CENTRE de RECHERCHE en INFORMATIQUE de NANCY

Campus Scientifique
Bâtiment LORIA
Boîte Postale 239
54506 Vandœuvre-lès-Nancy Cedex

Téléphone : (+ 33) 83.59.20.00
Télécopie : (+ 33) 83.41.30.79

Associative-Commutative Deduction with Constraints

Laurent Vigneron

CRIN 93-R-196

NOVEMBRE 1993

Presented at CADE'12, Nancy, June 1994 [Vig94a]

Associative-Commutative Deduction with Constraints*

Laurent Vigneron

CRIN-CNRS & INRIA-Lorraine

BP 239, 54506 Vandœuvre-lès-Nancy Cedex, France

E-mail : Laurent.Vigneron@loria.fr

Abstract

Associative-commutative equational reasoning is known to be highly complex for theorem proving. Hence, it is very important to focus deduction by adding constraints, such as unification and ordering, and to define efficient strategies, such as the basic requirements à la Hullot. Constraints are formulas used for pruning the set of ground instances of clauses deduced by a theorem prover. We propose here an extension of AC-paramodulation and AC-superposition with these constraint mechanisms ; we do not need to compute AC-unifiers anymore. The method is proved to be refutationally complete, even with simplification. The power of this approach is exemplified by a very short proof of the equational version of SAM's Lemma using `DATAC`, our implementation of the strategy.

Keywords : Automated Deduction, Associative and Commutative Theories, Resolution, Paramodulation, Superposition, Basic Strategy, Symbolic Constraints.

Contents

*This work is partially supported by the GDR Programmation and the Esprit Basic Research working group 6028, CCL.

1 Introduction

Automated deduction with equality and associative-commutative (AC) operators, i.e. binary operators f satisfying the axioms $f(f(x,y),z) \simeq f(x,f(y,z))$ and $f(x,y) \simeq f(y,x)$, has been a long standing open problem. Our solution for dealing with AC axioms is to work in the AC-congruence classes and to use associative and commutative identity equality, matching and unification. This idea of replacing axioms by ad-hoc mechanisms such as unification algorithms and inference rules was first proposed by Plotkin in [Plo72], and has been extended to term rewriting systems including associativity and commutativity in [PS81, JK86].

The paramodulation rule has been extended to AC theories in [RV91, Pau92, Wer92] and in [RV93], where we have proposed an inference system based on the ordered paramodulation strategy. In this paper, we add important restrictions in the line of [NR92a, BGLS92] for obtaining a basic strategy for AC-paramodulation and AC-superposition. The principle of the basic strategy, extension of the basic narrowing of [Hul80], is to forbid paramodulation in terms introduced by a substitution in a previous deduction. For instance, considering the paramodulation step (with the unifier $\{x \mapsto a,\ y \mapsto c\}$)

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

$f(x,c)$ is replaced by b, assuming that $f(a,c) \not\preceq b$ and $P(f(a,c)) \not\preceq Q(a,b)$. The subterm a of the deduced clause is blocked (framed) to show that no more paramodulation step is allowed in it. A refinement of this strategy is to block in addition the replacing term in the deduced clause (i.e. b in $P(b)$). This refinement is not valid for the superposition strategy [Rus91].

In the following, we refine and extend this strategy by associating to each clause a set of constraints as in [KKR90]. Then, when a new clause is generated, it inherits the constraints of its ancestors and also the AC-unification constraints (not computed) and ordering constraints produced by the last inference step. When we apply this technique to the previous example, we get :

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

where T_3 consists in the AC-unification constraints $f(x,c) =_{AC} f(a,y)$ and $z =_{AC} b$ (the last one corresponds to the basic strategy), and the ordering constraints $f(a,y) \succ b$ and $P(f(x,c)) \succ Q(x,b)$ for simulating an ordered strategy.

In a first version of this work[1], we had to compute AC-unifiers in a very restrictive case. Later Nieuwenhuis and Rubio [NR94] have proposed a very close result where no AC-unifier was needed (but with other drawbacks). In this paper, we have been able to get rid of the last few AC-unifiers needed by a very simple trick, inspired by the proof technique of Nieuwenhuis and Rubio. After discussion with them, this trick is compatible with their method.

Related works on the basic strategy and constraints are detailed in Section 2 ; we describe there our motivations for applying such techniques to deduction in AC theories. Our notations are described in the third section. Section 4 presents inference rules and the lifting lemma. Section 5 introduces several important simplification rules that are compatible with our inference rules. The method for proving the refutational completeness is explained in Section 6 and the detailed proof is given in the Appendix A. The last Section 7 details the proof of SAM's Lemma, which was produced by our implementation of the method described in this paper.

2 Related Work and Motivations

The notion of basic strategy was first introduced by Slagle [Sla74] with the definition of *blocked inferences*, i.e. inferences using substitutions which do not contain a reducible term, in the sense of term rewriting. This concept was extended to theories with permutative axioms (including AC axioms) by Ballantyne and Lankford [BL79]. In another setting but with close ideas, Kapur, Musser and Narendran [KMN85]

[1]Presented at the 2nd Workshop CCL, La Escala, Spain, September 1993.

have defined the *prime superposition*, which takes into account that many superpositions are useless in the Knuth-Bendix completion procedure. But, the first strategy including the main ideas of the basic strategy is the *modification method* of Brand [Bra75].

In his thesis, Hullot [Hul80] has introduced a related concept called *basic narrowing*, which consists in forbidding narrowing steps in subterms introduced by previous inference steps. It was extended to *basic superposition* and *basic paramodulation* by Nieuwenhuis and Rubio [NR92a], and Bachmair, Ganzinger, Lynch and Snyder [BGLS92]. Both these works apply the model construction technique of [BG90] to derive the refutational completeness of the strategy. In [NR92a], subterms where superpositions are forbidden never appear explicitly, since they are handled by unification constraints, while in [BGLS92], unifiers are computed and the blocked subterms are replaced by variables and appear in a substitution associated to a clause. To compare the basic superposition and paramodulation strategies, the first one permits to restrict to left-hand sides of equations, while the second one permits to block the right-hand sides when they replace subterms of a clause.

In this paper, we extend the basic paramodulation and superposition to AC theories. We use explicit constraints as in [NR92b] to get an homogeneous framework with ordering and unification constraints at the same level.

Such ordering constraints were used by Peterson [Pet90] and Martin and Nipkow [MN90] to orient equations like the commutativity axiom : $x \cdot y \rightarrow y \cdot x$ if $x > y$. But, constraints are also fundamental to trace the chosen orientations. Indeed, when an equation $(s \simeq t)$ is oriented as $s \rightarrow t$, it has to be coherent with further inference steps. This coherence is imposed by keeping the constraint $s > t$ which is added to the deduced clause.
A second kind of constraints we consider are AC-unification constraints. They permit to limit the number of generated clauses. Consider for instance the paramodulation from the equation $(x * x * x * x \simeq x)$ into the clause $P(y_1 * y_2 * y_3 * y_4)$, where $*$ is AC ; a clause will be generated for each most general AC-unifier, but there are 34 359 607 481 of them. Using an AC-unification constraint, only one clause is deduced : $P(x)\, [\![\, x * x * x * x =_{AC} y_1 * y_2 * y_3 * y_4 \,]\!]$. The advantage of such constraints was already pointed out in [KK89, KKR90] and in [Bür91].

The main result of this paper is the definition of a refutationally complete set of inference rules, compatible with simplification, for deduction with constraints in AC theories. These inference rules are non-trivial refinements of rules defined in [RV93], by including constraints of

- AC-unification, to express the basic strategy and limit the number of deduced clauses,
- Ordering, to express the orientation of equations and the ordered strategy, which consists in always applying inference steps between maximal literals of clauses.

The additional refinements defined in [RV93] remain complete, such as the maximal positions requirement, which consists in forbidding paramodulations at non maximal positions of AC-operators. Moreover, we never use the AC axioms explicitly ; they are handled though AC-unification constraints and the inference rules application.

Nieuwenhuis and Rubio [NR92b] have defined a complete set of inference rules based on superposition, with ordering and unification constraints. Their work deals with the empty theory. They have recently extended this result to AC theories [NR94]. However, we have important refinements on the rules named contextual paramodulation and extended paramodulation (Definitions 5 and 7) they do not have.
Both systems had the same problem : to keep the completeness property, the basic strategy has to be weakened by keeping the possibility of applying indirectly a paramodulation (or superposition) in the constraint part (Example 6). First, we wanted to avoid such inference steps, and we defined an inference system which had to compute some AC-unifiers. But, a nicer solution is to never compute AC-unifier and also to permit some paramodulations in constraints as proposed by Nieuwenhuis and Rubio in [NR94]. So, the inference system presented in this paper is a light transformation of our first system : no nore AC-unifiers are needed, and we show that paramodulations in constraints have to be allowed only in very restrictive cases (Definition 5) which can be tested with a very simple control. For these special cases, we used a trick inspired by the proof technique of Nieuwenhuis and Rubio. We conjecture that we have obtained an optimal version of basicness.
Last, our method is implemented and we report experiments on non-trivial problems.

3 Notations and Definitions

This section introduces the basic notions relevant to our work, based on the standard notations and definitions for term rewriting and unification given in [DJ90, JK91].

We assume that the operators from a given subset $\mathcal{F}_{AC}$ of $\mathcal{F}$ are associative and commutative, which means that for $f \in \mathcal{F}_{AC}$ the axioms $f(f(x,y),z) \simeq f(x,f(y,z))$ and $f(x,y) \simeq f(y,x)$ are implicit in the theory to be considered. The congruence on $\mathcal{T}(\mathcal{F},\mathcal{X})$, generated by these AC axioms, is called AC-equality and written $=_{AC}$. We have to mention that, in this paper, we do not use the flatten representation for terms. The set of *AC-matchers* from a term s into a term t is denoted by $AC\_match\{s,t\}$.

The relation $\sqsubset_{AC}$ represents the subterm property modulo AC, i.e. $u \sqsubset_{AC} v$ if there is a term v', AC-equal to v, such that u is a strict subterm of v'. $u \sqsubseteq_{AC} v$ if $u \sqsubset_{AC} v$ or $u =_{AC} v$. For instance, $a{*}b \sqsubset_{AC} b{*}(a{*}c)$, where $*$ is AC. This relation is extended to sets of terms : $\{s_1,\ldots,s_m\} \sqsubseteq_{AC} \{t_1,\ldots,t_n\}$ if $\forall i,\ \exists j,\ s_i \sqsubseteq_{AC} t_j$.

We denote by $\cap_{AC}$ (resp. $\cup_{AC}$ and $\subseteq_{AC}$) the intersection (resp. union and inclusion) of sets, using the AC-equality for comparing objects, i.e. terms or atoms. For example, $\{a{*}b, b, c\} \cap_{AC} \{b{*}a, c, d\} = \{a{*}b, c\}$, $\{a*b\} \cup_{AC} \{b*a, c\} = \{a*b, c\}$ and $\{a*b, c\} \subseteq_{AC} \{b*a, b, c\}$, where $*$ is AC. The extension of $\subseteq_{AC}$ to multisets is denoted by $\Subset_{AC}$. For instance, $\{a{*}b, c, c\} \Subset_{AC} \{b{*}a, c, b, c\}$, but $\{a{*}b, c, c\} \not\Subset_{AC} \{b{*}a, b, c\}$.

$\mathcal{MP}os(t)$ denotes the set of all non variable positions p in t such that either $\mathcal{H}ead(t_{|_p}) \notin \mathcal{F}_{AC}$, or, if $p \simeq p' \cdot i$, $\mathcal{H}ead(t_{|_p})$ and $\mathcal{H}ead(t_{|_{p'}})$ are not the same AC-operator.
Given a term t and an AC-operator f, $\mathcal{H}terms(t,f)$ is the multiset of the subterms below f at the root of t ; For instance, $\mathcal{H}terms(a*((a*x)*g(b)),*)$ is $\{a,a,x,g(b)\}$.

