Laboratoire
Lorrain
de Recherche
en Informatique
et ses Applications

Automated Deduction Techniques for Studying Rough Algebras

L. Vigneron

LORIA 98-R-140

Published in Fundamenta Informaticae, 33(1):85-103, February 1998

Bâtiment LORIA, Campus Scientifique, B.P. 239, F-54506 Vandœuvre-lès-Nancy CEDEX

Automated Deduction Techniques for Studying Rough Algebras

Laurent Vigneron
LORIA – Université Nancy 2
B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France
E-mail: vigneron@loria.fr

Abstract

We present a non-exhaustive state of the art in the domain of deduction in first-order logic with equality, describing different strategies introduced over the years: strategies for selecting deduction steps, for eliminating redundant information, for delaying the resolution of unsolved or too difficult problems, for applying deductions with built-in properties. We also present the system daTac that implements our deduction techniques modulo associative-commutative properties. Finally we detail an application of those techniques to the study of non-classical logics, work realised in collaboration with Prof. Wasilewska (Stony Brook University, New York).

Keywords: automated deduction, theorem proving, equational theories, first-order logic, constraint deduction, modal logic, rough sets.

1 Introduction

Designing new methods and strategies has been for many years a main preoccupation for researchers in automated reasoning for first-order logic. A first step has been done by J.A. Robinson [31] by introducing a proof system for classical logic based on a *resolution* rule. The next step was the addition of a *paramodulation* rule done by G.A. Robinson and Wos [30]. The paramodulation rule permits to deal with the equality predicate without describing explicitly its properties. It is also based on a notion of replacement. Over the time, several refinements have been added to this rule. Brand [7] has shown that only the reflexivity axiom ($x \simeq x$) is needed. Peterson [27] has shown that paramodulations into variables are useless. But the crucial step has been achieved by Hsiang and Rusinowitch [9]: they have introduced ordering restrictions to the application of these rules and proved the completeness of the *ordering strategy* defined as follows: *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 and clausal simplifications, have followed.

The complete Hsiang-Rusinowitch strategy and others are often unfortunately inefficient in presence of properties producing the divergence of derivations.

The first established solution was proposed by Plotkin [29]. He proposed to define an equational theory E, by extracting above properties from the set of clauses, and to use a *unification algorithm modulo* E. This result has been the basis of many works: Lankford and Ballantyne [14] for the particular case of associative and commutative theories, and Peterson and Stickel [28] for completion.

We present in this paper a non-exhaustive state of the art in this domain of deduction in first-order logic with equality, describing the different strategies introduced over the years: ordering strategies, superposition strategy, simplification strategies, constraint strategies. After showing Plotkin's works on automated deduction modulo an equational theory, we present our recent works in this domain. Then we describe the system daTac, that implements our deduction techniques for associative-commutative theories. Finally we detail an application of those techniques to the study of non-classical logics, work realised in collaboration with Prof. Wasilewska (Stony Brook University, New York).

2 Basic Notions

Let us define the basic notions used in this paper, based on the standard notations and definition for term rewriting and unification given in [8, 11].

We are given a set $\mathcal{F}$ of functional operators (f, g, ...), a set $\mathcal{P}$ of predicate operators (p, q, ...), and a set $\mathcal{X}$ of variables (x, y, ...). Constants from $\mathcal{F}$ will be denoted by a, b, c, ...
A *term* is a formula composed with elements of $\mathcal{F}$ and $\mathcal{X}$. A term is said to be *ground* if it does not contain any variable. Terms are denoted by s, t, ... An *atom* (A, B, ...) is a formula $p(t_1, \ldots, t_n)$ where $p \in \mathcal{P}$, and the t_i are terms. An *equation* is an atom whose predicate operator is $\simeq$. In the following, the equality operator being commutative, we will not distinguish the atoms $(s \simeq t)$ and $(t \simeq s)$. A *literal* (L) is a positive (A) or a negative ($\neg A$) atom. A *clause* (C, D, ...) is a disjunction of literals: $L_1 \vee \ldots \vee L_n$.
A term s is a *subterm* of a term t, denoted by $t[s]$, if s occurs at an inside position of t. Considering a term $t[s']$ means replacing s by s' in $t[s]$. The set of variables of a term t is denoted by $\mathcal{V}ar(t)$. A *substitution* (σ) is a mapping from variables to terms. $\sigma(t)$ denotes the term resulting from the replacement of the variables of t by the associated terms. $\sigma(L_1 \vee \ldots \vee L_n)$ (resp. $\sigma(\neg A)$) is equivalent to $\sigma(L_1) \vee \ldots \vee \sigma(L_n)$ (resp. $\neg\sigma(A)$). A substitution σ *unifies* terms t_1 and t_2 if the terms $\sigma(t_1)$ and $\sigma(t_2)$ are syntactically equal (denoted by $\sigma(t_1) = \sigma(t_2)$). A substitution σ is a *matching* of the term t_1 in the term t_2 if $\sigma(t_1) = t_2$.

Given a set of axioms E, called an equational theory, a substitution σ is an *E-unifier* of terms t_1 and t_2 if $\sigma(t_1) =_E \sigma(t_2)$, i.e. if $\sigma(t_1)$ and $\sigma(t_2)$ are equivalent in the theory E. A substitution σ is an *E-matching* of the term t_1 in the term t_2 if $\sigma(t_1) =_E t_2$.

A deduction rule whose premises are clauses C_1 and C_2 and whose conclusion is the clause C is denoted by:

$$\frac{C_1 \qquad C_2}{C}$$

3 Deduction Techniques

In the following, we present deduction techniques developed for applying deductions in the first-order logic with the equality predicate. These techniques are based on the resolution and paramodulation rules.

3.1 Deduction in First-Order Logic

An essential step in the automated deduction field has been achieved in 1965 when J.A. Robinson has defined the *resolution* rule [31]. The principle of the resolution rule is to identify two opposite literals in two clauses, and to deduce the new clause composed with the disjunction of the other literals of these clauses.

Definition 3.1 (Resolution) $\dfrac{A_1 \vee D_1 \qquad \neg A_2 \vee D_2}{\sigma(D_1 \vee D_2)}$
where σ unifies A_1 and A_2, i.e. $\sigma(A_1) = \sigma(A_2)$.

A less formal interpretation of this rule is the following: the first rule means "`if not` A_1`, then` D_1"; the second rule means "`if` A_2`, then` D_2". If A_1 and A_2 are identical, this is natural to conclude "D_1 `or` D_2".

The resolution rule is usually defined together with a rule called *factoring*. Its principle is to eliminate multiple occurrences of literals. So, applying the factoring rule consists in generating clauses where multiple occurences of the same literal have been deleted.

Definition 3.2 (Factoring) $\dfrac{L_1 \vee L_2 \vee \ldots \vee L_n \vee D_1}{\sigma(L_1 \vee D_1)}$
where σ unifies L_1, L_2, $\ldots$, L_n.

