Termination and Completion modulo Associativity, Commutativity and Identity

Jean-Pierre Jouannaud Claude Marché
Laboratoire de Recherche en Informatique
Bat. 490
Université Paris-Sud et CNRS UA 410
91405 ORSAY CEDEX
FRANCE

Abstract

Rewriting with associativity, commutativity and identity has been an open problem for a long time. In 1989, Baird, Peterson and Wilkerson introduced the notion of constrained rewriting, to avoid the problem of non-termination inherent to the use of identities. We build up on this idea in two ways: by giving a complete set of rules for completion modulo these axioms; by showing how to build appropriate orderings for proving termination of constrained rewriting modulo associativity, commutativity and identity.

1 Introduction

Equations are ubiquitous in mathematics and the sciences. Among the most common equations are associativity, commutativity and identity (existence of a neutral element). Rewriting is an efficient way of reasoning with equations, introduced by Knuth and Bendix [12]. When rewriting, equations are used in one direction chosen once and for all. Unfortunately, orientation alone is not a complete inference rule: given a set of equational axioms E, there may be equal terms (in the theory of the axioms) which cannot be rewritten to a same term once the axioms are oriented into rules. Knuth and Bendix showed how to recover completeness by adding another inference rule, called "critical pairs computation", which adds new equational consequences by unifying left-hand sides of rules. A basic assumption of this technique is that rewriting terminates for any input term. In case of associativity and commutativity (hereafter denoted by AC) however, this assumption cannot be fullfilled. Peterson and Stickel (see also Lankford and Ballantyne [13]) have shown how to solve this case by building associativity and commutativity in the rewriting process, as well as in the computation of critical pairs [15]. This has been further generalized to an arbitrary finitary theory E (i.e. any equation in the theory possesses a finite complete set of most general unifiers) whose congruence classes are finite [10], excluding therefore the case of associativity, commutativity and identity (hereafter denoted by AC1). The latter theory is of keen interest for two reasons: the identity axiom comes along with associativity in most practical examples; for a given equation, associative-commutative unification yields in general many more most general solutions than AC1-unification, as shown by Bürkert [7].

Unfortunately, AC1-rewriting is non-terminating in most practical cases. For example, in the theory of Abelian groups, we get the following infinite derivation (using the rule $-(x+y) \to (-x)+(-y)$ for computing the inverse of a sum):

$$\begin{array}{lll} -0 & =_{\mathrm{AC1}} & -(0+0) \to \quad (-0)+(-0) \\ & =_{\mathrm{AC1}} & -(0+0)+(-0) \to \quad (-0)+(-0)+(-0) \\ & etc\ldots & \end{array}$$

Since this phenomenon is simply related to the existence of a subterm $x+y$ in the left-hand side of a rule, which collapses when either x or y is instantiated to 0, it may happen quite often. A consequence is that AC1-rewrite orderings cannot really exist. Peterson, Wilkerson and Baird introduced "constrained" rewriting as a means to avoid non-terminating computations (see also [11]). The basic idea is to forbid instantiating a variable by an identity in case this may lead to non-termination. We show in section 2 how constrained rewriting allows to solve the problem of non-termination. In particular, we show how to obtain an ordering for proving termination of constrained AC1-rewriting from an almost arbitrary AC-rewrite ordering. Section 3 then describes a class of AC1-completion procedures by a set of non-deterministic inference rules, together with its correctness and completeness proof. The latter is based on the technique of "proof algebras", originally introduced in [2], whose principle is to code the inference rules as rewrite rules on proofs. The completeness proof of AC1-completion, however, is not a trivial application of this technique, since one inference rule may yield several rewrite rules on proofs. This makes it doubtful that such a proof could be carried out without this powerful technique.

Our notations and definitions are consistent with [9]. We assume them known.

2 Constrained AC1-rewriting

We start with a few elementary definitions.

Definition 2.1 *Let $\mathcal{F} = \mathcal{L} \cup \mathcal{A} \cup \mathcal{Z}$ be a finite set of function symbols, where $\mathcal{L}$ is the set of* free symbols, *$\mathcal{A}$ is the set of* associative-commutative symbols, *and $\mathcal{Z}$ is the set of* identities. *Each $0 \in \mathcal{Z}$ is the identity of $a + \in \mathcal{A}$. This correspondance is given by a one-to-one mapping $zero : \mathcal{A} \to \mathcal{Z}$. We use $+$ and $*$ to denote two particular AC1-symbols with respective identities 0 and 1.*

An $\mathcal{F}$-algebra is an AC1-algebra if it validates the set of AC1 equations:

$$\forall + \in \mathcal{A}, 0 = zero(+) \left\{ \begin{array}{rcl} (x+y)+z & = & x+(y+z) \\ x+y & = & y+x \\ x+0 & = & x \end{array} \right.$$

Definition 2.2 *In the following, we denote by $s\downarrow$ the normal form of s for the following canonical rewriting system:*

$$\forall + \in \mathcal{A}, 0 = zero(+) \left\{ \begin{array}{l} x+0 \to x \\ 0+x \to x \end{array} \right.$$

Definition 2.3 *Two terms s and t are* unifiable modulo AC1 *if there exists a substitution σ such that $s\sigma =_{AC1} t\sigma$. We denote $CSU_{AC1}(s,t)$ a* complete set of AC1-unifiers *(or a set of* most general AC1-unifiers*) of s and t, i.e.:*

$$\left\{ \begin{array}{l} \forall \theta \in CSU_{AC1}(s,t)\ s\theta =_{AC1} t\theta \\ \forall \sigma\ s\sigma =_{AC1} t\sigma \Longrightarrow \exists \theta \in CSU_{AC1}(s,t) \exists \sigma'\ \sigma =_{\mathrm{AC1}} \theta\sigma' \end{array} \right.$$

The definition of unification is modified to obtain the definition of *matching* by requiring that one term only, say s, can be instantiated by σ.