Our ordered paramodulation strategy is based on a *complete simplification ordering* $\succ$ on terms and atoms, total on ground AC-congruence classes (see definition in [RV93]) and AC-compatible [NR91, RN93]. By overloading of notation, given a literal L and a clause D, $L \succ D$ (resp. $L \prec D$) means that the atom corresponding to L is greater than each (resp. less than one) atom of D.

In the following, we use *constrained clauses*, denoted by $C\,[\![\,T\,]\!]$ as in [NR92b], where C is a clause in the first-order logic and T is a conjunction of constraints, atomic formulas of type $s =_{AC} t$ or $s \succ t$, for terms or atoms s and t. The solutions of a set of constraints T are the ground substitutions defined by :

$$
\begin{array}{ll}
- & \mathcal{S}ol(c \wedge c') \ = \ \mathcal{S}ol(c) \ \cap \ \mathcal{S}ol(c') \\
- & \mathcal{S}ol(s =_{AC} t) \ = \ \{\ \sigma \mid s\sigma \text{ and } t\sigma \text{ are AC-equal } \} \\
- & \mathcal{S}ol(s \succ t) \ = \ \{\ \sigma \mid s\sigma \succ t\sigma\ \}
\end{array}
$$

Some other constraints are used to test emptiness of the intersection of sets of terms, and to test inclusion of multisets of terms. $C\,[\![\,\sigma\,]\!]$ will denote a shorthand of $C\,[\![\,\bigwedge_{x \in \mathcal{D}om(\sigma)} x =_{AC} x\sigma\,]\!]$. A set of constraints T is *satisfiable* if it admits at least one solution.

4 Inference Rules

Before defining our inference rules, let us recall the main ideas of our theorem proving strategies. First, the ordered strategy forces the application of inference rules between maximal literals of clauses. Second, the AC-unification constraints, which are never evaluated, simulate the basic strategy.
When a clause is deduced by an inference step, we do not specify that its set of constraints has to be satisfiable. Indeed, there are two strategies : to check the satisfiability of the constraints either at each step, or only when the empty clause is generated. The first solution is costly, but it avoids further useless deductions. So, the choice of the strategy is left to the user.

4.1 Paramodulation Strategy

We define in the following 6 inference rules, presented as fractions with the initial clauses on top and the deduced clause below. We illustrate their application on an example, where $*$ is an AC-operator, and $P >_p Q >_p \simeq >_p * >_p g >_p a >_p b >_p c >_p d >_p e$ is the precedence on operators. Ordering constraints will be simplified as much as possible thanks to the previous precedence, and AC-unification constraints will be simplified by decomposition, as in [JK91].

Definition 1 (Constrained AC-Factoring)

$$\frac{L_1 \vee \ldots \vee L_n \vee D \;\; [\![T_1]\!]}{L_1 \vee D \;\; [\![T_1 \wedge T_2]\!]} \quad \textit{where } T_2 \textit{ is } \; L_1 =_{AC} \ldots =_{AC} L_n \;\wedge\; L_1 \succ D$$

Example 1 If we apply this AC-factoring rule to the clause $P(x_1 * y) \vee P(b * x) \vee Q(x,y) \; [\![x_1 =_{AC} a]\!]$, we obtain $P(x_1 * y) \vee Q(x,y) \; [\![T]\!]$, where T is simplified to $x_1 =_{AC} a \;\wedge\; x_1 * y =_{AC} b * x$, since the ordering constraint $P(x_1 * y) \succ Q(x,y)$ is trivially satisfied and the AC-unification constraint $P(x_1 * y) =_{AC} P(b * x)$ can be decomposed into $x_1 * y =_{AC} b * x$. ♦

Definition 2 (Constrained AC-Resolution)

$$\frac{A_1 \vee D_1 \;\; [\![T_1]\!] \qquad \neg A_2 \vee D_2 \;\; [\![T_2]\!]}{D_1 \vee D_2 \;\; [\![T_1 \wedge T_2 \wedge T_3]\!]} \quad \textit{where } T_3 \textit{ is } \; A_1 =_{AC} A_2 \;\wedge\; A_1 \succ D_1 \vee D_2$$

Example 2 An AC-resolution step between clauses $P(x_1 * y) \vee Q(y) \; [\![x_1 =_{AC} a]\!]$ and $\neg P(x_2 * x) \vee Q(x) \; [\![x_2 =_{AC} b]\!]$ produces the clause $Q(y) \vee Q(x) \; [\![T]\!]$, where T is simplified to $x_1 =_{AC} a \wedge x_2 =_{AC} b \wedge x_1 * y =_{AC} x_2 * x$. ♦

The next rule simulates a resolution step with the reflexivity axiom ($x \simeq x$).

Definition 3 (Constrained AC-Reflexion)

$$\frac{\neg(s \simeq t) \vee D \;\; [\![T_1]\!]}{D \;\; [\![T_1 \wedge T_2]\!]} \quad \textit{where } T_2 \textit{ is } \; s =_{AC} t \;\wedge\; (s \simeq t) \succ D$$

Example 3 By applying this AC-reflexion rule to the clause $\neg(x_1 * y \simeq b * x) \vee (x * y \simeq c) \; [\![x_1 =_{AC} a]\!]$, we deduce $(x * y \simeq c) \; [\![T]\!]$, where T is $x_1 =_{AC} a \;\wedge\; x_1 * y =_{AC} b * x \;\wedge\; (x_1 * y \simeq b * x) \succ (x * y \simeq c)$. ♦

The previous two inference rules are the only ones for generating the *empty clause* (denoted $\square$), i.e. a clause with no literals and a satisfiable set of constraints.

The next two inference rules are the paramodulation and the contextual paramodulation. The first one is well known, but the second one is specific to AC theories ; this rule and the extended paramodulation, defined later, simulate the associativity axiom. The experienced reader will remark that these rules simulate the use of extensions of clauses [PS81]. An example of contextual paramodulation (without constraints) is :

$$\frac{(a * b \simeq c) \qquad P((a * d) * b)}{P(c * d)}$$

where the contextual paramodulation is applied in the subterm $(a * d) * b$, which is AC-equal to $(a * b) * d$, and the term d is what we call the context of the paramodulation step.
In the next two rules, the position p, where the paramodulation is applied, has to be non-variable and maximal for AC-operators (see definition of $\mathcal{MP}os$ in Section 3). Indeed, if a (contextual) paramodulation can be applied at a non-maximal position of an AC-operator f, a contextual paramodulation at the corresponding maximal position of f is equivalent. For instance, applying a paramodulation (or contextual paramodulation) in the subterm $f(t_1, t_2)$ of the clause $P(f(f(t_1, t_2), t_3))$ is equivalent to a contextual paramodulation in the subterm $f(f(t_1, t_2), t_3)$ where t_3 is in the context.

Definition 4 (Constrained AC-Paramodulation)

$$\frac{(l \simeq r) \vee D_1 \;\; [\![T_1]\!] \qquad L \vee D_2 \;\; [\![T_2]\!]}{L[x_r]_p \vee D_1 \vee D_2 \;\; [\![T_1 \wedge T_2 \wedge T_3]\!]}$$

where x_r *is a new variable,* $p \in \mathcal{MP}os(L)$
and T_3 *is* $L_{|_p} =_{AC} l \;\wedge\; x_r =_{AC} r \;\wedge\; l \succ r \;\wedge\; (l \simeq r) \succ D_1 \;\wedge\; L \succ D_2$

Note that in the deduced clause, the term r *is replaced by a new variable* (x_r) *to avoid further paramodulation steps inside.*

Example 4 An AC-paramodulation from $(x_1 * y \simeq c) \vee (g(y) \simeq d)$ $[\![x_1 =_{AC} a]\!]$ into $P(x_2 * z) \vee Q(z)$ $[\![x_2 =_{AC} b]\!]$, in the subterm $x_2 * z$, produces the clause $P(x_r) \vee (g(y) \simeq d) \vee Q(z)$ $[\![T]\!]$ where T is transformed into the constraints $x_1 =_{AC} a \ \wedge \ x_2 =_{AC} b \ \wedge \ x_1 * y =_{AC} x_2 * z \ \wedge \ x_r =_{AC} c$. ♦

In the next rule, we use a new kind of variables, called *extension variables* and indexed by an AC-operator, for representing the context of contextual paramodulation steps. These special variables implement the same notion of irreducibility subterms used by Nieuwenhuis and Rubio in [NR94], and we detail their purpose in the following.

Definition 5 (Constrained AC-Contextual Paramodulation)

$$\frac{(l \simeq r) \vee D_1 \ \ [\![T_1]\!] \qquad L \vee D_2 \ \ [\![T_2]\!]}{L[f(x_r, z_f)]_p \vee D_1 \vee D_2 \ \ [\![T_1 \wedge T_2 \wedge T_3 \wedge T_3']\!]}$$

where x_r is a new variable, $p \in \mathcal{MP}os(L)$,
z_f is a new extension variable for the AC-operator f ,
and T_3 is $L_{|_p} =_{AC} f(l, z_f) \ \wedge \ x_r =_{AC} r \ \wedge \ l \succ r \ \wedge \ (l \simeq r) \succ D_1 \ \wedge \ L \succ D_2$
and T_3' is the conjunction of the constraints $\mathcal{H}terms(l, f) \not\subseteq_{AC} \mathcal{H}terms(t, f)$, for each term t in $\mathcal{H}terms(L_{|_p}, f)$ which is not an extension variable for f.

This rule allows to replace only parts of the subterm at position p of the literal L, provided that the operator at p is AC. So, the idea is that L is divided in two parts : the first one has to be AC-unifiable with l, and the second one (referred by z_f) is the context. The constraints on $\mathcal{H}terms$ mean that we did not apply a paramodulation into a non-extension variable of L, assuming we forget the use of maximal positions of AC-operators ; moreover, they guarantee that the root operator of l is f.

Example 5 If we apply a contextual paramodulation from the clause $(x_1 * b \simeq c) \vee (g(b) \simeq d)$ $[\![x_1 =_{AC} a]\!]$ into the clause $P(x_2 * (u * d)) \vee Q(u)$ $[\![x_2 =_{AC} b]\!]$, in the subterm $x_2 * (u * d)$, we deduce $P(x_r * z_*) \vee (g(b) \simeq d) \vee Q(u)$ $[\![T]\!]$. The set of constraints T can be simplified to $x_1 =_{AC} a \ \wedge \ x_2 =_{AC} b \ \wedge \ x_1 * z_* =_{AC} u * d \ \wedge \ x_r =_{AC} c \ \wedge \ \{a, b\} \not\subseteq_{AC} \mathcal{H}terms(u, *)$. ♦

The role of extension variables z_f is important but does not need a strong control. Indeed, in other inference rules, they are considered as other variables. But, in a contextual paramodulation, we do not put back the context, i.e. the non-used subterms, in the deduced clause and it produces the loss of completeness as shown in the next example. To correct this, we allow further contextual paramodulations finding the term to be replaced in this context, marked by an extension variable. This is the reason why the condition on $\mathcal{H}terms$ is not applied to extension variables. As mentioned earlier, extension variables implement the same notion of irreducibility as in [NR94]. Indeed, in the proof of completeness, each term appearing in the constraints is assumed to be irreducible ; but, when a ground term t is associated to an extension variable z_f, the irreducibility is required only for the terms of $\mathcal{H}terms(t, f)$.

Example 6 Given three clauses $(a * b \simeq c)$ (1) , $P((a * a) * (b * b))$ (2) and $\neg P(c * c)$ (3) , we can prove that they form an AC-inconsistent system in the following way :
A contextual paramodulation from (1) into (2) at the subterm $(a * a) * (b * b)$ produces the clause $P(x * z_*) [\![x =_{AC} c \ \wedge \ z_* =_{AC} a * b]\!]$ (4) ; then, another contextual paramodulation from (1) into (4) at the subterm $x * z_*$ permits to deduce $P(y * z_*') [\![z =_{AC} a * b \ \wedge \ z_*' =_{AC} c \ \wedge \ y =_{AC} c]\!]$ (5) . Then, a resolution between (3) and (5) produces the empty clause $\Box [\![z =_{AC} a * b \ \wedge \ z' =_{AC} c]\!]$.
If the condition on $\mathcal{H}terms$ was generalized to variables of extension, we could not deduce the clause (5) and find a contradiction since the left-hand side $a * b$ of the equation would be found in a variable. ♦

But, this is the only role of such extension variables and the next rule has the highest priority.