3.2 Deduction with Equality

When using the equality predicate ($\simeq$), one should describe all its properties:

Reflexivity:	$(x \simeq x)$
Symmetry:	$\neg(x \simeq y) \vee (y \simeq x)$
Transitivity:	$\neg(x \simeq y) \vee \neg(y \simeq z) \vee (x \simeq z)$
$\mathcal{F}$-Reflexivity:	$\forall f \in \mathcal{F}, \forall i,\ \neg(x_i \simeq y_i) \vee (f(\ldots, x_i, \ldots) \simeq f(\ldots, y_i, \ldots))$
$\mathcal{P}$-Reflexivity:	$\forall p \in \mathcal{P}, \forall i,\ \neg(x_i \simeq y_i) \vee \neg p(\ldots, x_i, \ldots) \vee p(\ldots, y_i, \ldots)$

Using all these properties as clauses would lead to a terribly inefficient deduction system.

The *paramodulation* rule has been introduced by G.A. Robinson and Wos [30] for replacing the properties of the equality predicate. This rule introduces the notion of replacement of a term by an equal one.

Definition 3.3 (Paramodulation) $\dfrac{(t_1 \simeq t_2) \vee D_1 \qquad L_2[t'_1] \vee D_2}{\sigma(L_2[t_2] \vee D_1 \vee D_2)}$
where σ unifies t_1 and t'_1.

The subterm t'_1 of L_2 is identified with the left-hand side of the equation $(t_1 \simeq t_2)$ (thanks to the substitution σ), and then replaced by the right-hand side of this equation.

But even with this paramodulation rule, the reflexivity axiom has to be explicitely kept. An additional rule has been defined for simulating the role of this reflexivity property of the equality predicate. This rule is called *reflexion* (or *trivialisation*). It permits to eliminate a negative equation whose both sides can be unified.

Definition 3.4 (Reflexion) $\dfrac{(t_1 \simeq t_2) \vee D_1}{\sigma(D_1)}$
where σ unifies t_1 and t_2.

3.3 Completeness

When defining a deduction system, this is important to prove its completeness.

Theorem 3.1 *The resolution, factoring, paramodulation and reflexion rules form a refutationally complete deduction system.*

A deduction system is said to be *refutationally complete* if it will always derive a contradiction from an inconsistent system. This means, given a set of clauses S and a clause C, if C is a logical consequence of S, the deduction system will always derive a contradiction from S and the negation of C. This contradiction is represented by a clause without literals, called the *empty clause*.

4 Deduction Strategies

In this section, we present several strategies whose purpose is to guide the deduction steps. Ordering strategies permit to select literals in clauses. The superposition strategy describes a variant of the paramodulation rule. Simplification strategies eliminate or transform redundant clauses. Constraint strategies keep a track of all the choices done.
All these strategies can be combined without endangering the completeness result presented in the previous section.

4.1 Ordering Strategies

For getting an efficient deduction system, this is essential to define strategies for selecting deduction steps and literals in which to apply them. For this purpose, we can define a measure of complexity over terms and use it for comparing them. Such a measure is called an ordering. We present below two strategies of selection of literals.

Ordering Strategy. The ordering strategy has been defined by Hsiang and Rusinowitch [9]. It is based on a *total simplification ordering* $>$ over terms, i.e. an ordering satisfying the following properties:

- Well-founded (there is no infinite decreasing sequence),
- Monotonic (if $s > t$, then $u[s] > u[t]$),
- Stable by subterm ($t[s] > s$),
- Stable by instanciation (if $s > t$, then $\sigma(s) > \sigma(t)$),
- Total on ground terms ($\forall s, t$ ground, either $s > t$, or $t > s$, or $s = t$).

The definition of this strategy consists of two laws:

1. To apply deduction steps only between maximal literals of clauses,
2. For a paramodulation step, the replaced term (left-hand side of the equation) has to be greater than the replacing one (right-hand side).

The principle of this last point is to use equations as rewrite rules.

Positive Strategy. The positive strategy is a variant of the ordering strategy. Its motivation is the following: for deriving the empty clause, i.e. finding a contradiction, literals have to be eliminated from clauses. There are only two rules eliminating literals: the resolution and the reflexion. We can remark both of them eliminate at least a negative literal. So, if we avoid to generate many clauses with negative literals, searching a contradiction should be easier.
The principle of the positive strategy is precisely to avoid generating clauses with negative literals: a deduction can use a clause with negative literals provided that at least one of these negative literals is involved in the deduction. The definition of this strategy can be formulated as follows: for using a positive literal in a deduction step, it has to belong to a positive clause, i.e. a clause without any negative literal.
The conditions of maximality of the literals involved in a deduction step are similar to those of the ordering strategy, except for negative literals: a negative literal will have to be maximal w.r.t. the other negative literals of its clause; there is no condition with the positive literals of its clause.

4.2 Superposition Strategy

The superposition strategy is based on the definition of the *superposition* rule, a variant of the paramodulation rule. This superposition rule applies replacements using equations, like the paramodulation rule. But when a replacement is applied to an equation, with the superposition strategy, it has to be applied in the greatest side of the equation.

For preserving the completeness result, we have to define an additional rule, the *equational factoring*. This rule transforms clauses so that the left-hand side of the maximal equation, i.e. its greatest side, does not occur in any other equation.

Definition 4.1 (Equational Factoring) $\dfrac{(t_1 \simeq t_2) \vee (t'_1 \simeq t_3) \vee D_1}{\sigma((t'_1 \simeq t_3) \vee \neg(t_2 \simeq t_3) \vee D_1)}$
where σ unifies t_1 and t'_1, the equation $(t_1 \simeq t_2)$ is maximal in the clause, and $\sigma(t_1)$ is greater than $\sigma(t_2)$ and $\sigma(t_3)$.

4.3 Simplification Strategies

To be able to simplify a set of clauses is an essential task for efficiency. Simplification consists in replacing a clause by a simpler one, or even deleting useless clauses.

Transforming a clause into a simpler one can be done by term rewriting. The rule corresponding to this technique is called *demodulation* (or *simplification*). For example, given an equation $(t_1 \simeq t_2)$ where t_1 is greater than t_2, each clause of the form $C[\sigma(t_1)]$ can be replaced by the clause $C[\sigma(t_2)]$.
A clause can be deleted if it contains a positive equation $(t \simeq t)$, or the same atom appearing both as a positive and as a negative literal. Another deletion technique is the *subsumption*, which can be schematised as follows: if a clause C_1 belongs to a set of clauses S, then any clause of the form $\sigma(C_1) \vee D$ can be removed from S.

Note that in all those simplification and deletion rules, if a substitution has to be applied to variables, it always concerns the variables of the clause used for the simplification. The substitution never transforms variables of the simplified clause. The computation of such a substitution is called a matching problem.

