \cdot (x_2 \cdot x_3) \simeq (x_1 \cdot x_2) \cdot x_3)$
  (4) $\vdash\ (1 \cdot x_1 \simeq x_1)$
  (5) $\vdash\ (x_1 \cdot 1 \simeq x_1)$

- $(K, +, \cdot, 0, 1)$ is an idempotent semi-ring:

  (6) $\vdash\ (x_1 \cdot (x_2 + x_3) \simeq (x_1 \cdot x_2) + (x_1 \cdot x_3))$
  (7) $\vdash\ ((x_1 + x_2) \cdot x_3 \simeq (x_1 \cdot x_3) + (x_2 \cdot x_3))$
  (8) $\vdash\ (0 \cdot x_1 \simeq 0)$
  (9) $\vdash\ (x_1 \cdot 0 \simeq 0)$

- Definition of the operator *star*:

  (10) $\vdash\ ((x_1 \cdot x_1^*) + 1 \simeq x_1^*)$
  (11) $\vdash\ ((x_1^* \cdot x_1) + 1 \simeq x_1^*)$
  (12) $((x_1 \cdot x_2) + x_2 \simeq x_2)\ \vdash\ ((x_1^* \cdot x_2) + x_2 \simeq x_2)$
  (13) $((x_1 \cdot x_2) + x_1 \simeq x_1)\ \vdash\ ((x_1 \cdot x_2^*) + x_1 \simeq x_1)$

The property we want to prove is: $\forall x_1,\ x_1^{**} = x_1^*$. In this purpose, we add a new clause, which is the skolemization of the negation of this property: (a is a new constant)

(P) $(a^{**} \simeq a^*)\ \vdash$

The theorem prover `DATAC` is run with these 14 initial clauses, and uses the *Associative Path Ordering* of [4, 5] for comparing terms, with the following precedence on operators: $\cdot >_{\mathcal{F}} + >_{\mathcal{F}}\ ^* >_{\mathcal{F}} 1 >_{\mathcal{F}} 0$.
Deductions are applied thanks to inference rules defined in Section 3, combining the positive strategy with the superposition strategy. It derives entirely automatically a contradiction, the empty clause written $\Box$, in the following way:

Extended E-Superposition between *2* and *10*

(15) $\vdash\ (1+(x_1 \cdot x_1^*) \simeq x_1^* + 1)$

E-Simplification from *10* into *15*

(15) $\vdash\ (x_1^* + 1 \simeq x_1^*)$

Extended E-Superposition between *10* and *15*

(52) $\vdash\ (x_2^* + (x_1 \cdot x_1^*) \simeq x_1^* + x_2^*)$

Right E-Superposition from *15* into *6*

(63) $\vdash\ ((x_1 \cdot 1) + (x_1 \cdot x_2^*) \simeq x_1 \cdot x_2^*)$

E-Simplification from *5* into *63*

(63) $\vdash\ (x_1 + (x_1 \cdot x_2^*) \simeq x_1 \cdot x_2^*)$

E-Simplification from *63* into *13*

(13) $((x_1 \cdot x_2) + x_1 \simeq x_1)\ \vdash\ (x_1 \cdot x_2^* \simeq x_1)$

Right E-Superposition from *15* into *7*

(69) $\vdash\ ((1 \cdot x_2) + (x_1^* \cdot x_2) \simeq x_1^* \cdot x_2)$

E-Simplification from *4* into *69*

(69) $\vdash\ (x_1 + (x_2^* \cdot x_1) \simeq x_2^* \cdot x_1)$

E-Simplification from *69* into *12*

(12) $((x_1 \cdot x_2) + x_2 \simeq x_2)\ \vdash\ (x_1^* \cdot x_2 \simeq x_2)$

Left E-Superposition from *52* into *12*

(125) $(x_1^* + x_1^* \simeq x_1^*)\ \vdash\ (x_1^* \cdot x_1^* \simeq x_1^*)$

Clausal E-Simplification in *125* thanks to *2*

(125) $\vdash\ (x_1^* \cdot x_1^* \simeq x_1^*)$

Extended E-Superposition between *10* and *63*

(135) $\vdash\ ((x_1 \cdot x_1^*) + 1 \simeq x_1^* + x_1)$

E-Simplification from *10* into *135*

(135) $\vdash\ (x_1^* + x_1 \simeq x_1^*)$

Left E-Superposition from *125* into *13*

(161) $(x_1^* + x_1^* \simeq x_1^*) \vdash (x_1^* \cdot x_1^{**} \simeq x_1^*)$

Clausal E-Simplification in *161* thanks to *2*

(161) $\vdash (x_1^* \cdot x_1^{**} \simeq x_1^*)$

Right E-Superposition from *161* into *69*

(473) $\vdash (x_1^* \cdot x_1^{**} \simeq x_1^{**} + x_1^*)$

E-Simplification from *135* into *473*

(473) $\vdash (x_1^* \cdot x_1^{**} \simeq x_1^{**})$

E-Simplification from *161* into *473*

(473) $\vdash (x_1^{**} \simeq x_1^*)$

Clausal E-Simplification in P thanks to *473*

(P) $\square$

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 ordered strategy to prune the search space. Moreover, we have described a procedure for computing contexts, from the theory E only, i.e. without need of the initial set of clauses.
Our system has been proved refutationally complete for regular theories. This is an important improvement of previous results of Wertz [21] and Paul [13]. At last, our system is implemented in DATAC and we have given an example of proof of property.

Our technique of deduction modulo some equations has shown its large interest in our theorem prover DATAC, for the case of associative and commutative theories. However, to test it on other theories, we need to study orderings for comparing terms, since they are very few in the literature. This lack of ordering may be solved by term rewriting techniques for dealing with parts of these theories, such as in [11].

In another point of view, our inference rules apply replacements thanks to the equality predicate. It would be interesting to generalize them to a transitive relation, as used in [2].

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. Associative-Commutative Superposition. Technical report MPI-I-93-267, Max Planck Institut für Informatik, Saarbrücken, 1993.

[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 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] D. S. Lankford and A. Ballantyne. Decision Procedures for Simple Equational Theories with Permutative Axioms: Complete Sets of Permutative

Reductions. Technical report, Univ. of Texas at Austin, Dept. of Mathematics and Computer Science, 1977.

[11] 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.

[12] 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.

[13] E. (E-mail: etienne.paul@issy.cnet.fr) Paul. E-Semantic Tree. Unpublished paper, 70 pages, 1994.

[14] G. E. Peterson and M. E. Stickel. Complete Sets of Reductions for Some Equational Theories. *Journal of the ACM*, 28:233–264, 1981.

[15] G. Plotkin. Building-in Equational Theories. *Machine Intelligence*, 7:73–90, 1972.

[16] M. Rusinowitch. *Démonstration automatique — Techniques de réécriture.* InterEditions, 1989.

[17] M. Rusinowitch. Theorem-proving with Resolution and Superposition. *Journal of Symbolic Computation*, 11:21–49, 1991.

[18] M. Rusinowitch and L. Vigneron. Automated Deduction with Associative Commutative Operators. In Ph. Jorrand and J. Kelemen, editors, *Fundamental of Artificial Intelligence Research*, volume 535 of *Lecture Notes in Computer Science*, pages 185–199. Springer-Verlag, 1991.

[19] M. Rusinowitch and L. Vigneron. Automated Deduction with Associative-Commutative Operators. *Applicable Algebra in Engineering, Communication and Computing*, 6(1):23–56, January 1995.

[20] L. Vigneron. *Déduction automatique avec contraintes symboliques dans les théories équationnelles.* Thèse de Doctorat d'Université, Université Henri Poincaré - Nancy 1, November 1994. Available as research report *CRIN 94-T-266.*

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