Definition 6 (Deletion of Extension Variables) *If there is an extension variable z_f in the constraints T of a clause $C [\![T]\!]$ which does not appear in C, this clause has to be replaced by $C [\![T\sigma]\!]$, where σ is the substitution $\{z_f \mapsto z\}$ and z is a new variable.*

This definition is motivated by the following properties :

1. If an extension variable z_f appears only in the constraints of a clause, it cannot be used by a contextual paramodulation step and also loses its special interest.

2. Any extension variable z_f has a single occurrence in a clause : it is created by the contextual paramodulation rule, and, to be duplicated, it has to instantiate another variable. But, since unifiers are not computed, variables are never instantiated and also an extension variable cannot appear twice in a clause.

3. As a consequence, an extension variable may only appear in a clause under the AC-operator for which it is defined.

In the last two inference rules, *a further refinement can be defined.* If the literal L is a negative equation $\neg(s \simeq t)$, and if the position of the (contextual) paramodulation is in the term s, we can add the ordering constraint $s \succ t$, specifying that this inference step has to be applied in the maximal side of the equation.

The following extended paramodulation rule can be viewed as a mutual contextual paramodulation between two clauses, at the head of the left-hand side of their maximal equation.

Definition 7 (Constrained AC-Extended Paramodulation)

$$\frac{(l_1 \simeq r_1) \vee D_1 \;\; [\![T_1]\!] \qquad\qquad (l_2 \simeq r_2) \vee D_2 \;\; [\![T_2]\!]}{(f(x_{r_1}, z_1) \simeq f(x_{r_2}, z_2)) \vee D_1 \vee D_2 \;\; [\![T_1 \wedge T_2 \wedge T_3]\!]}$$

where x_{r_1}, x_{r_2}, z_1 *and* z_2 *are new variables,* $f \in \mathcal{F}_{AC}$
and T_3 *is* $f(l_1, z_1) =_{AC} f(l_2, z_2) \;\wedge\; x_{r_1} =_{AC} r_1 \;\wedge\; x_{r_2} =_{AC} r_2$
$\wedge\; l_1 \succ r_1 \;\wedge\; l_2 \succ r_2 \;\wedge\; (l_1 \simeq r_1) \succ D_1 \;\wedge\; (l_2 \simeq r_2) \succ D_2$
$\wedge\; \mathcal{H}terms(l_1, f) \cap_{AC} \mathcal{H}terms(l_2, f) \;\neq\; \emptyset$
$\wedge\; \mathcal{H}terms(z_1, f) \cap_{AC} \mathcal{H}terms(z_2, f) \;\simeq\; \emptyset$

In the deduced clause, the terms r_1 and r_2 and the contexts z_1 and z_2 are replaced by variables ; it is different from the contextual paramodulation where the context is marked by an extension variable ; indeed, our proof technique for completeness (transfinite semantic trees) has shown that paramodulations in these contexts were useless. The last constraints force the existence of a maximal overlap between l_1 and l_2. In other words, we have to consider the maximal set of shared subterms between l_1 and l_2.

Example 7 If we apply an extended paramodulation between the constrained clauses $(x_1 * b \simeq c) \vee (g(b) \simeq d) \;[\![x_1 =_{AC} a]\!]$ and $(a * x_2 \simeq e) \vee (g(x_2) \simeq b) \;[\![x_2 =_{AC} d]\!]$, the deduced clause is $(x_{r_1} * z_1 \simeq x_{r_2} * z_2) \vee (g(b) \simeq d) \vee (g(x_2) \simeq b) \;[\![T]\!]$, and the simplified form of T is $x_1 =_{AC} a \;\wedge\; x_2 =_{AC} d \;\wedge\; x_{r_1} =_{AC} c \;\wedge\; x_{r_2} =_{AC} e \;\wedge\; z_1 =_{AC} d \;\wedge\; z_2 =_{AC} b$. ♦

So, we have defined a set of inference rules that drastically limits the number of possible deductions, thanks to the ordering and AC-unification constraints, but particularly because we never compute any AC-unifier ; we need only to check the AC-unifiability. In the following, we show that our results also apply to the superposition strategy.

4.2 Superposition Strategy

The previous inference rules can be refined for the superposition strategy. The main difference between superposition and paramodulation is that superposition is only applied into maximal sides of equational literals. So, the three paramodulation rules are replaced by rules of *Constrained AC-Superposition*, *Constrained AC-Contextual Superposition* and *Constrained AC-Extended Superposition*. Their definitions differ from the paramodulation ones by :

1. If a (contextual) superposition is applied from an equation $(l \simeq r)$ into the term s of a positive equational literal $(s \simeq t)$, the constraints $s \succ t$ and $(s \simeq t) \succ (l \simeq r)$ are added.

2. In inference rules, the constraints representing the maximality of an equational literal $(l \simeq r)$ or $\neg(l \simeq r)$ (where l is assumed to be greater than r) in a clause have to be modified :
   - $(l \simeq r)$ is maximal in a clause $(l \simeq r) \vee D$ if $(l \simeq r) \succ L$ for any positive literal L of D and,

$l \succ s$ and $l \succ t$ for any negative literal $\neg(s \simeq t)$ of D.

- $\neg(l \simeq r)$ is maximal in a clause $\neg(l \simeq r) \vee D$ if $(l \simeq r) \succ \neg L$ for any negative literal L of D and, $(l \simeq l) \succ L$ for any positive literal L of D.

3. The right-hand sides are no more replaced by new variables ; they appear explicitly in the deduced clause.

For completeness, a new rule has to be added :

Definition 8 (Constrained AC-Equational Factoring)

$$\frac{(u_1 \simeq v_1) \vee (u_2 \simeq v_2) \vee D \;\; [\![T_1]\!]}{(u_1 \simeq v_1) \vee \neg(v_1 \simeq v_2) \vee D \;\; [\![T_1 \wedge T_2]\!]}$$

where T_2 *is* $u_1 =_{AC} u_2 \;\wedge\; u_1 \succ v_1 \;\wedge\; (u_1 \simeq v_1) \succ (u_2 \simeq v_2) \vee D$

This inference rule transforms a clause so that it does no more contain two positive equational literals with the same left-hand side.

Nieuwenhuis and Rubio have defined a similar system of inference rules for superposition [NR94]. Both inference systems do not compute any AC-unifier and permit l to be in a variable of $L_{|_p}$ for the contextual superposition rule. But, they do not restrict it to extension variables. So, they allow useless inference steps which consist in applying a superposition step in a variable, as exemplified by the following :

Example 8 Given the following system,

$$((a*a)*x \simeq a*x) \;\; (1) \qquad (b*c \simeq c) \;\; (2) \qquad P((a*b)*(a*c)) \;\; (3)$$

we can deduce the clause $P(a*x) [\![x =_{AC} b*c]\!]$ (4) by a superposition from (1) into (3). Then they allow a contextual superposition from (2) into (4) which produces $P(c*y) [\![y =_{AC} a]\!]$ (5) . If we forget that we are working at maximal positions of AC-operators, this last step corresponds to a superposition from (2) into the variable x, which is well-known to be useless. With our constraints, we forbid it and a similar clause can be derived by a contextual superposition from (2) into (3) which produces $P(c*z_*) [\![z_* =_{AC} a*a]\!]$ (4′) and a superposition from (1) into (4′) which produces $P(a*x) [\![x =_{AC} c]\!]$ (5′) .
The clauses (5) and(5′) are apparently different, but if we use the propagation of constrained subterms in both sequences of deduction, the terms a and c appeared in the constraints. So, the deduced clauses should be $P(y*x) [\![x =_{AC} c \;\wedge\; y =_{AC} a]\!]$ (5″) . ♦

Moreover, Nieuwenhuis and Rubio do not have our restrictions on $\mathcal{H}terms$ for the contextual and extended superpositions.

Now, we have to introduce the lifting lemma for our inference rules, since the proof of completeness, based on transfinite semantic trees, is first obtained in the ground case (see Section 6).

4.3 Lifting Lemma

The purpose of a lifting lemma is to show that inferences on ground clauses are instances of inferences on the corresponding general clauses. But, lifting a paramodulation step may be impossible, when it replaces a subterm introduced by the substitution which makes the clause ground. The solution of [Pet83] is to restrict paramodulation to ground clauses where variables are substituted by irreducible terms w.r.t. a term rewriting system. The technique has been applied to prove the AC-completeness of inference rules described in [RV93], and we are going to generalize it for the constrained paramodulation and superposition strategies.

The next proposition is a consequence of the stability property of the ordering $\succ$.

Proposition 9 *Let A and B be two objects. If $A\sigma \succ B\sigma$ for a substitution σ, then $A \not\preceq B$.*

A consequence of this proposition is :

Corollary 10 *Let σ be a ground AC-unifier of objects $A_1, \ldots, A_m$, and B_1 and B_2 be two objects such that : $B_1\sigma \succ B_2\sigma$. There exists an AC-mgu τ of $A_1, \ldots, A_m$ such that : $B_1\tau \not\preceq B_2\tau$.*

This corollary permits to lift ordering constraints, and the substitution τ, defined by $\sigma = \tau\rho$ (for a ground substitution ρ), allows us to lift the AC-unification constraints.

We have to check that whenever the conditions for applying inference rules to the ground level are satisfied, they are also valid at the general level. These conditions are (for paramodulation and superposition rules) :

1. p is a non-variable position in L
2. The constraints on $\mathcal{H}terms$ are satisfiable

The first condition remains valid at the general level since it concerns an occurrence in C (for a constrained clause $C [\![T]\!]$). For the second ones, using the substitutions σ, τ and ρ defined above, and given two terms s and t, if the constraint $\mathcal{H}terms(s\sigma, f) \not\subseteq_{AC} \mathcal{H}terms(t\sigma, f)$ is satisfied, the constraint $\mathcal{H}terms(s\tau, f) \not\subseteq_{AC} \mathcal{H}terms(t\tau, f)$ is satisfiable by ρ, since $\sigma = \tau\rho$.
The reasoning is the same if we test emptiness or non-emptiness of the intersection between $\mathcal{H}terms(s\sigma, f)$ and $\mathcal{H}terms(t\sigma, f)$. So, all these constraints are satisfiable at the general level and we can state the main lemma :

Lemma 11 (Lifting Lemma) *Let $C_1 [\![T_1]\!], \ldots, C_n [\![T_n]\!]$ be clauses and σ a ground solution of all the T_i, $i = 1 \ldots n$. If an inference rule R applies to $C_1 [\![\sigma]\!], \ldots, C_n [\![\sigma]\!]$ and produces a clause $C [\![\sigma_c]\!]$, the same inference rule R applies to clauses $C_1 [\![T_1]\!], \ldots, C_n [\![T_n]\!]$ and generates a clause $D [\![T]\!]$ such that C and D are AC-equal, and σ_c is a solution of T.*

5 Simplification Rules

In this section, we define subsumption and simplification rules compatible with the constrained inference rules described in the previous section. But, these rules have strong conditions to be applied ; we also propose new ones which are more often applicable. These rules are fundamental for the efficiency of our theorem prover, since they allow us to delete redundant clauses.

5.1 Constrained Subsumption

The purpose of subsumption is to eliminate clauses which are redundant. In the classical definition, *a clause C_1 subsumes a clause C_2 if there is a substitution ρ such that $C_1\rho \subseteq C_2$*. Since clauses are considered as multisets, C_2 has at least as many literals as C_1. It is extended to :

Definition 12 (Constrained AC-Subsumption) *A constrained clause $C_1 [\![T_1]\!]$ subsumes a constrained clause $C_2 [\![T_2]\!]$ if there is a solution σ_1 of T_1 such that : for each solution σ_2 of T_2, there is a substitution ρ such that $C_1\sigma_1\rho$ is a subset of $C_2\sigma_2$, i.e. $\exists\sigma_1 \in \mathcal{S}ol(T_1)$, $\forall\sigma_2 \in \mathcal{S}ol(T_2)$, $\exists\rho$, $C_1\sigma_1\rho \subseteq_{AC} C_2\sigma_2$*

The standard way for applying this rule consists in the deletion of the *strictly subsumed* clauses, i.e. of clauses which are subsumed by a clause, but which do not subsume this clause. However, in our case, this causes the loss of the completeness. Indeed, as shown in [BGLS92], it may happen when the terms of the range of $\sigma_1\rho$, $\mathcal{R}an(\sigma_1\rho)$, are not included in those of σ_2, $\mathcal{R}an(\sigma_2)$. So, first, we check that this condition is satisfied.