4.4 Constraint Strategies

Constraints can be added to clauses for keeping a track of all the choices done. For instance, the maximality of a literal in a clause may be impossible to decide. However, we can apply the deduction step with this literal, and add this maximality condition as a constraint of the deduced clause. When this last clause is used in a deduction step, we check whether it does not falsify the maximality constraint.
Constraints can also be used for avoiding useless deductions. In the following, we present two constraint strategies: the *basic strategy* and the *constraint strategy*.

Basic Strategy. The principle of the basic strategy [5], extension of the basic narrowing from Hullot [10], is to forbid paramodulation into alien terms. An alien term in a clause is a term introduced by a substitution in a previous deduction step. For instance, considering the paramodulation step (with the unifier $\{x \mapsto a,\ y \mapsto c\}$)

$$\frac{(f(a,y) \simeq y) \qquad p(f(x,c)) \vee q(x,b)}{p(\boxed{c}) \vee q(\boxed{a}, b)}$$

$f(x,c)$ is replaced by c, assuming that $p(f(a,c)) > q(a,b)$. The subterms a and c of the deduced clause are alien terms, since they have been introduced by the substitution. They are blocked (framed) to show they cannot be reached by a paramodulation step.
With constraints, the deduced clause is:

$$p(y) \vee q(x,b)\ \ [\![p(f(a,c)) >^{?} q(a,b)\ \wedge\ x = a\ \wedge\ y = c]\!]$$

This representation with constraints shows the isolation of the alien terms, as they do not appear any more in the body of the clause.

Constraint Strategy. The constraint strategy [13, 19] differs from the basic strategy by the fact that it does not solve the unification problem. This problem is added as a constraint for the deduced clause; we only have to check its satisfiability.
So, we associate a conjunction of constraints to each clause. When a new clause is deduced, it inherits the constraints of its parents, and we add the new constraints introduced by the deduction step. The example detailed for the basic strategy becomes:

$$\frac{(f(a,y) \simeq y)\ [\![T_1]\!] \qquad p(f(x,c)) \vee q(x,b)\ [\![T_2]\!]}{p(y) \vee q(x,b)\ [\![T_1 \wedge T_2 \wedge T_3]\!]}$$

where T_3 contains the unification constraint $f(x,c) =^{?} f(a,y)$, and the ordering constraint $p(f(x,c)) >^{?} q(x,b)$ for representing an unsolved condition of the ordering strategy.

5 Deduction modulo Equational Theories

For applying deductions in an equational theory, the first step is to decompose the initial set of clauses onto two parts: the set of equations E and the set of all the remaining clauses S. Our purpose is then to apply deduction steps using the clauses of S, and to consider E as an implicit theory. Therefore, deducing a property P from clauses of S means that P is a logical consequence of S and E.

As said in the introduction, the motivation for working modulo a theory E is to eliminate the explicit use of some very costly axioms. For instance, the use of a property such as the

commutativity of an operator f, $(f(x_1,x_2) \simeq f(x_2,x_1))$, is non-deterministic, since it cannot be oriented as a rewrite rule for a paramodulation step.
Some other properties generate infinite derivations. A well-known example is the associativity of an operator g, $(g(g(x_1,x_2),x_3) \simeq g(x_1,g(x_2,x_3)))$. Given an equation $(g(a,b) \simeq c)$, we can derive the new equation $(g(a,g(b,z_1)) \simeq g(c,z_1))$ from a paramodulation step into the subterm $g(x_1,x_2)$ of the associativity axiom, with the substitution $\{x_1 \mapsto a, x_2 \mapsto b\}$ and renaming x_3 with z_1. Using this new equation, we can apply a paramodulation step into the same subterm $g(x_1,x_2)$ of the associativity axiom, generating the equation $(g(a,g(g(b,z_1),z_2) \simeq g(g(c,z_1),z_2)$. In this way, we can derive infinitely many new equations, adding a new variable z_i at each step.

5.1 Plotkin's Work

The first results in the domain of deduction modulo an equational theory have been obtained by Plotkin [29]. He has defined a deduction system based on resolution and paramodulation rules. We detail his result in this paper because the techniques he has used have been the starting point of our works. However, we give in the following a simplified version of his deduction rules; the original ones include the notion of factoring.

The first rule defined by Plotkin is the **E-resolution**: from two clauses $A_1 \vee D_1$ and $\neg A_2 \vee D_2$ we derive the clause $\sigma(D_1 \vee D_2)$; σ is a substitution unifying A_1 and A_2 in the theory E; this means the equality of $\sigma(A_1)$ and $\sigma(A_2)$ is a logical consequence of E.
Plotkin defines an **E-trivialisation** rule, in which unification in E is used instead of the classical unification.

The E-paramodulation rule defined by Plotkin is the most interesting one.

Definition 5.1 (E-Paramodulation) $\dfrac{(t_1 \simeq t_2) \vee D_1 \qquad [\neg]p(\ldots,t,\ldots) \vee D_2}{\sigma([\neg]p(\ldots,s[x/t_2],\ldots) \vee D_1 \vee D_2)}$
where σ is an E-unifier of t and $s[x/t_1]$, and $< x, s >$ is a special pair.

Let us first explain the notions used in this definition. A paramodulation step is applied to a positive or a negative literal ($[\neg]$). $s[x/u]$ represents the term s in which the variable x is replaced by the term u. A *special pair* is a pair variable-term defined as follows:

- $< x, x >$ is a special pair.
- $< x, s >$ is special if s is a term $f(x_1,\ldots,x_{i-1},s',x_{i+1},\ldots,x_n)$, $< x, s' >$ is a special pair, the variables x_j are all distinct, and no x_j is equal to x.

In this rule, we can notice the replacement is not applied to a subterm in the literal, but directly at the top of term t, occuring as an argument of the predicate operator. The position in t where the replacement is really applied is indicated by the variable x in term s. This kind of replacement is in fact a simulation of a replacement using the equation $(s[x/t_1] \simeq s[x/t_2])$ instead of $(t_1 \simeq t_2)$. Such an equation is called an *extension* of the initial equation.

For summarising, the techniques used by Plotkin are:

1. Unification in the theory E,
2. Implicit extensions.

But many steps for applying deductions are non-deterministic:

- No selection of literals in clauses,

- No orientation of the equations when used for replacements,

- No help for building special pairs.

5.2 Completeness Results

Proving the refutational completeness of a deduction system is a very hard task. Let us present the main results obtained for deduction in an equational theory.

First of all, Plotkin proved his deduction system was complete for any equational theory E. This result is very impressive. But due to too general definition of the paramodulation rule, and to all non-determinisms, Plotkin's system is impossible to be used efficiently.

Years later, new deduction systems have been defined for more specific classes of theories. The first studied theories have been the associative-commutative theories, as they appear everywhere in practice. Completeness proofs have been obtained by Rusinowitch and Vigneron [34, 35], Paul [23], and Bachmair and Ganzinger [4].