Definition 13 (Application of the Strict Constrained AC-Subsumption) *If a clause $C_1 [\![T_1]\!]$ subsumes strictly a clause $C_2 [\![T_2]\!]$ and if $\mathcal{R}an(\sigma_1\rho) \sqsubseteq_{AC} \mathcal{R}an(\sigma_2)$, the clause $C_2 [\![T_2]\!]$ is deleted.*

This definition has to be modified for taking into account the extension variables introduced by the contextual paramodulation and superposition rules (Definition 5). The only modification is in the calculus of $\mathcal{R}an(\sigma_i)$ in previous definition : if z_f is an extension variable appearing under the AC-operator f in C_i, the set of terms $\mathcal{H}terms(z_f\sigma_i, f)$ will replace $z_f\sigma_i$ in $\mathcal{R}an(\sigma_i)$.

From now on, whenever we will talk about the range of a substitution, it will refer to this new definition.

5.2 Constrained Simplification

Usually, a clause $L \vee D$ is simplified by an equality $(l \simeq r)$ at a position p of the literal L if there is a substitution ρ such that : $L_{|_p} = l\rho$ and $l\rho \succ r\rho$. This definition becomes :

Definition 14 (Constrained AC-Simplification) *A clause* $(l \simeq r) \vee D_1 \ [\![T_1]\!]$ *simplifies a clause* $L \vee D_2 \ [\![T_2]\!]$ *at a position* $p \in \mathcal{MP}os(L)$ *if :*

$$\exists \sigma_1 \in \mathcal{S}ol(T_1) \ , \ \forall \sigma_2 \in \mathcal{S}ol(T_2) \ , \ \exists \rho \in AC\_match\{l\sigma_1, L_{|_p}\sigma_2\} \text{ solution of } T_1\sigma_1 \ ,$$
$$\begin{cases} D_2\sigma_2 =_{AC} D_2'\sigma_2 \vee D_2''\sigma_2 \quad \text{and} \quad D_1\sigma_1\rho \subseteq_{AC} D_2'\sigma_2 \\ l\sigma_1\rho \succ r\sigma_1\rho \\ (l \simeq r)\sigma_1\rho \prec L\sigma_2 \vee D_2''\sigma_2 \end{cases}$$

This defines a conditional simplification, since the simplifying clause may contain other literals than $(l \simeq r)$, provided that they are in the instances of the clause D_2 too. Notice that set inclusion is used rather than multiset inclusion in the definition. It means that there may be a literal which appears twice in $D_1\sigma_1\rho$ and only once in $D_2'\sigma_2$. Moreover, our proof technique imposes that a literal of $L \vee D_2$, which is not in D_1, has to be greater than $(l \simeq r)$.

Definition 15 (Constrained AC-Contextual Simplification) $(l \simeq r) \vee D_1 \ [\![T_1]\!]$ *simplifies contextually* $L \vee D_2 \ [\![T_2]\!]$ *at a position* $p \in \mathcal{MP}os(L)$ *if :*

$$\exists \sigma_1 \in \mathcal{S}ol(T_1) \ , \ \exists \nu \in AC\_mgus\{f(y,z), L_{|_p}\} \ ,$$
$$\forall \sigma_2 \in \mathcal{S}ol(T_2) \ , \ \exists \mu \ , \ \sigma_2 \simeq \nu\mu \ , \ \exists \rho \in AC\_match\{l\sigma_1, y\nu\mu\} \text{ solution of } T_1\sigma_1 \ ,$$
$$\begin{cases} \text{either } \mathcal{H}ead(y\nu) = \mathcal{H}ead(l\sigma_1\rho) = f \in \mathcal{F}_{AC} \\ \quad \text{or } y\nu \text{ is an extension variable for } f \text{ and } \mathcal{H}ead(y\nu\mu) = f \\ D_1\sigma_1\rho \subseteq_{AC} D_2\sigma_2 \\ l\sigma_1\rho \succ r\sigma_1\rho \end{cases}$$

This rule is the counterpart of the contextual paramodulation. But, here we cannot use the notion of extension variable to express the context ; we have to split $L_{|_p}$ explicitly into two parts by calculating AC-unifiers ($AC\_mgus\{f(y,z), L_{|_p}\}$) : one is matched by $l\sigma_1$ ($y\nu$), and the other (the context) has to be reinserted in the deduced clause ($z\nu$). The substitution μ is needed to relate ν and σ_2. We do not have to check that there is an atom in $L\sigma_2 \vee D_2\sigma_2$ which is greater than $(l \simeq r)\sigma_1\rho$, since $L\sigma_2$ obviously satisfies this condition.
The next example shows the loss of completeness if we used the technique of extension variables, like in the contextual paramodulation.

Example 9 Given the AC-inconsistent system, where f is AC and $c \succ d$,

$$\begin{array}{llll} P(f(f(a,d), f(b,c))) & (1) & (f(a,b) \simeq b) & (2) \\ \neg P(f(f(d,d), b)) & (3) & (c \simeq d) & (4) \end{array}$$

The clause (1) can be simplified by (2) into $P(f(b, z_f)) \, [\![z_f =_{AC} f(c,d)]\!] \ (1)$. But then, there is no way to prove the AC-inconsistency. With the previous definition, the simplified clause is $P(f(b, f(c,d))) \ (1)$, which can be simplified by (4) into $P(f(b, f(d,d))) \ (1)$. A resolution step between (1) and (3) yields the contradiction. ♦

The classical application of such simplification rules is to replace the clause $L \vee D$ by $L[r]_p \vee D$. But, we encounter the same problems as for the subsumption ; so a condition on the ranges of substitutions is again necessary :

Definition 16 (Application of the Constrained AC-Simplifications) *If a clause* $C_1 [\![T_1]\!]$ *simplifies a clause* $C_2 [\![T_2]\!]$ *at a position* p *and if* $\mathcal{R}an(\sigma_1\rho) \sqsubseteq_{AC} \mathcal{R}an(\sigma_2)$, *the clause* $C_2 [\![T_2]\!]$ *is replaced by* $L[r]_p \vee D_2 \ [\![T_2 \ \wedge \ T_1'\sigma_1 \ \wedge \ L_{|_p} =_{AC} l\sigma_1 \ \wedge \ \sigma_{1|_{\mathcal{V}ar(r)}}]\!]$ *for the simplification rule, and* $L[f(r, z\nu)]_p \vee D_2 \ [\![T_2 \ \wedge \ T_1'\sigma_1 \ \wedge \ y\nu =_{AC} l\sigma_1 \ \wedge \ \sigma_{1|_{\mathcal{V}ar(r)}} \ \wedge \ \nu_{|_{\mathcal{V}ar(L_{|_p})}}]\!]$ *for the contextual simplification rule ;* T_1' *is the subset of non AC-unification constraints of* T_1*,*

5.3 Special Subsumption and Simplification

We saw that conditions on ranges of substitutions have to be respected to keep the completeness of our inference rules. But, they limit the number of applicable simplifications. To balance with this, we modify the definitions of subsumption and simplification rules so that, even if conditions on ranges are not satisfied, the rules can be applied provided that we partially instantiate the skeleton of the simplifying clause C_1 [BGLS92].
However, this is not sufficient since a constraint may have several solutions. Hence, to partially instantiate C_1 may produce the loss of some of these substitutions. The solution we propose is to create a new clause, the instantiation of $C_1 [\![\sigma_1]\!]$, and to transform $C_1 [\![T_1]\!]$ into $C_1 [\![T_1 - \sigma_1]\!]$, i.e. specifying that σ_1 is no more a solution of the constraints. To denote this, we can either use AC-disunification, i.e. $C_1 [\![T_1 \wedge \{ \bigvee_{x \in \mathcal{D}om(\sigma_1)} x \neq_{AC} x\sigma_1 \}]\!]$, or ordering constraints, i.e. $C_1 [\![T_1 \wedge \{ \bigvee_{x \in \mathcal{D}om(\sigma_1)} x \prec x\sigma_1 \vee x \succ x\sigma_1 \}]\!]$. But, the second technique raises problems with the new variables introduced by the AC-unification algorithm.

So, for the subsumption and the simplification rules, if $\mathcal{R}an(\sigma_1 \rho) \not\sqsubseteq_{AC} \mathcal{R}an(\sigma_2)$, the actions are :

- to create the new clause $C_1 \tau [\![\sigma_1' \wedge T_1' \sigma_1']\!]$, where T_1' is the subset of non AC-unification constraints of T_1, $\sigma_1 \simeq \tau\sigma_1'$ and $\mathcal{R}an(\sigma_1'\rho) \sqsubseteq_{AC} \mathcal{R}an(\sigma_2)$,
- to transform the clause $C_1 [\![T_1]\!]$ into $C_1 [\![T_1 - \sigma_1]\!]$,
- for simplification and contextual simplification rules, in the deduced clause, to instantiated the right-hand side of the equation r by τ and to use σ_1' instead of σ_1 in the constraints.

5.4 Some Particular Simplification Rules

The simplification rules defined previously are applied only when all the instances of a clause can be simplified by the same clause, and at the same position. This is very restrictive, and we can imagine that, if only one instance of a clause is simplifiable, it may be interesting to simplify it. This is why we define *local simplification* rules, by specifying that we consider only one solution ($\exists \sigma_2 \in \mathcal{S}ol(T_2)$). Applications of local simplifications consist first in creating a new clause, the result of the simplification step, second in transforming the clause $L \vee D_2 \; [\![T_2]\!]$, by adding a constraint to specify that σ_2 is discarded from the set of solutions of T_2.

Example 10 If we apply a local simplification from the clause $(a * b \simeq c) [\![ ]\!]$ (1) into the clause $P(a * y, z) [\![y * z =_{AC} d * b]\!]$ (2) , using the substitution $\sigma_2 = \{ y \mapsto b, \; z \mapsto d \}$, the generated clause is $P(c, z) [\![y =_{AC} b \; \wedge \; z =_{AC} d]\!]$ (3) , and the clause (2) is replaced by $P(a * y, z) [\![T]\!]$, where T is the set of constraints $y * z =_{AC} d * b \; \wedge \; \{ y \neq_{AC} b \; \vee \; z \neq_{AC} d \}$, transformed to $y =_{AC} d \; \wedge \; z =_{AC} b$. ⧫

With the same principle, we define the *local strict subsumption* rule, which is applied when an instance of a clause is strictly subsumed.

An important simplification is that, given a constrained clause $C [\![T]\!]$ and a solution σ of T, *if the clause $C [\![\sigma]\!]$ is simplifiable in $\mathcal{R}an(\sigma)$, it is redundant.* Directly from the proof, we see that we never need ground instances of clauses that are reducible in their substitution part. This deletion is also *locally* applied, and we notice that, if every instance of a clause is simplifiable in its substitution (even by different clauses) this clause is redundant and can be deleted.

Finally, a number of other classical simplification rules can be adapted (locally or not) to the constrained deduction. For instance, we can delete clauses which contain a literal AC-equal to an identity ($s \simeq s$) and clauses which contain literals L and $\neg L'$, where L and L' are AC-equal.
Another one is the *clausal simplification* rule, which deletes all instances of a literal L in every clause of S, if $\neg L$ is a clause of S. We can also delete every instance of $\neg(x \simeq x)$ (it is a kind of *trivial reflexion*). A clausal simplification step may be viewed as a resolution step, followed by the deletion of one of the parent clauses.

6 Refutational Completeness

Let INF be the set of inference rules and simplification rules described in previous sections. A *derivation* from a set of clauses S_0 is a sequence $S_0, S_1, \ldots$ of sets of constrained clauses obtained by successive applications of rules of INF. Let S^* denote $\bigcup_{i\geq 0} S_i$. A derivation $S_0, S_1, \ldots$ is *fair* if each clause $C\, [\![T]\!]$ of $\bigcap_{i\geq j} INF(S_i)$ is subsumed by a clause $C'\, [\![T']\!]$ of S^*. A clause $C\, [\![T]\!]$ is *persistent* in a fair derivation $S_0, S_1, \ldots$ if there is an index k such that $C\, [\![T]\!]$ belongs to $\bigcap_{i\geq k} S_i$. A set of clauses S is *AC-unsatisfiable* if it has no *AC-consistent* model, i.e. no model coherent with the AC axioms.
The following theorem states that INF is refutationally complete in AC theories.

Theorem 17 *Let S be an AC-unsatisfiable set of clauses with empty constraints. Then, every fair derivation from S generates the empty clause.*

The method we use for proving this theorem is based on the notion of transfinite semantic trees introduced in [Rus89, HR91], which was adapted to AC theories in [RV93] for the paramodulation strategy and in [Vig94b] for the superposition strategy. We present here only a sketch of the proof, but it is detailed in the Appendix A.
Sketch of Proof : *In the empty theory, the proof is done by defining the transfinite tree of all possible interpretations of the Herbrand base $\mathcal{A}(\mathcal{P}, \mathcal{F})$, and considering its maximal consistent semantic tree (MCT), i.e. the maximal subtree of interpretations which do not falsify a ground instance of a clause of S^*. Then, the rightmost path of that subtree is proved empty, which implies that MCT is empty too, and also that the empty clause belongs to S^*.*

But, this rightmost path of MCT may not be a model for AC theories. So, we define by induction a sequence Π of nodes in MCT, which is the rightmost AC-consistent path. Then, we prove that Π is empty, by showing that its last node falsifies a clause deduced by an inference step between clauses labeling successors of nodes of Π. MCT has to be empty too, and also S^* contains the empty clause.

The problem with the basic strategy is that no paramodulation is allowed in the constrained part of a clause. Indeed, when a ground instance $C\, [\![\sigma]\!]$ of a constrained clause labels a node I, it means that the interpretation I falsifies the ground clause $C\sigma$; but, to show that MCT is empty, a paramodulation in a subterm introduced by the substitution σ in $C\sigma$ may be needed. So, to avoid this problem, we impose that a ground clause $C\, [\![\sigma]\!]$ labels a node I of a semantic tree if $C\sigma$ is falsified by this interpretation and if each term of $\mathcal{R}an(\sigma)$ is irreducible by the ground equations of I.

Another problem is raised by the simplification rules : the clauses used by an inference step may never appear in the same S_i. But, fortunately, this never keeps from generating the empty clause, since we show that every node, successor of a node of Π, can be labeled by a persistent clause, i.e. a clause which is never simplified. $\square$
The case of non-empty initial constraints can be treated as in [NR92b].

7 Example of Resolution : SAM's Lemma

We present a fully detailed example of a proof by refutation, using inference rules described in Sections 4 and 5. However, these rules are not yet completely implemented in our prover `DATAC` ; indeed, AC-unifications are eagerly solved. Note that, in our implementation, clauses $\neg A_1 \vee \ldots \vee \neg A_m \vee B_1 \vee \ldots \vee B_n$ are represented as sequents $A_1, \ldots, A_m \Rightarrow B_1, \ldots, B_n$. So, superposition and simplification rules are be decomposed in left- and right- rules, whether they are applied to the left or to the right of the sign $\Rightarrow$.
`DATAC` is written in CAML Light, a functional language of the ML family. It runs on SUN, HP and IBM PC workstations. The AC-unification and AC-matching procedures are based on Stickel's and Hullot's algorithms respectively.

The example below is the well known SAM's Lemma, posed by Bumcroft in 1965 as an open problem in lattice theory, and taken from the paper *Semi-Automated Mathematics* of Guard et al. [GOBS69]. We use its equational formulation from McCune [McC88]. To our knowledge, the only completely automatic proof of an equational version of this lemma was done by RRL [ZK88]. The lemma states that :

if x and $meet(u, v)$ are complements, and y and $join(u, v)$ are complements,
then $meet(join(y, meet(x, v)), join(y, meet(x, u)))$ is y.

Operators *meet* and *join*, written "." and "+" in the proof, are associative and commutative. The operator $COMP$, which denotes the complement, is commutative. The greatest element is 1. The least element is 0. Lattices are characterized by clauses 1 to 8, modularity by 9 ; the complement is defined by clauses 10 to 12, and the skolemized negation of the theorem is expressed by clauses 13 to 15. The ordering used is a variant of the APO, with the precedence $COMP >_p \simeq$ and $. >_p + >_p B >_p A >_p R_2 >_p R_1 >_p 1 >_p 0$.

>> Initial set of clauses :

Clause 1 : $\Rightarrow\ x_1.x_1 \simeq x_1$ *Clause 2 :* $\Rightarrow x_1 + x_1 \simeq x_1$

Clause 3 : $\Rightarrow\ x_1.(x_1 + x_2) \simeq x_1$ *Clause 4 :* $\Rightarrow x_1 + (x_1.x_2) \simeq x_1$

Clause 5 : $\Rightarrow\ x_1.0 \simeq 0$ *Clause 6 :* $\Rightarrow x_1 + 0 \simeq x_1$

Clause 7 : $\Rightarrow\ x_1.1 \simeq x_1$ *Clause 8 :* $\Rightarrow x_1 + 1 \simeq 1$

Clause 9 : $x_1.x_2 \simeq x_1\ \Rightarrow\ x_2.(x_1 + x_3) \simeq x_1 + (x_3.x_2)$

Clause 10 : $COMP(x_1, x_2)\ \Rightarrow\ x_1.x_2 \simeq 0$ *Clause 11 :* $COMP(x_1, x_2)\ \Rightarrow\ x_1 + x_2 \simeq 1$

Clause 12 : $x_1 + x_2 \simeq 1\ ,\ x_1.x_2 \simeq 0\ \Rightarrow\ COMP(x_1, x_2)$

Clause 13 : $\Rightarrow\ COMP(R_1, A + B)$ *Clause 14 :* $\Rightarrow\ COMP(R_2, A.B)$

Clause 15 : $(R_1 + (R_2.A)).(R_1 + (R_2.B)) \simeq R_1\ \Rightarrow$

>> AC-Resolution between 10 and 13

Clause 16 : $\Rightarrow\ z_1.z_2 \simeq 0\quad [\![\ z_1 =_{AC} R_1\ \wedge\ z_2 =_{AC} B + A\ ]\!]$

>> AC-Resolution between 10 and 14

Clause 18 : $\Rightarrow\ z_1.z_2 \simeq 0\quad [\![\ z_1 =_{AC} R_2\ \wedge\ z_2 =_{AC} B.A\ ]\!]$

>> Left AC-Superposition from 9 into 15

Clause 99 : $z_1.z_2 \simeq R_1\ ,\ z_1 + (z_3.z_2) \simeq R_1\ \Rightarrow$

$[\![\ z_1 =_{AC} R_1\ \wedge\ z_2 =_{AC} R_1 + (R_2.B)\ \wedge\ z_3 =_{AC} R_2.A\ ]\!]$

>> Left AC-Clausal Simplification in 99 thanks to 3

Clause 99 : $z_1 + (z_3.z_2) \simeq R_1\ \Rightarrow\quad [\![\ z_1 =_{AC} R_1\ \wedge\ z_2 =_{AC} R_1 + (R_2.B)\ \wedge\ z_3 =_{AC} R_2.A\ ]\!]$

>> AC-Extended Superposition between 3 and 9

Clause 123 : $x_1.z_1 \simeq x_1\ \Rightarrow\ (x_1 + (x_4.z_1)).x_3 \simeq x_3.z_2$

$[\![\ z_1 =_{AC} x_2 + x_3\ \wedge\ z_2 =_{AC} x_4 + x_1\ \wedge\ \mathcal{H}terms(x_3, .)\ \cap_{AC}\ \{x_4 + x_1\} \simeq \emptyset\ ]\!]$

>> Left AC-Contextual Superposition from 3 into 123

Clause 170 : $x_1.z_. \simeq z_3\ \Rightarrow\ (z_3 + (x_4.z_1)).x_3 \simeq x_3.z_2$

$[\![\ z_1 =_{AC} x_2 + x_3\ \wedge\ z_2 =_{AC} x_4 + (x_1.x_2)\ \wedge\ z_3 =_{AC} x_1.x_2\ \wedge\ z_. =_{AC} x_2$

$\wedge\ \mathcal{H}terms(x_3, .)\ \cap_{AC}\ \{x_4 + (x_1.x_2)\} \simeq \emptyset\ \wedge\ \{x_2 + x_3\}\ \not\subseteq_{AC}\ \mathcal{H}terms(x_1, .)$

$\wedge\ ((x_1.x_2).(x_2 + x_3) \simeq x_1.x_2)\ \succ\ (((x_1.x_2) + (x_4.(x_2 + x_3))).x_3 \simeq x_3.(x_4 + (x_1.x_2)))\ ]\!]$

>> Trivial AC-Reflexion in 170

Clause 170 : $\Rightarrow\ (z_3 + (x_4.z_1)).x_3 \simeq x_3.z_2$

$[\![\ z_1 =_{AC} x_2 + x_3\ \wedge\ z_2 =_{AC} x_4 + (x_1.x_2)\ \wedge\ z_3 =_{AC} x_1.x_2$

$\wedge\ \mathcal{H}terms(x_3, .)\ \cap_{AC}\ \{x_4 + (x_1.x_2)\} \simeq \emptyset\ \wedge\ \{x_2 + x_3\}\ \not\subseteq_{AC}\ \mathcal{H}terms(x_1, .)$

$\wedge\ ((x_1.x_2).(x_2 + x_3) \simeq x_1.x_2)\ \succ\ (((x_1.x_2) + (x_4.(x_2 + x_3))).x_3 \simeq x_3.(x_4 + (x_1.x_2)))\ ]\!]$

>> Right AC-Superposition from 16 into 170

Clause 322 : $\Rightarrow\ z_1.z_2 \simeq (z_3 + 0).z_1$

$[\![\ z_1 =_{AC} A\ \wedge\ z_2 =_{AC} R_1 + (x_1.B)\ \wedge\ z_3 =_{AC} x_1.B\ \wedge\ \{A + B\}\ \not\subseteq_{AC}\ \mathcal{H}terms(x_1, .)\ ]\!]$

>> Right AC-Simplification from 6 into 322

Clause 322 : $\Rightarrow\ z_1.z_2 \simeq z_3.z_1$

$[\![\ z_1 =_{AC} A\ \wedge\ z_2 =_{AC} R_1 + (x_1.B)\ \wedge\ z_3 =_{AC} x_1.B\ \wedge\ \{A + B\}\ \not\subseteq_{AC}\ \mathcal{H}terms(x_1, .)\ ]\!]$

>> Left AC-Contextual Simplification from 322 into 99

Clause 99 : $z_1 + ((z_3.z_2).z_4) \simeq R_1\ \Rightarrow\quad [\![\ z_1 =_{AC} R_1\ \wedge\ z_2 =_{AC} A\ \wedge\ z_3 =_{AC} R_2.B\ \wedge\ z_4 =_{AC} R_2\ ]\!]$

>> Left AC-Contextual Simplification from 1 into 99

Clause 99 : $z_1 + (z_4.(z_2.z_3)) \simeq R_1\ \Rightarrow\quad [\![\ z_1 =_{AC} R_1\ \wedge\ z_4 =_{AC} R_2\ \wedge\ z_3 =_{AC} B\ \wedge\ z_2 =_{AC} A\ ]\!]$

>> Left AC-Simplification from 18 into 99

Clause 99 : $z_1 + 0 \simeq R_1\ \Rightarrow\quad [\![\ z_1 =_{AC} R_1\ ]\!]$

>> Left AC-Clausal Simplification in 99 thanks to 6

Clause 99 : $\square$

>> The initial system is AC-inconsistent <<

To give some statistics on this proof, from 15 initial clauses, 187 applications of inference rules and 474 simplifications, 307 clauses were generated ; it shows that our prover is not enough optimized for the choice of inferences to try. But, the more interesting point is that only 103 clauses were retained. These deletions are caused by simplifications in constraints (71), forward subsumptions (16) and eliminations of trivial clauses (132). Moreover, forward subsumption and the detection of trivial clauses have avoided the creation of 146 redundant clauses.