In 1992, Wertz [45] proved the refutational completeness of his deduction system for all the theories E satisfying the following property:

$$\forall s, t \text{ ground, if } s =_E t[s], \text{ then } \forall s' \text{ ground, } s' =_E t[s'] \tag{1}$$

The technique used by Wertz for proving the completeness is based on the model construction of Bachmair and Ganzinger [3]. Paul also proved the completeness for the same class of theories of a similar deduction system [22], but using a different proof technique based on the transfinite semantic trees method of Hsiang and Rusinowitch [9].
We have generalised the deduction system developed for associative-commutative theories in [35] to an equational theory E, and we proved its refutational completeness for any regular theory [39, 40]:

$$E \text{ is } \textit{regular} \text{ if } \forall (t_1 \simeq t_2) \in E,\ \mathcal{V}ar(t_1) = \mathcal{V}ar(t_2) \tag{2}$$

Our proof technique is also based on the transfinite semantic trees method [9], but differs significantly from Paul's since we define a completely different semantic tree. We also have proved our deduction system was complete for theories satisfying the property (1), as some non-regular theories satisfy (1), while many regular theories do not satisfy (1).

Example 5.1 ((1) $\not\Rightarrow$ (2)) *Let E be $\{f(x,y) \simeq x\}$, where f is the only functional operator, and a is the only constant. E is not regular, but E satisfies the property (1). Indeed, each ground term is E-equal to a.*

Example 5.2 ((2) $\not\Rightarrow$ (1)) *Let E be $\{f(x,0) \simeq x\}$. E is regular. The ground term $f(0,0)$ is E-equal to 0. However, if there exists a constant a, $f(0,a)$ is not E-equal to a. E does not satisfy the property (1).*

5.3 Strategies

In [39], we prove the ordering, simplification and superposition strategies described in Section 4 can be used without loss of completeness for deduction modulo E. In the next paragraphs, we discuss on different available strategies.

Ordering Strategies. The ordering and positive strategies are still compatible provided that the ordering $>$ used for comparing terms is compatible with the theory E, i.e.:

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

Very few orderings satisfying this property have been defined. We have only some for associative-commutative theories [17, 33] and associative theories [32].

Remark 5.1 *There is a relation between the definition of such orderings and the last completeness results: given a theory E, if there exists a total E-compatible ordering, E is regular [1].*

Simplification Strategies. The strategies for eliminating or transforming redundant clauses are still valid. For applying them, we have to use an E-matching algorithm, and some simplification steps may require to be applied with extensions of equations.

Constraint Strategies. The basic and constraint strategies were proved valid for deduction in associative-commutative theories [38, 18] and associative theories [32]. The theoretical advantages of the constraint strategy are even more significant:

- Only one clause is deduced at each step, instead of as many as solutions of the E-unification problem. And some problems may have many solutions. Considering for instance the unification problem $x * x * x * x =^?_{AC} y_1 * y_2 * y_3 * y_4$ where $*$ is associative-commutative (AC), instead of generating one clause per principal solution (i.e. 34,359,607,481 clauses), we generate only one clause. This also permits to work in associative theories, while some unification problems have infinitely many solutions.

- We only need to check the satisfiability of the E-unification problem. For instance, the gain is exponential in the case of associative-commutative theories.

Extension Strategies. In Plotkin's paramodulation rule, we have shown the implicit use of extensions for equations: from an equation $(t_1 \simeq t_2)$, an equation $(s[t_1] \simeq s[t_2])$ is simulated. These extensions have been studied by Jouannaud and Kirchner [12], and by Bachmair and Dershowitz [2]. Let us detail a simple example explaining why extensions have to be considered.

Example 5.3 *Let E be the associativity property of an operator f:*

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

Let S be the set of the following three clauses:

$$\left\{ \begin{array}{ll} (f(a,b) \simeq c) & (1) \\ (f(a, f(b,d)) \simeq e) & (2) \\ \neg(f(c,d) \simeq e) & (3) \end{array} \right.$$

S is incoherent with the theory E, but using the classical paramodulation rule with E-unification, there is no possible deduction step. However, the incoherence can be shown as follows:

$$\begin{array}{ccc} f(\underbrace{f(a,b)}, d) & =_E & \underbrace{f(a, f(b,d))} \\ \downarrow_{(1)} & & \downarrow_{(2)} \\ f(\overbrace{c}, d) & \simeq & \overbrace{e} \end{array}$$

From the associativity axiom and equations (1) *and* (2)*, we can deduce* $(f(c,d) \simeq e)$*, contradicting the clause* (3)*.*

The solution is to consider the extension $(f(f(a,b),z) \simeq f(c,z))$ *of the first equation. A paramodulation step from this extension into* (2) *generates immediately* $(f(c,d) \simeq e)$*.*

Extensions can be considered in an explicit way or in an implicit way. Explicitly, one defines a deduction rule generating the possible extensions of a given equation. This technique has been used by Peterson and Stickel [28], Bachmair and Ganzinger [4], Wertz [45] and Paul [22]. Its main problem is that it needs a strong control of the generated extensions for avoiding useless deduction steps.
Instead, we have used Plotkin's technique, implicit extensions, by coding them in the paramodulation rule [35, 40]. This is realized thanks to the pre-computation of contexts. A context is the skeleton added around each member of the equation to be extended. For instance in the previous example, $f(\cdot, z)$ is the context used for extending $(f(a,b) \simeq c)$. This computation of contexts is combined with algorithms detecting redundant ones.

6 The System daTac

The system daTac[1], standing for *Déduction Automatique dans des Théories Associatives et Commutatives*, has been developed at LORIA (Nancy, France). This software is written in CAML Light ($\simeq$18000 lines), a language of the ML family developed at INRIA [15]. It can be run on Alpha, Sun, HP or IBM PC workstations. Its graphical interface is written in Tcl/Tk [21].

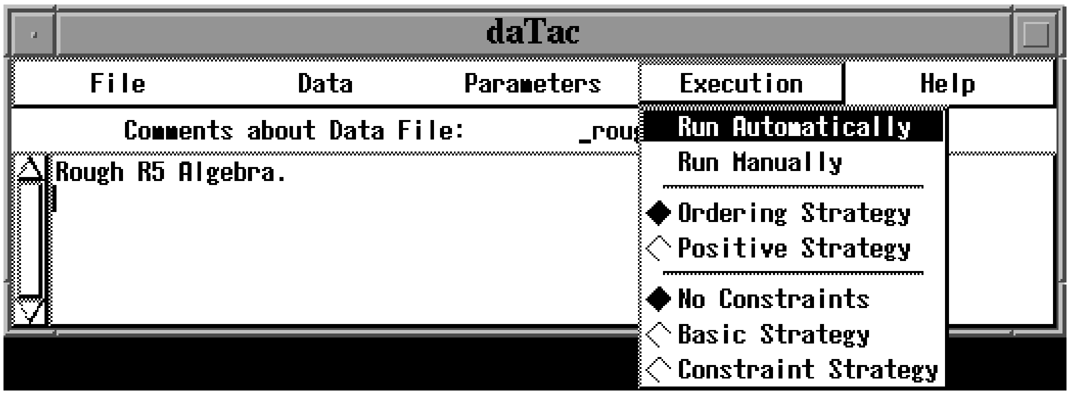

Figure 1: The System daTac.

daTac is defined for automated deduction in theories E composed of commutative and associative-commutative axioms. The implemented deduction techniques are detailed in [35, 38, 39]. Many strategies can be combined:

- For selecting literals: ordering or positive strategy.
- For the equality predicate: paramodulation or superposition strategy.
- For eliminating redundant clauses: subsumption, simplification, ...
- For E-unification problems: basic or constraint strategy.