8 Conclusion and Further Works

We have designed a refutationally complete inference system for AC theories which never compute AC-unifiers and which incorporates basicness and ordering requirements. These refinements are expressed with constraints in a very natural way, and we have shown that Constrained AC-paramodulation is compatible with simplification rules. Such rules are fundamental for efficiency of theorem provers. Another way to use our inference rules would be to build the precedence on operators while resolving a system, such as some implementations of Knuth-Bendix. Some further optimizations are possible, as the propagation of blocked subterms in a clause [BGLS92]. For instance, in the introductory example, the subterm b may be blocked in the atom $Q(\boxed{a}, b)$ to obtain the clause $P(\boxed{b}) \vee Q(\boxed{a}, \boxed{b})$, since it is already blocked in $P(\boxed{b})$. Using constraints, it would be denoted by $P(y) \vee Q(x, y)\, [\![\, x =_{AC} a \;\wedge\; y =_{AC} b\,]\!]$.

These strategies have been implemented in our theorem prover **DATAC**, and experiments with non-trivial examples are successful. The method used here to prove completeness of inference systems in AC theories has been adapted to other strategies such as superposition [Rus91]. But, an essential point to be studied and implemented is the propagation of ordering constraints.

Acknowledgments : I wish to thank Michaël Rusinowitch and Claude and Hélène Kirchner for their pertinent remarks and their support for this work. Especially, I would like to thank Robert Nieuwenhuis and Albert Rubio for discussions on topics related to this paper.

References

[BG90] 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.

[BGLS92] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation and superposition. In *Proceedings 11th International Conference on Automated Deduction, Saratoga Springs (N.Y., USA)*, pages 462–476, 1992.

[BL79] A. M. Ballantyne and D. S. Lankford. The refutation completeness of blocked permutative narrowing and resolution. In *Proc. of Fourth Workshop on Automated Deduction, Austin, Texas*, 1979.

[Bra75] D. Brand. Proving theorems with the modification method. *SIAM Journal of Computing*, 4:412–430, 1975.

[Bür91] H.-J. Bürckert. *A Resolution Principle for a Logic with Restricted Quantifiers*, volume 568 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1991.

[DJ90] 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.

[GOBS69] J. R. Guard, F. C. Oglesby, J. H. Bennett, and Settle. Semi-automated mathematics. *Journal of the ACM*, 16:49–62, 1969.

[HR91] 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.

[Hul80] J.-M. Hullot. *Compilation de Formes Canoniques dans les Théories équationelles*. Thèse de Doctorat de Troisième Cycle, Université de Paris Sud, Orsay (France), 1980.

[JK86] 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.

[JK91] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: a rule-based survey of unification. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic. Essays in honor of Alan Robinson*, chapter 8, pages 257–321. MIT Press, Cambridge (MA, USA), 1991.

[KK89] C. Kirchner and H. Kirchner. Constrained equational reasoning. In *Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation, Portland (Oregon)*, pages 382–389. ACM Press, July 1989. Report CRIN 89-R-220.

[KKR90] C. Kirchner, H. Kirchner, and M. Rusinowitch. Deduction with symbolic constraints. *Revue d'Intelligence Artificielle*, 4(3):9–52, 1990. Special issue on Automatic Deduction.

[KMN85] D. Kapur, D. R. Musser, and P. Narendran. Only prime superpositions need be considered in the Knuth-Bendix procedure, 1985. Computer Science Branch, Corporate Research and Development, General Electric, Schenectady, New York.

[McC88] W. McCune. Challenge equality problems in lattice theory. In E. Lusk and R. Overbeek, editors, *Proceedings 9th International Conference on Automated Deduction, Argonne (Ill., USA)*, volume 310 of *Lecture Notes in Computer Science*, pages 704–709. Springer-Verlag, 1988.

[MN90] U. Martin and T. Nipkow. Ordered rewriting and confluence. In M. E. Stickel, editor, *Proceedings 10th International Conference on Automated Deduction, Kaiserslautern (Germany)*, volume 449 of *Lecture Notes in Computer Science*, pages 366–380. Springer-Verlag, 1990.

[NR91] P. Narendran and M. Rusinowitch. Any ground associative-commutative theory has a finite canonical system. In R. V. Book, editor, *Proceedings 4th Conference on Rewriting Techniques and Applications, Como (Italy)*. Springer-Verlag, 1991.

[NR92a] 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.

[NR92b] R. Nieuwenhuis and A. Rubio. Theorem proving with ordering constrained clauses. In D. Kapur, editor, *Proceedings of CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 477–491. Springer-Verlag, 1992.

[NR94] R. Nieuwenhuis and A. Rubio. AC-superposition with constraints: no AC-unifiers needed. In A. Bundy, editor, *Proceedings 12th International Conference on Automated Deduction, Nancy (France)*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 545–559. Springer-Verlag, June 1994.

[Pau92] E. Paul. A general refutational completeness result for an inference procedure based on associative-commutative unification. *Journal of Symbolic Computation*, 14(6):577–618, 1992.

[Pet83] G. Peterson. A technique for establishing completeness results in theorem proving with equality. *SIAM Journal of Computing*, 12(1):82–100, 1983.

[Pet90] G. E. Peterson. Complete sets of reductions with constraints. In M. E. Stickel, editor, *Proceedings 10th International Conference on Automated Deduction, Kaiserslautern (Germany)*, volume 449 of *Lecture Notes in Computer Science*, pages 381–395. Springer-Verlag, 1990.

[Plo72] G. Plotkin. Building-in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[PS81] G. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, 28:233–264, 1981.

[RN93] 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.

[Rus89] M. Rusinowitch. *Démonstration automatique-Techniques de réécriture*. InterEditions, 1989.

[Rus91] M. Rusinowitch. Theorem-proving with resolution and superposition. *Journal of Symbolic Computation*, 11:21–49, 1991.

[RV91] M. Rusinowitch and L. Vigneron. Automated deduction with associative commutative operators. In P. Jorrand and J. Kelemen, editors, *Fundamental of Artificial Intelligence Research*, volume 535 of *Lecture Notes in Computer Science*, pages 185–199. Springer-Verlag, 1991.

[RV93] M. Rusinowitch and L. Vigneron. Automated deduction with associative commutative operators. Internal report 1896, Institut National de Recherche en Informatique et Automatique, May 1993. To appear in *J. of Applicable Algebra in Engineering, Communication and Computation*.

[Sla74] J. R. Slagle. Automated theorem-proving for theories with simplifiers, commutativity and associativity. *Journal of the ACM*, 21(4):622–642, 1974.

[Vig94a] 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.

[Vig94b] L. Vigneron. Superposition in AC Theories: Proof of Completeness by Semantic Trees. Technical report 94-R-045, Centre de Recherche en Informatique de Nancy, Vandœuvre-lès-Nancy, 1994.

[Wer92] U. Wertz. First-order theorem proving modulo equations. Technical Report MPI-I-92-216, Max Planck Institut für Informatik, April 1992.

[ZK88] H. Zhang and D. Kapur. First-order theorem proving using conditional rewrite rules. In E. Lusk and R. Overbeek, editors, *Proceedings 9th International Conference on Automated Deduction, Argonne (Ill., USA)*, volume 310 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 1988.

A Proof of Refutational Completeness

In this appendix, we first introduce some important notions of our proof technique, like the definitions of the maximal consistent semantic tree and the quasi-rightmost path. We detail the proof of completeness for the AC-superposition strategy, showing that most of the subproofs are identical to the standard ones in unconstrained AC-deduction [RV93, Vig94b]. The completeness proof of the paramodulation strategy is simpler than the next proof for superposition, but we detail the additional cases of blocking right-hand sides.

A.1 Proof Technique

In this paper, we do not present the basic notions on transfinite semantic trees and interpretations (they can be found in [HR91]), but we introduce some new notions used in our proof.

The Herbrand base $\mathcal{A}(\mathcal{P},\mathcal{F})$ (theset of ground atoms) is ordered as an increasing sequence $\{A_i\}_{i<\lambda}$ by a complete simplification ordering which is AC-compatible, i.e. given two AC-equal atoms A_α and A_γ, each atom A_β between A_α and A_γ has to be AC-equal to A_α. Given an atom A_α, we write W_α the initial segment $\{A_i \mid i<\alpha\}$. Given the ordinal α, an *interpretation* on W_α is a mapping I (sometimes written I_α) from W_α to $\{T,F\}$ satisfying :

- if $(s \simeq s) \in W_\alpha$, then $I(s \simeq s) = T$
- if $(s \simeq t),\ B[s],\ B[t] \in W_\alpha$ and $I(s\simeq t) = T$, then $I(B[s]) = I(B[t])$

By the next theorem, it is possible to build inductively all the interpretations. An atom is reducible if it can be rewritten by a smaller equational atom satisfied by the interpretation.

Theorem 18 *[HR91] Let* $I : W_{\alpha+1} \rightarrow \{T,F\}$ *be such that I is an interpretation on* W_α*. Let J be the restriction of I to* W_α*. Then, I is an interpretation on* $W_{\alpha+1}$ *iff :*

(1) A_α *is reducible to an atom B and* $I(A_\alpha) \simeq I(B)$ *, or*
(2) A_α *is irreducible, of the form* $(t\simeq t)$ *, and* $I(A_\alpha)\simeq T$ *, or*
(3) A_α *is irreducible and not of the form* $(t \simeq t)$ *.*

Let I and J be two interpretations defined as in the previous theorem. We say that *I is an extension of J*. The collection of all interpretations is called a *transfinite semantic tree*. A *node* is an element of the tree, and the right node of an irreducible atom A_α is the interpretation R such that $R(A_\alpha) = F$.

We define an interpretation I_α on W_α to be *AC-inconsistent* if there are two AC-equal atoms A and B in W_α which are differently interpreted by I_α, i.e. $I_\alpha(A) = T$ and $I_\alpha(B) = F$.

Definition 19 *Let T be a transfinite semantic tree. The* maximal consistent semantic tree *of a set of constrained clauses S, written MCT(S), is the maximal subtree of T such that :*
For every node I_α *in MCT(S), and every clause C AC-equal to a ground instance* $C'\sigma$ *of a constrained clause* $C'\ [\![T]\!]$ *in S, where* σ *is a ground solution of T,* $I_\alpha(C) \simeq T$ *if*

1. *either C is an AC axiom,*
2. *or, for each variable x in an atom A of* C'*,*
   (a) *if x is an extension variable* x_f*, for each term u, AC-equal to a term of the set* $\mathcal{H}terms(x_f\sigma, f)$*, u is irreducible by any equality* $(l \simeq r)$ *satisfied by* I_α *and smaller than* $A\sigma$ $(l \succ r)$*,*
   (b) *if x is not an extension variable* x_f*, for each term u AC-equal to* $x\sigma$*, u is irreducible by any equality* $(l\simeq r)$ *satisfied by* I_α *and smaller than* $A\sigma$ $(l \succ r)$*.*

We note that the condition 2 can be interpreted as the irreducibility of the substitution σ.
Then, a *failure node* (*FN* for short) is an interpretation I_α falsifying a clause C, AC-equal to a ground instance of a constrained clause $C'\ [\![T]\!]$ of S, where C satisfies one of the previous conditions 1 and 2.

Let us define AC by the set of all the AC axioms. The following lemma states that there is no path in *MCT(*$S \cup AC$*)* which is defined on the entire Herbrand base, provided that S is AC-unsatisfiable.

Lemma 20 *Let S be a set of clauses ; then, S is AC-unsatisfiable if and only if every maximal path in MCT(*$S\cup AC$*) extends to a failure node.*

Adding AC to S is necessary in order to introduce failure nodes when an interpretation falsifies an equation $(u \simeq u')$ where $u =_{AC} u'$. Such an interpretation may be consistent but not AC-consistent and it should be discarded from the set of potential models of S.