[1] daTac is available via the PROTHEO Group Home Page (`http://www.loria.fr/equipes/protheo`).

daTac proposes either to prove properties by refutation, or by straightforward deduction from clauses. The refutation technique is a proof existence technique, i.e. we add the negation of a clause to be proved to the set of initial clauses and we search for a contradiction. In this case, if the proof exists the prover would say: "yes, it is a theorem", i.e. the prover acts as a proof existence checker. Of course, the whole system is, as a classical predicate logic prover, semi-decidable. In the straightforward deduction, the prover acts as a deductive system, i.e. the end product is a set of properties with their formal proof. This second technique has been mainly used in the application described in the next section.

At the end of an execution, the user can ask for a lot of extra information. In particular, the prover can provide a proof of a derived property, or of the contradiction found. Many statistics are also available, such as the number of deduction steps, the number of simplification steps, and the number of deletions.

7 An Application: The Study of Rough Algebras

The theory of rough sets provides a methodological framework to study classification problems with imprecise or incomplete information. The theory was originally introduced by Pawlak in [24, 25]. It provides a complementary model to fuzzy [26] or evidence [36] models for dealing with imprecise, noisy, or incomplete information. It is also a departure point for studies in machine learning, knowledge discovery and lately in database mining [46]. It has also been applied to various domains such as medical diagnosis, information retrieval, control algorithms acquisition, and market analysis.

In [41], together with Professor Wasilewska (Stony Brook University, New York), we have presented two new connections for the rough set theory. We linked it with the automated theorem proving and algebraic logic. More precisely, we first used the long established [20] relationship between the rough sets and modal S5 logic, and hence with some topological Boolean algebras and secondly, we used their newly established [6, 42] relationship with *rough algebras*. Then we used the theorem prover daTac to discover new theorems in two modal and two rough classes of algebras and to examine the relationship between these classes.

In our research we have used mainly daTac for straightforward deduction, but we have also used it as a theorem prover for comparing theorems of modal and rough algebras, respectively. The rough algebras under consideration are quasi-Boolean version of some closure algebras introduced in [16]. We call them here modal S4 and S5 algebras. We single out two classes of rough algebras and call them because their similarity with modal S4 and S5 algebras, the rough R4 and R5 algebras. In the definition of all these algebras, the operators $\mathtt{I}$ and $\mathtt{C}$ represent the topological *interior* and *closure* over sets (see Figure 2). We say two sets A and B are *approximatively* (or *roughly*) *equal* if $\mathtt{I(A)} = \mathtt{I(B)}$ and $\mathtt{C(A)} = \mathtt{C(B)}$.
A natural set theoretical interpretation of the properties of the topological Boolean algebras is established by Stone representation theorem [37]. For example, $\mathtt{a} \cdot \mathtt{Ia} = \mathtt{Ia}$ means that any set $\mathtt{A}$ contains its interior $\mathtt{IA}$. The representation theorem provides an intuitive motivation for new properties and is a useful source of counter-examples.

The structure of rough algebras is very complicated and their properties are much less intuitive than the standard modal algebras. The role of the prover in finding and proving new theorems or properties of these algebras is hence essential in developing their consistent theory.

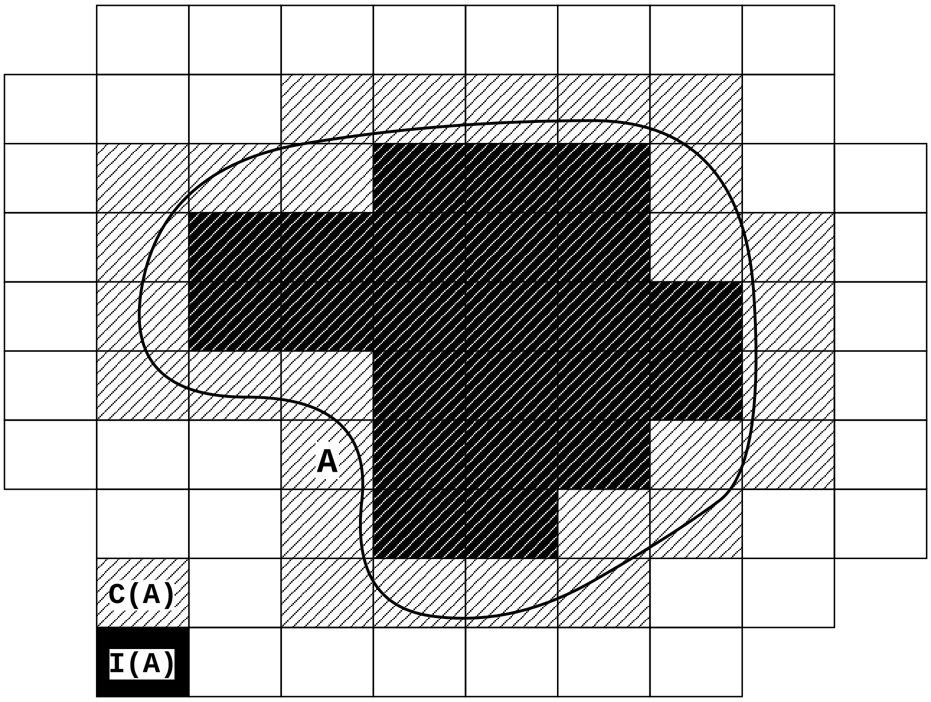

Figure 2: Set Approximation.

7.1 Axiomatisation of Rough and Modal Algebras

The *R4 algebra* $(A, \mathtt{1}, +, \cdot, \sim, \mathtt{I}, \mathtt{C})$ is defined by: $(A, \mathtt{1}, +, \cdot)$ is a distributive lattice with unit element $\mathtt{1}$,

$$\begin{array}{rcl@{\qquad}rcl}
(\mathtt{x1} \cdot \mathtt{x2} & \simeq & \mathtt{x2} \cdot \mathtt{x1}) & ((\mathtt{x1} \cdot \mathtt{x2}) \cdot \mathtt{x3} & \simeq & \mathtt{x1} \cdot (\mathtt{x2} \cdot \mathtt{x3})) \\
(\mathtt{x1} + \mathtt{x2} & \simeq & \mathtt{x2} + \mathtt{x1}) & ((\mathtt{x1} + \mathtt{x2}) + \mathtt{x3} & \simeq & \mathtt{x1} + (\mathtt{x2} + \mathtt{x3})) \\
((\mathtt{x1} \cdot \mathtt{x2}) + \mathtt{x2} & \simeq & \mathtt{x2}) & (\mathtt{x1} \cdot (\mathtt{x2} + \mathtt{x3}) & \simeq & (\mathtt{x1} \cdot \mathtt{x2}) + (\mathtt{x1} \cdot \mathtt{x3})) \\
(\mathtt{x1} \cdot (\mathtt{x1} + \mathtt{x2}) & \simeq & \mathtt{x1}) & ((\mathtt{x1} + \mathtt{x2}) \cdot (\mathtt{x1} + \mathtt{x3}) & \simeq & \mathtt{x1} + (\mathtt{x2} \cdot \mathtt{x3})) \\
(\mathtt{x1} + \mathtt{1} & \simeq & \mathtt{1}) & (\mathtt{x1} \cdot \mathtt{1} & \simeq & \mathtt{x1})
\end{array}$$

operator $\sim$ is a quasi-complement,

$$(\sim\sim\mathtt{x1} \ \simeq \ \mathtt{x1}) \qquad (\sim(\mathtt{x1} + \mathtt{x2}) \ \simeq \ \mathtt{x1} \cdot \sim\mathtt{x2})$$

and $\mathtt{I}$ and $\mathtt{C}$ are defined by:

$$\begin{array}{rcl@{\qquad}rcl}
(\mathtt{I}(\mathtt{x1} \cdot \mathtt{x2}) & \simeq & \mathtt{I}(\mathtt{x1}) \cdot \mathtt{I}(\mathtt{x2})) & (\mathtt{I}(\mathtt{x1}) \cdot \mathtt{x1} & \simeq & \mathtt{I}(\mathtt{x1})) \\
(\mathtt{I}(\mathtt{I}(\mathtt{x1})) & \simeq & \mathtt{x1}) & (\mathtt{I}(\mathtt{1}) & \simeq & \mathtt{1}) \\
(\mathtt{C}(\mathtt{x1}) & \simeq & \sim\mathtt{I}(\sim\mathtt{x1}))
\end{array}$$

The *R5 algebra* is an R4 algebra with the additional property:

$$(\mathtt{C}(\mathtt{I}(\mathtt{x1})) \ \simeq \ \mathtt{I}(\mathtt{x1}))$$

This property is called the *clopen property*, as it means that any closed set is open, and any open set is closed.

The *modal algebras S4 and S5* are defined like the rough R4 and R5 algebras respectively, except the complement is entirely defined; the following properties are added:

$$(\sim\mathtt{x1} + \mathtt{x1} \ \simeq \ \mathtt{1}) \qquad (\sim\mathtt{x1} \cdot \mathtt{x1} \ \simeq \ \sim\mathtt{1})$$

7.2 Quasi-Boolean Algebras

Our first work on those rough algebras has been to study more precisely the quasi-Boolean algebras. In such algebras, the operators $\cdot$ and $+$ are associative and commutative. We have specified the definition of quasi-Boolean algebras in daTac, i.e. the 8 equations given in Section 7.1 (the associativity and commutativity properties being declared built-in, they are not explicitely mentioned as equations).

We have run daTac, without any property to prove, but with the intention to deduce all the properties that are consequences of the 8 initial axioms. After 20 seconds, the prover has stopped its execution. It had kept 11 equations. No more useful deduction step was possible. These 11 equations represent a *canonical system* of all the properties of quasi-Booleans algebras:

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
((\mathtt{x1}\cdot\mathtt{x2})+\mathtt{x2} & \simeq & \mathtt{x2}) & (\mathtt{x1}\cdot(\mathtt{x2}+\mathtt{x3}) & \simeq & (\mathtt{x1}\cdot\mathtt{x2})+(\mathtt{x1}\cdot\mathtt{x3})) \\
(\mathtt{x1}+\mathtt{1} & \simeq & \mathtt{1}) & (\mathtt{x1}\cdot\mathtt{1} & \simeq & \mathtt{x1}) \\
(\sim\sim\mathtt{x1} & \simeq & \mathtt{x1}) & (\sim(\mathtt{x1}+\mathtt{x2}) & \simeq & \sim\mathtt{x1}\cdot\sim\mathtt{x2}) \\
(\mathtt{x1}+\mathtt{x} & \simeq & \mathtt{x1}) & (\mathtt{x1}\cdot\mathtt{x1} & \simeq & \mathtt{x1}) \\
(\mathtt{x1}\cdot\sim\mathtt{1} & \simeq & \sim\mathtt{1}) & (\sim\mathtt{1}+\mathtt{x1} & \simeq & \mathtt{x1}) \\
(\sim(\mathtt{x1}\cdot\mathtt{x2}) & \simeq & \sim\mathtt{x1}+\sim\mathtt{x2}) & & &
\end{array}
$$

This means these equations can be oriented from left to right as rewrite rules, and any property of a quasi-Boolean algebra can be immediately reduced into a tautology by these 11 rules.

The strategy used for this execution is a combination of the ordering strategy, the superposition strategy and simplification strategies. The basic strategy can also be combined, bringing an acceleration of 20%.

7.3 Study of Rough and Modal Algebras

For studying the rough algebras R4 and R5 and the modal algebras S4 and S5, we have run daTac from their specification, and we have stopped the execution after 10000 clauses deduced. As for the study of quasi-Boolean algebras, we have used a strategy combining ordering, superposition and simplification strategies. Each execution lasted around 20 hours (with a Digital Alpha 500), and used 35 Mb of memory. The results are detailed in Figure 3. This table presents the

	R4	*R5*	*S4*	*S5*
Initial Clauses	16	17	18	19
Generated Clauses	9987	10025	9994	9984
Final Clauses	684	472	863	676
Deduction Steps	4770	4119	5752	5228
Simplification Steps	28739	31958	32532	33651
Deletion Steps	14461	16113	14946	15008

Figure 3: Study of Rough and Modal Algebras.

number of initial clauses for each algebra, then the number of new clauses generated, and the number of remaining clauses when stopping the execution. Finally, the number of deduction steps, simplification steps, and deleted clauses are displayed. This table shows the essential role of simplification strategies.

Properties of these rough and modal algebras are very difficult to find by hand. This is even very difficult, given a property, to decide if it belongs to one of these algebras. In the thousands of properties generated by daTac, we have discovered very interesting ones. We have also been able to prove some interesting patterns, such as:

$$\begin{aligned}
&\mathtt{I(C(x1)} + \ldots + \mathtt{C(xn)} + \mathtt{I(y1)} + \ldots + \mathtt{I(ym))} \\
&\qquad\simeq\ \mathtt{C(x1)} + \ldots + \mathtt{C(xn)} + \mathtt{I(y1)} + \ldots + \mathtt{I(ym)} \\
&\mathtt{C(C(x1)} \cdot \ldots \cdot \mathtt{C(xn)} \cdot \mathtt{I(y1)} \cdot \ldots \cdot \mathtt{I(ym))} \\
&\qquad\simeq\ \mathtt{C(x1)} \cdot \ldots \cdot \mathtt{C(xn)} \cdot \mathtt{I(y1)} \cdot \ldots \cdot \mathtt{I(ym)}
\end{aligned}$$

A more detailed study of the properties deduced by daTac for those algebras can be found in [43].