To prove the Theorem 17, we reason by contradiction. We assume that S is an AC-unsatisfiable set of clauses and that S^* does not contain the empty clause. Therefore $MCT(S^* \cup AC)$ is not empty. Let us define the sets

$$\begin{aligned} \mathcal{GS} &= \{\, C' \llbracket \sigma \rrbracket \mid \exists C' \llbracket T \rrbracket \in S^*,\ \sigma \in \mathcal{S}ol(T) \,\} \\ \mathcal{AC} &= \{\, (u \simeq u') \llbracket \rrbracket \mid u, u' \in \mathcal{T}(\mathcal{F}),\ u =_{AC} u',\ top(u) \in \mathcal{F}_{AC} \,\} \\ \mathcal{S}^* &= \mathcal{GS} \cup \mathcal{AC} \end{aligned}$$

$\mathcal{GS}$ is also the set of ground instances of clauses of S^*, and $\mathcal{AC}$ is the set of ground instances of AC axioms. By definition of the maximal consistent semantic tree, $MCT(S^*)$ is equivalent to $MCT(S^* \cup AC)$. Hence, $MCT(\mathcal{S}^*)$ is non-empty too. To contradict this non-emptiness of $MCT(\mathcal{S}^*)$, we will choose an AC-consistent path in this subtree and prove that it is empty.

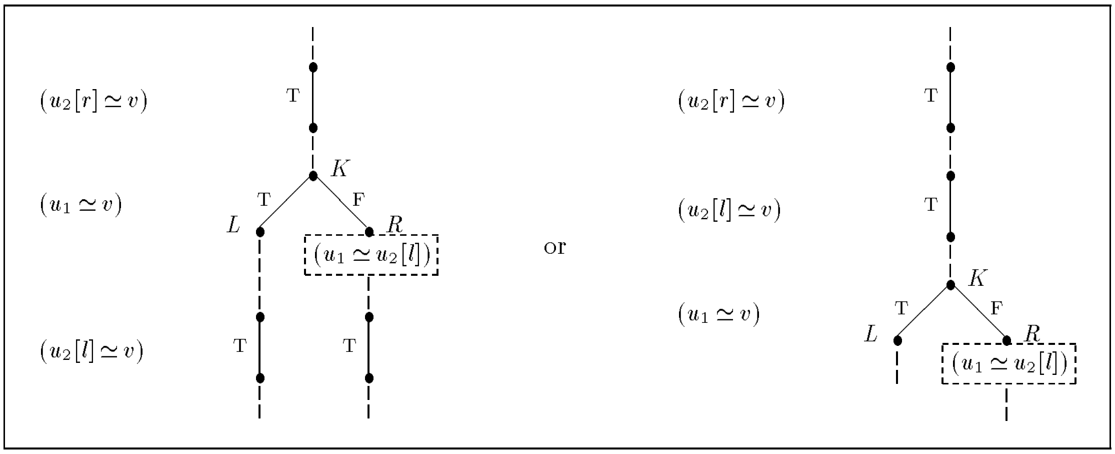


Figure 1: Quasi-Failure Node

But first, we shall need to define an extension of the notion of failure node, quasi-failure node, in order to detect an AC-inconsistency as soon as it occurs on an interpretation. Since an AC-inconsistency may be well-hidden, the notion of quasi-failure node is rather tricky ; it is illustrated by the Figure 1.

Definition 21 (Quasi-Failure Node) *Let K be a node of $MCT(\mathcal{S}^*)$, defined on W_α, such that A_α is an atom $(u_1 \simeq v)$, with $u_1 \succ v$; K admits two extensions L and R : $L(A_\alpha) \simeq T$, $R(A_\alpha) \simeq F$. Then, R is a quasi-failure node (QFN for short) if there is a term $u_2[l]$, AC-equal to u_1, which is reducible by an atom $(l \simeq r)$ to $u_2[r]$, with $l \succ r$ and $K(u_2[r] \simeq v) \simeq T$. The label of a quasi-failure node is defined to be $(u_1 \simeq u_2[l])$.*

With the superposition strategy, it is forbidden to reduce right-hand sides of equations. So, we have to define a special failure node, called distant failure node and illustrated by the Figure 2, to avoid situations where only the right-hand side of a positive equation is reducible.

Definition 22 (Distant Failure Node) *Let K be a node of $MCT(\mathcal{S}^*)$, defined on W_α, such that A_α is an irreducible equational atom $(u \simeq v)$, with $u \succ v$; K admits two extensions L and R : $L(A_\alpha) = T$, $R(A_\alpha) = F$, and $R \in MCT(\mathcal{S}^*)$. Then, R is a distant failure node (DFN for short) at the level of an atom $(u \simeq w)$ and labeled by a clause $(u \simeq w) \vee D$, if R is not a quasi-failure node, and R is the root of a subtree of $MCT(\mathcal{S}^*)$ whose rightmost path ends with a failure node M_R, labeled by a clause $(u \simeq w) \vee D$ and such that : $w \succ v$, $K(w \simeq v) = T$, and $\forall v' < w$, $M_R(u \simeq v') = F$.*

We define now the quasi-rightmost path of $MCT(\mathcal{S}^*)$ as its rightmost path which does not contain quasi-failure nodes or distant failure nodes.

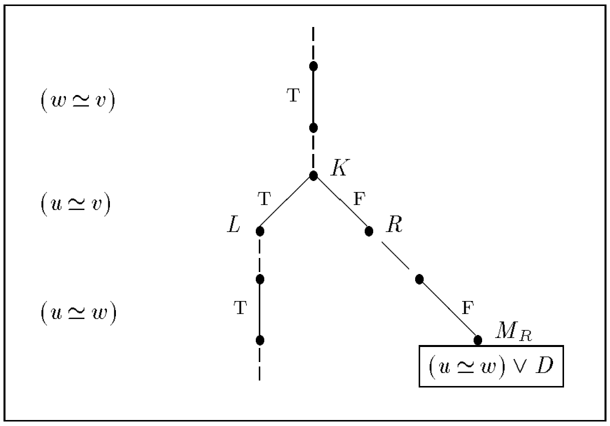


Figure 2: Distant Failure Node

Definition 23 (Quasi-Rightmost Path) *The quasi-rightmost path (QRP shortly) of $MCT(\mathcal{S}^*)$ is the interpretation Q_γ, defined on W_γ, and built as follows : Q_0 is the empty interpretation ; we assume that Q_i has been defined for all $i < \alpha$; then, we extend the path to Q_α, if possible, by :*

- *If α is a limit ordinal, as in the classical case [HR91], we simply define Q_α by $\bigcup_{i<\alpha} Q_i$. Then, Q_α belongs to $MCT(\mathcal{S}^*)$,*
- *If α is not a limit ordinal, it has a predecessor α^-. Several cases may occur :*
  - *If Q_{α^-} has exactly one successor J, then either J is in $MCT(\mathcal{S}^*)$ and Q_α is J, or J is not in $MCT(\mathcal{S}^*)$ and γ is α^-,*
  - *If Q_{α^-} has exactly two successors L and R, $L(A_{\alpha^-}) = T$ and $R(A_{\alpha^-}) = F$, then*
    * *If R is a QFN, a FN or a DFN, then either L is a FN and γ is α^-, or L is not a FN and Q_α is L,*
    * *If R is neither a QFN, nor a FN, nor a DFN, Q_α is R.*

A.2 Proof of Completeness

To prove that the quasi-rightmost path Q_γ is empty, we first need to show that the failure nodes and distant failure nodes along the QRP are labeled by persistent clauses, i.e. clauses never strictly subsumed or simplified, and that the QRP is non-empty and AC-consistent by construction.

Proposition 24 *Let K be a failure node, right extension of a node Q_α in the QRP or extension of Q_γ. If the QRP is AC-consistent, K falsifies a persistent clause.*

The proof of this proposition is exactly the same as the standard proof in unconstrained AC-deduction, which is detailed in [RV93, Vig94b]. The only problem which could be set relies on the definition of failure nodes, and more precisely on the irreducibility of substitutions. But, our simplification rules impose that, whenever a clause $C_2 \llbracket \sigma_2 \rrbracket$ is deleted thanks to a clause $C_1 \llbracket \sigma_1 \rrbracket$, either all the terms of the range of σ_1 are in the range of σ_2, or the clause $C_1 \llbracket \sigma_1 \rrbracket$ is transformed into a clause $C_1\sigma_1' \llbracket \sigma_1'' \rrbracket$ (with $\sigma_1'\sigma_1'' = \sigma_1$) so that this property is satisfied between σ_1'' and σ_2.

A consequence of this proposition is that, failure nodes being labeled by persistent clauses, inference rules between such clauses are always possible, and the fairness of the derivation ensures that the deduced clauses are subsumed by persistent clauses ; so, if a clause C, deduced by an inference step, is falsified by an interpretation, this interpretation falsifies a persistent clause subsuming C.

Proposition 25 *If $MCT(\mathcal{S}^*)$ is non-empty, the quasi-rightmost path Q_γ of $MCT(\mathcal{S}^*)$ is non-empty and AC-consistent.*

Proof : *First, Q_γ is non-empty because the interpretation Q_1, defined on W_1, belongs to $MCT(\mathcal{S}^*)$ and, since it has only one atom in its domain, it cannot be a QFN or a DFN.*

Second, we have to prove that Q_γ is AC-consistent. We reason by contradiction : if Q_γ is AC-inconsistent, there exists a minimal ordinal α such that Q_α is AC-inconsistent. Necessarily, α is not a limit ordinal : for a limit ordinal α, Q_α is defined by $\bigcup_{i<\alpha} Q_i$, and then some Q_i should be AC-inconsistent ; this is impossible since α is minimal. α has also a predecessor α^-. Let K be the interpretation Q_{α^-}, and let B denote the atom A_{α^-}. By minimality of α, K is AC-consistent. Then,

$$\exists A_\beta < B \ , \ A_\beta =_{AC} B \ \text{ and } \ Q_\alpha(B) \neq Q_\alpha(A_\beta)$$

We easily prove that B is an equality $(u_2 \simeq v)$ and A_β is an equality $(u_1 \simeq v)$, where $u_1 =_{AC} u_2$ and $u_1 \succ v$. Assuming that $A_\beta = (u_1 \simeq v)$ is the smallest atom such that $A_\beta =_{AC} B$ and $Q_\alpha(A_\beta) \neq Q_\alpha(B)$, several cases are possible :

- *If K has exactly one successor, Q_α : B is reducible and A_β is irreducible. Moreover, $K(A_\beta) = F$; it implies that a QFN should occur at the level of A_β.*
- *If K has two successors L and R, at least one of them (Q_α) is not a FN, a DFN or a QFN, $L(B) \simeq T$ and $R(B) \simeq F$,*
  - *If Q_α is L : $K(A_\beta) = F$ and R is a QFN, i.e. there is an atom B', AC-equal to B, which is reducible by an irreducible equality $(g \simeq d)$ whose right node is a FN or a DFN ; moreover, A_β is reducible by an equality $(l \simeq r)$ whose right node is a FN or a DFN. From this, we can deduce that Q_α falsifies a clause of $\mathcal{GS}$, deduced by an extended superposition between clauses labeling the right nodes of $(g \simeq d)$ and $(l \simeq r)$. g and l being irreducible, the contexts of this extended superposition are irreducible too, since they are subterms of g and l. This is the reason why we did not used extension variables to represent them.*
  - *If Q_α is R : $K(A_\beta) = T$ and either A_β is reducible and also R should be a QFN, or A_β is irreducible and the right node of A_β is a FN, a DFN or a QFN ; so, R should be a FN, a DFN or a QFN too.*

All these cases are proved impossible in the same way as in [RV93], with some additional cases to consider distant failure nodes [Vig94b] ; also Q_α cannot belong to the QRP Q_γ, and this sequence of interpretations is AC-consistent. □

However, there is one case left to verify. When an extended superposition (or paramodulation) step is applied, the deduced clause is falsified by Q_α, but to ensure that this clause labels a failure node in the QRP, we have to check that the substitution representing its constraints is irreducible (condition 2 in the definition of *MCT*). This is proved in the next proposition.