For getting an idea of how huge is the research space in those examples, we have asked daTac to derive properties with the strategy width-first. From the 17 initial clauses of R5, daTac has generated 95 properties whose proof is of length 1, but only 30 have been kept from the 112 clauses (17+95). Starting from those 30 properties, 1178 clauses have been generated, 149 have been kept. For properties for which the proof length is 3, we have started once again deductions from the 149 clauses found, but daTac has not been able to generate all the possible properties, by lack of memory.
This confirms efficient strategies have to be invoked for guiding the prover.

7.4 Comparison of Rough and Modal Algebras

We have also defined a procedure for comparing algebras using daTac [41]. This procedure has been applied to the comparison of rough and modal algebras. Its principle is to compare the sets of clauses obtained for each algebra, using techniques such as identity checking, subsumption, but also by studying the deductions steps applied for generating a particular property.

A *vs* B	$A \cap B$	*trivial* A	*future* B	*pure* A
$R4$ *vs* $R5$	301	367	16	–
$R5$ *vs* $R4$	301	0	63	108
$R4$ *vs* $S4$	130	0	554	–
$S4$ *vs* $R4$	130	51	0	682
$R5$ *vs* $S5$	126	0	346	–
$S5$ *vs* $R5$	126	46	0	504
$S4$ *vs* $S5$	469	383	11	–
$S5$ *vs* $S4$	469	11	151	45

Figure 4: Comparison of Rough and Modal Algebras.

Figure 4 describes the results obtained by our technique when comparing R4 and R5, R4 and S4, R5 and S5, and S4 and S5. Comparing A *vs* B, the column $A \cap B$ represents the number of properties common to A and B. The column *trivial* A shows the number of clauses of A subsumed by B or simplified into a trivial property by B. The column *future* B shows the number of clauses of A that are proved using only clauses of $A \cap B$; this means those properties would have been also derived by B if we had not stopped arbitrarily the execution. The last column presents the number of pure A properties, i.e properties of A that are not properties of B; there is no value in this column if $A \subseteq B$.

See [41] for more details concerning the procedure for comparing algebras and the results obtained for those S4, S5, R4 and R5 algebras.

7.5 Generalised Rough Sets

The notion of rough equality is usually restricted to the rough sets only. In [44], we have extended this notion to general topological spaces and generalised rough sets (topological spaces with the clopen property, $\mathtt{C(I(x))} \simeq \mathtt{I(x)}$), and consequently to topological and rough Boolean algebras, respectively.

Given a topological (rough) Boolean algebra $\mathcal{A}$, some natural questions arise: is the topological (rough) equality a congruence with respect to some (or all) operations of $\mathcal{A}$? Can we define an algebra $\mathcal{B}$ similar to $\mathcal{A}$, such that topological equality is a congruence with respect to all its operations?

We have proved that if $\mathcal{A}$ is a topological Boolean algebra, the topological equality is a congruence with respect to the complement operation only. If $\mathcal{A}$ is a rough Boolean algebra, the rough equality is a congruence with respect to complement, $\mathtt{I}$ and $\mathtt{C}$ operations. We have also described the construction from $\mathcal{A}$ of an algebra $\mathcal{B}$, such that rough equality is a congruence with all operations of $\mathcal{B}$.

8 Conclusion

We have presented in this paper a non-exhaustive state of the art in the domain of the automated deduction in first-order logic with equality. We have described the main strategies that have been defined for improving resolution- and paramodulation-based deduction systems. We have also presented how to apply deductions modulo some properties. The results obtained are encouraging, since applying deductions with a theory built-in is something very natural.

An application of these deduction techniques has been developed. It shows how useful an automatic tool can be for logicians and mathematicians. With a prover like daTac, almost impossible tasks for a human being become possible, but still very difficult. For instance, when studying an algebra, thousands of properties have been derived. A lot of them were useless, and some were of no interest from a logician's point of view. And when we finally stopped the prover, we got several hundred remaining properties; we had try to to understand them and to classify them.
This would be very interesting to design heuristics or strategies for getting faster useful properties, looking for properties of a special kind, or identifying patterns of properties. Such heuristics could be automatic or interact with the user.

We have given one example of application of automated deduction, but there are many. For instance, in software engineering, one of the most important research domain is the verification of programs. Several automatic tools are already used in this purpose. The techniques developed in this article can also be used for verifying programs modelled with algebraic specifications.

Acknowledgements: I would like to thank Anita Wasilewska for having shown me a very interesting application of the techniques I developed during my PhD Thesis, and for all the work we did and still do together. I also would like to thank the referee for the comments on this application to the study of rough algebras, and the ideas concerning further work.

References

[1] Baader, F.: "Combination of Compatible Reduction Orderings that are Total on Ground Terms". *Proceedings 12th IEEE Symposium on Logic in Computer Science*, Warsaw

(Poland), 1997, 2–13.

[2] Bachmair, L. and Dershowitz, N.: "Completion for rewriting modulo a congruence". *Theoretical Computer Science*, 67(2-3), 1989, 173–202.

[3] Bachmair, L. and Ganzinger, H.: "On Restrictions of Ordered Paramodulation with Simplification". M. E. Stickel (ed.), *Proceedings 10th International Conference on Automated Deduction*, volume 449 of Lecture Notes in Computer Science, Springer-Verlag, Kaiserslautern (Germany), 1990, 427–441.

[4] Bachmair, L. and Ganzinger, H.: "Associative-Commutative Superposition". *Technical report MPI-I-93-267*, Max Planck Institut für Informatik, Saarbrücken, 1993.

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

[6] Banerjee, M. and Chakraborty, M. K.: "Rough Algebra". *Bulletin of Polish Academia of Science (Mathematics)*, 41(4), 1993, 299–297.

[7] Brand, D.: "Proving Theorems with the Modification Method". *SIAM Journal of Computing*, 4, 1975, 412–430.

[8] Dershowitz, N. and Jouannaud, J.-P.: "Rewrite Systems". J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, Elsevier Science Publishers B. V. (North-Holland), 1990.

[9] Hsiang, J. and Rusinowitch, M.: "Proving Refutational Completeness of Theorem Proving Strategies: The Transfinite Semantic Tree Method". *Journal of the ACM*, 38(3), 1991, 559–587.

[10] Hullot, J.-M.: "Compilation de Formes Canoniques dans les Théories Équationelles". *Thèse de Doctorat de Troisième Cycle*, Université de Paris Sud, Orsay (France), 1980.

[11] Jouannaud, J.-P. and Kirchner, C.: "Solving Equations in Abstract Algebras: a Rule-based Survey of Unification". J.-L. Lassez and G. Plotkin (ed.), *Computational Logic. Essays in honor of Alan Robinson*, chapter 8, The MIT press, Cambridge (MA, USA), 1991, 257–321.