Proposition 26 *Let Q_β be an AC-consistent interpretation, restriction of the QRP Q_γ. Let Q_{α_1} and Q_{α_2} be two restrictions of Q_β, admitting a failure node or distant failure node as right extension. If an extended superposition step is applied between clauses labeling these right extensions and if the deduced clause is falsified by Q_β, it labels a failure node cutting the QRP.*

Proof : *Let $C_1 = (l_1 \simeq r_1) \vee D_1$ and $C_2 = (l_2 \simeq r_2) \vee D_2$ be clauses labeling the right extensions of Q_{α_1} and Q_{α_2}. It means that the left-hand sides l_1 and l_2 are irreducible, and there are constrained clauses $C_1' = (l_1' \simeq r_1') \vee D_1' \llbracket T_1 \rrbracket$ and $C_2' = (l_2' \simeq r_2') \vee D_2' \llbracket T_2 \rrbracket$ in S^* such that each C_i is AC-equal to $C_i'\sigma_i$, where σ_i is a ground irreducible substitution (condition 2 in the definition of MCT) satisfying T_i. An extended superposition step is applied between the ground constrained clauses $C_1' \llbracket \sigma_1 \rrbracket$ and $C_2' \llbracket \sigma_2 \rrbracket$, and generates a new ground constrained clause $C' \llbracket \sigma \rrbracket$.*
If we assume that the interpretation Q_β falsifies a clause C, AC-equal to $C'\sigma$, the only condition to check that a FN (labeled by C) cuts the QRP is the irreducibility of the ground substitution σ.
For each variable x of the domain of σ :

1. *If x is in the domain of a σ_i, $x\sigma_i$ is equal to $x\sigma$ and also each term AC-equal to $x\sigma$ is irreducible. If x is an extension variable x_f for the AC-operator f in C_i', the condition of irreducibility of its $\mathcal{H}terms$ is always valid.*

2. *If the inference step was an extended paramodulation, x may be the variable x_{r_1} or x_{r_2} introduced by the inference step (see Definition 7) : it appears in the constrained part by $x =_{AC} r'_1\sigma_1$; the next reasoning would be similar if it was $x =_{AC} r'_2\sigma_2$.*
   So, $x\sigma$ is the term $r'_1\sigma$, AC-equal to r_1. We have to prove that there is no term r''_1 AC-equal to r_1 and reducible by an equality $(g \simeq d)$ (with $g \succ d$).
   If such a term exists, the equality $(l_1 \simeq r''_1)$ is reducible by $(g \simeq d)$ to $(l_1 \simeq r''_1[d])$. Since Q_β is AC-consistent, $(l_1 \simeq r_1)$, $(l_1 \simeq r''_1)$ and also $(l_1 \simeq r''_1[d])$ are valid for Q_β. But, $(l_1 \simeq r''_1[d]) \succ (l_1 \simeq r_1)$ and therefore $(l_1 \simeq r_1)$ has to be reducible to $(r''_1[d] \simeq r_1)$; it contradicts the irreducibility of $(l_1 \simeq r_1)$.
   So, each term AC-equal to r_1 is irreducible.

3. *Else, x is one of the variables z_1 and z_2 (see Definition 7). If x is z_1, by definition of the extended superposition, $l_2 =_{AC} f(u, z_1\sigma)$, where f is the AC-operator at the head of l_2, and u is a ground term. If there is a term t, AC-equal to $z_1\sigma$, which is reducible by an equality $(g \simeq d)$ $(g \succ d)$, the term $f(u,t)$ (AC-equal to l_2) is reducible by $(g \simeq d)$ too. But, it means that the right extension of Q_{α_2} has to be a QFN instead of a FN or a DFN. This is in contradiction with the hypotheses.*
   The reasoning is similar if x is z_2.

So, a failure node labeled by C cuts the QRP above the node Q_β. □

Now that we have proved that the quasi-rightmost path Q_γ is non-empty and AC-consistent by construction, we show in the following that a contradiction yields when examining its last node.

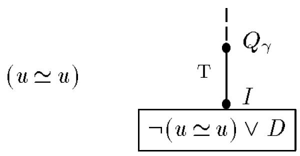
Proposition 27 *The quasi-rightmost path should be empty.*

Proof : *The last node Q_γ of the QRP is defined on W_γ ; it belongs to $MCT(\mathcal{S}^*)$, and its extensions are FNs, DFNs or QFNs. Let us denote the atom A_γ by B. As in the empty theory [HR91], there are three main cases :*

- *either Q_γ has one extension I, and B is irreducible,*
- *or Q_γ has two extensions L and R,*
- *or Q_γ has one extension I, and B is reducible.*

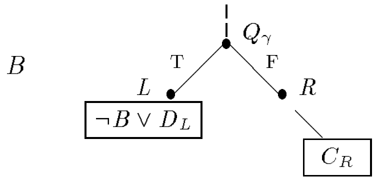
In the following, by abuse of notation, we say that a clause C labeling a failure node belongs to $\mathcal{GS}$ if there is a clause C' $[\![\sigma]\!]$ in $\mathcal{GS}$ such that $C'\sigma =_{AC} C$. An inference with C is an inference with C' $[\![\sigma]\!]$.

<u>**Case 1 :**</u> *Q_γ has one extension I, and B is irreducible*



A reflexion applied to the clause $\neg(u \simeq u) \vee D$ of $\mathcal{GS}$ produces a clause falsified by Q_γ.

<u>**Case 2 :**</u> *Q_γ has two extensions L and R (thus B is irreducible)*

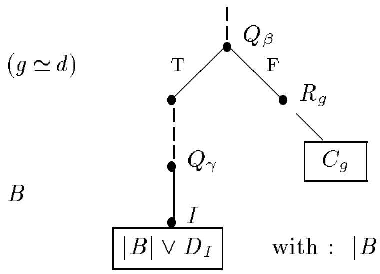


- *If R is a FN, C_R belongs either to $\mathcal{GS}$ and a resolution is applied between clauses labeling L and R, or C_R belongs to $\mathcal{AC}$ and a reflexion is applied to the clause labeling L.*
- *If R is a QFN, there is an atom B', AC-equal to B, which is reducible by an equation $(l \simeq r)$ with $l \succ r$, and whose right node is a FN or a DFN. So, we apply a (contextual) superposition from the clause labeling this right node into the clause labeling L.*

- *If R is a DFN, let M_R be the failure node under R, labeled by C_R. We apply a (contextual) superposition from C_R into the clause labeling L.*

In these three cases, we deduce a clause of $\mathcal{GS}$, falsified by Q_γ for the two first cases, and by M_R for the third one. So, it contradicts the construction of Q_γ.

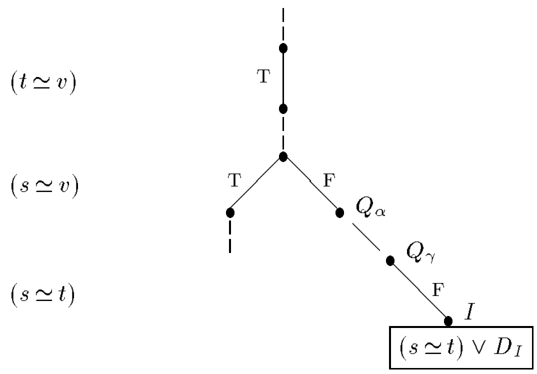
<u>**Case 3 :**</u> *Q_γ has one extension I, and B is reducible.*



with : $|B| = B$ if $I(B) = F$, $\neg B$ otherwise

There is an equation $(g \simeq d)$ reducing an atom AC-equal to B, whose right node is a FN or a DFN.

- *If $I(B) = T$ or B is not an equational atom : $|B|$ is the maximal literal of the clause $|B| \vee D_I$. So, a (contextual) superposition from the clause labeling R_g into $|B| \vee D_I$ produces a clause falsified by Q_γ.*
- *If $I(B) = F$, $B = (s \simeq t)$ and s is reducible : the literal B may not be maximal in the clause $B \vee D_I$. In this case, there is a literal $\neg(s' \simeq u)$ in D_I where $s' =_{AC} s$. So, a (contextual) superposition from the clause labeling R_g into $B \vee D_I$ in s' produces a clause smaller than $B \vee D_I$ and falsified by I.*
  Assuming that the clause labeling I is the smallest clause of $\mathcal{GS}$ falsified by I, B is maximal in $B \vee D_I$, and a (contextual) superposition from the clause labeling R_g into $(s \simeq t) \vee D_I$ (at a position in s). The deduced clause is falsified by Q_γ.
- *If $I(B) = F$, $B = (s \simeq t)$ and s is irreducible :*



  The literal B is maximal in $B \vee D_I$, and we can prove that Q_γ should not belong to the quasi-rightmost path because a node Q_α of the QRP should be a distant failure node labeled by $B \vee D_I$, as shown in the figure.

All cases for Q_γ have yielded a contradiction with the construction of the quasi-rightmost path. So, this path has to be empty. □

The last property to be check is that, when a clause is deduced by an inference step between clauses labeling failure nodes or distant failure nodes along the QRP, and if it is falsified by Q_γ, it labels a failure node in the QRP. Indeed, like for the extended superposition, we can imagine that the ground substitution representing its constraints may be reducible. We show in the following that it cannot happen.

Proposition 28 *Let us assume that the QRP Q_γ is AC-consistent. If an inference step is applied between clauses labeling failure nodes or distant failure nodes, extensions of a node in this QRP, and if the deduced clause is falsified by Q_γ, it labels a failure node cutting the QRP.*

Proof : *Let $C_1, \ldots, C_n$ be clauses labeling failure nodes or distant failure nodes. There are constrained clauses $C'_1 [\![\sigma_n]\!], \ldots, C'_n [\![\sigma_n]\!]$ in $\mathcal{GS}$ such that each C_i is AC-equal to $C'_i\sigma_i$, and the σ_i are ground irreducible substitutions (condition 2 in the definition of MCT). The inference step, applied between these ground constrained clauses, generates a new ground constrained clause $C' [\![\sigma]\!]$. If we assume that the interpretation Q_γ falsifies a clause C, AC-equal to $C'\sigma$, the only condition to check that a failure node (labeled by C) occurs in the QRP is the irreducibility of the ground substitution σ.*

Let us assume that the σ_i have disjoint domains of variables (otherwise we rename variables). Then, for each variable x in the domain of σ :

1. *If x is in the domain of a σ_i, $x\sigma_i$ is equal to $x\sigma$ since σ_i is ground, and also each term AC-equal to $x\sigma$ is irreducible. If x is an extension variable x_f, since C_i labels a failure node, the $\mathcal{H}terms$ of $x_f\sigma_i$ have to be irreducible.*

2. *If x is the extension variable x_f introduced by a contextual superposition step, and if a term t of $\mathcal{H}terms(x_f\sigma, f)$ is reducible by an equation $(g \simeq d)$ $(g \succ d)$, another (contextual) superposition from the clause labeling the right node of $(g \simeq d)$ into C_2 generates a clause labeling a failure node in the QRP. This reasoning is well-founded since we go at a deeper position in the clause.*

3. *Else, x is a new variable introduced by the inference step. But, only three inference rules introduce new variables : the three paramodulation rules. Let us study two of them : paramodulation and contextual paramodulation. The third one was treated in the Proposition 26.*
   In both cases, denoting C'_1 by $(l'_1 \simeq r'_1) \vee D'_1$ and C_1 by $(l_1 \simeq r_1) \vee D_1$, x is described by the constraint $x =_{AC} r'_1$, produced by the inference step ; so, $x\sigma$ is equal to $r'_1\sigma_1$ (and AC-equal to r_1). We have to prove that there is no term r''_1, AC-equal to r_1, that is reducible by an equality $(g = d)$ (with $g \succ d$). It is exactly the same reasoning as the case 2 of Proposition 26.

Hence, all terms associated to a variable by the substitution σ are irreducible, and a failure node, labeled by C, occurs in the QRP. □

To summarize, we know that, if the maximal consistent semantic tree is non-empty, by construction the quasi-rightmost path is non-empty too and AC-consistent. But, we have just proved that this QRP was empty. The only solution is that $MCT(\mathcal{S}^*)$ is empty and therefore the empty clause belongs to S^*.