[12] Jouannaud, J.-P. and Kirchner, H.: "Completion of a Set of Rules Modulo a Set of Equations". *SIAM Journal of Computing*, 15(4), 1986, 1155–1194. Preliminary version in Proceedings 11th ACM Symposium on Principles of Programming Languages, Salt Lake City (USA), 1984.

[13] Kirchner, C., Kirchner, H. and Rusinowitch, M.: "Deduction with Symbolic Constraints". *Revue d'Intelligence Artificielle*, 4(3), 1990, 9–52. Special issue on Automatic Deduction.

[14] Lankford, D. S. and Ballantyne, A.: "Decision Procedures for Simple Equational Theories with Associative Commutative Axioms: Complete Sets of Associative Commutative Reductions". *Technical report*, University of Texas at Austin, Department of Mathematics and Computer Science, 1977.

[15] Leroy, X. and Mauny, M.: "The Caml Light System, Release 0.5. Documentation and Users' Manual". *Technical report*, INRIA Rocquencourt, 1993.

[16] McKinsey, J. C. C. and Tarski, A.: "The Algebra of Topology". *Annales of Mathematics*, 45, 1944, 141–191.

[17] Narendran, P. and Rusinowitch, M.: "Any Ground Associative-Commutative Theory has a Finite Canonical System". R. V. Book (ed.), *Proceedings 4th Conference on Rewriting Techniques and Applications*, Springer-Verlag, Como (Italy), 1991.

[18] Nieuwenhuis, R. and Rubio, A.: "AC-Superposition with Constraints: No AC-Unifiers Needed". A. Bundy (ed.), *Proceedings 12th International Conference on Automated Deduction*, volume 814 of Lecture Notes in Artificial Intelligence, Springer-Verlag, Nancy (France), 1994, 545–559.

[19] Nieuwenhuis, R. and Rubio, A.: "Theorem Proving with Ordering and Equality Constrained Clauses". *Journal of Symbolic Computation*, 19(4), 1995, 321–352.

[20] Orlowska, E. and Pawlak, Z.: "Expressive Power of Knowledge Representation Systems". *International Journal of Man-Machine Studies*, 20, 1984, 485–500.

[21] Ousterhout, J. K.: "Tcl and the Tk toolkit, volume ISBN 0.201.63337.X". Addison-Wesley Publishing Company, 1994.

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

[23] Paul, E.: "A General Refutational Completeness Result for an Inference Procedure Based on Associative-Commutative Unification". *Journal of Symbolic Computation*, 14(6), 1992, 577–618.

[24] Pawlak, Z.: "Rough Sets". *International Journal of Computation and Information Sciences*, 11, 1982, 341–356.

[25] Pawlak, Z.: *Rough Sets*, Theory and Decision Library, Kluwer Academic Publishers, 1991.

[26] Pawlak, Z.: "Rough Sets – A New Approach to Vagueness". *Fuzzy Logic for Management of Uncertainty*, Wiley and Sons, New York, 1992, 105–118.

[27] Peterson, G.: "A Technique for Establishing Completeness Results in Theorem Proving with Equality". *SIAM Journal of Computing*, 12(1), 1983, 82–100.

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

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

[30] Robinson, G. A. and Wos, L. T.: "Paramodulation and first-order theorem proving". B. Meltzer and D. Mitchie (ed.), *Machine Intelligence*, 4, Edinburgh University Press, 1969, 135–150.

[31] Robinson, J. A.: "A Machine-oriented Logic Based on the Resolution Principle". *Journal of the ACM*, 12, 1965, 23–41.

[32] Rubio, A.: "Automated Deduction with Constrained Clauses". *PhD thesis*, Universitat Politècnica de Catalunya, Barcelona, Spain, 1994.

[33] Rubio, A. and Nieuwenhuis, R.: "A Precedence-based Total AC-Compatible Ordering". C. Kirchner (ed.), *Proceedings 5th Conference on Rewriting Techniques and Applications*, volume 690 of Lecture Notes in Computer Science, Springer-Verlag, Montreal (Canada), 1993, 374–388.

[34] Rusinowitch, M. and Vigneron, L.: "Automated Deduction with Associative Commutative Operators". Ph. Jorrand and J. Kelemen (ed.), *Proceedings of Fundamental of Artificial Intelligence Research*, volume 535 of Lecture Notes in Computer Science, Springer-Verlag, 1991, 185–199.

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

[36] Skowron, A. and Grzymala-Busse, J.: *From Rough Set Theory to Evidence Theory*, Advances in Dempster-Shafer Theory of Evidence, Wiley and Sons, New York, 1993.

[37] Stone, M. H.: "Boolean Algebras and their Relation to Topology". *Proceedings of National Academy of Sciences*, 20, 1934, 197–202.

[38] Vigneron, L.: "Associative-Commutative Deduction with Constraints". A. Bundy (ed.), *Proceedings 12th International Conference on Automated Deduction*, volume 814 of Lecture Notes in Artificial Intelligence, Springer-Verlag, Nancy (France), 1994, 530–544.

[39] Vigneron, L.: "Automated Deduction with Symbolic Constraints in Equational Theories". *PhD thesis*, Université Henri Poincaré – Nancy 1, 1994. (In French).

[40] Vigneron, L.: "Positive Deduction modulo Regular Theories". H.K. Büning (ed.), *Annual Conference of the European Association for Computer Science Logic*, volume 1092 of Lecture Notes in Computer Science, Springer-Verlag, Paderborn (Germany), 1995, 468–485. Selected paper.

[41] Vigneron, L. and Wasilewska, A.: "Modal and Rough Algebras". P. Borne, G. Dauphin-Tanguy, C. Sueur, and S. El Khattabi (ed.), *Proceedings of the CESA'96 International Multiconference (Computational Engineering in Systems Applications)*, Symposium on Modelling, Analysis and Simulation, IMACS/IEEE, Lille (France), 1996, 1107–1112.

[42] Wasilewska, A.: "Topological Rough Algebras". *Rough Sets and Database Mining*, Kluwer Academic Publishers. To appear.

[43] Wasilewska, A. and Vigneron, L.: "Rough Equality Algebras". *Proceedings of the Joint Conference on Information Sciences, Workshop on Rough Set Theory* (invited session), Wrightsville Beach (North Carolina), 1995.

[44] Wasilewska, A. and Vigneron, L.: "On Generalized Rough Sets". *Proceedings of the third Joint Conference on Information Sciences, fifth international Workshop on Rough Sets and Soft Computing (RSSC'97)*, Research Triangle Park (North Carolina), 1997.

[45] Wertz, U.: "First-order theorem proving modulo equations". *Technical Report MPI-I-92-216*, Max Planck Institut für Informatik, 1992.

[46] Ziarko, W. P.: "Rough Sets, Fuzzy Sets and Knowledge Discovery". *Workshops in Computing Series*, Springer-Verlag, 1994.