For any two terms s and t, $CSU_{AC1}(s,t)$ is finite as a consequence of the combination method described in [6] and the elementary case (with $\mathcal{F}$ reduced to a set of constant symbols) described in [14].

The interesting property of AC1-unification is that in many cases, $CSU_{AC1}(s,t)$ is much smaller than $CSU_{AC}(s,t)$: for example, if x, u, v, w, t are variables, then $x+x+x$ and $u+v+w+t$ have a unique most general AC1-unifier while there are 1044569 most general AC-unifiers [7]. Matching enjoys similar properties. We think that this property would make the AC1-completion faster than AC-completion. In [4], examples are given which confirm this hope.

2.1 Constrained rewriting

The idea of Baird, Peterson and Wilkerson is to forbid the instantiations of variables by zeros, when they cause termination problems. For example, the computation of the inverse of a sum in Abelian groups becomes:

$$(x \neq 0) \wedge (y \neq 0) \mid -(x+y) \rightarrow (-x) + (-y)$$

Definition 2.4 *A* constrained rule *(for our purpose) is a triple written out as $\Phi \mid l \rightarrow r$, where Φ is a quantifier-free formula built up with the logical connectives $\wedge$ and $\vee$, from atoms of the form $x \neq 0$ where $x \in \mathcal{V}ar(l)$ and $0 \in \mathcal{Z}$.*

Definition 2.5 (satisfiability of a constraint) *By $\Phi\sigma$ satisfiable we mean that the formula $\Phi\sigma$ holds when the symbol $=$ is interpreted as $=_{\mathrm{AC1}}$*

For instance, $x \mapsto 0+0$ does not satisfy $x \neq 0$. Notice that satisfiability of a constraint can be easily checked: since $x\sigma \neq_{\mathrm{AC1}} 0$ is equivalent to $(x\sigma)\!\downarrow \neq 0$ syntactically, $\Phi\sigma$ satisfiable if and only if $\Phi(\sigma\!\downarrow)$ holds when $=$ is interpreted as syntactic equality.

Constrained rewriting checks whether an instance of the rule satisfies the constraint. As usual, we have three different notions of rewriting, without equations, in the quotient algebra, or with extensions "à la Peterson-Stickel" [15]. All of them are used in the completion process. We do not recall the corresponding notions for the AC-case (without constraints), which can be readily obtained from the definitions below by taking the constraints out and replacing $AC1$ by AC.

Definition 2.6 *s rewrites to t by $\Phi \mid l \rightarrow r$ at position p (written out as $s \xrightarrow[\Phi|l\rightarrow r]{p} t$) if:*

$$p \in \mathcal{FP}os(s) \text{ and } \exists\sigma \text{ substitution s.t. } \begin{cases} s|_p = l\sigma \\ t = s[r\sigma]_p \\ \Phi\sigma \text{ satisfiable} \end{cases}$$

Written out as $s \xrightarrow[\Phi|l\rightarrow r/AC1]{} t$, constrained rewriting modulo AC1 *must satisfy:*

$$\exists s', t', p \text{ and } \sigma \text{ s.t.} \begin{cases} s =_{\mathrm{AC1}} s' \\ t =_{\mathrm{AC1}} t' \\ s' \xrightarrow[\Phi|l\rightarrow r]{p} t' \end{cases}$$

Written out as $s \xrightarrow[AC1\backslash\Phi|l\rightarrow r]{p} t$, constrained extended AC1-rewriting must satisfy:

$$p \in \mathcal{FP}os(s) \text{ and } \exists\sigma \text{ substitution s.t. } \begin{cases} s|_p =_{AC1} l\sigma \\ t = s[r\sigma]_p \\ \Phi\sigma \text{ satisfiable} \end{cases}$$

Note that /AC1-rewriting actually rewrites AC1-equivalence classes, while AC1\-rewriting uses AC1-matching, hence applies a restricted form of $=_{\mathrm{AC1}}$ between s and $s[l\sigma]_p$.

2.2 AC1-rewrite orderings

Given a set R of rules, the non-deterministic computation of a normal form of a term requires that there are no infinite chains in the rewrite graph, a property called *termination.* Proving termination can be simply achieved by comparing for each rule $l \rightarrow r \in R$ its left-hand side l with its right-hand side r in some appropriate rewrite ordering:

Definition 2.7 *Given a set E of equational axioms, an E-*rewrite ordering *is a quasi-ordering $\geq_E$ satisfying the following properties ($>_E$ is the associated* strict *ordering, that is $s >_E t \iff s \geq_E t \wedge t \not\geq_E s$):*

- E-compatibility*: its associated equivalence is a congruence containing $=_E$, that is $s =_E t \Longrightarrow s \geq_E t$ and $t \geq_E s$*
- termination*: there is no infinite descending sequence $s_1 >_E s_2 >_E s_3 >_E \ldots$*
- monotonicity*: $s >_E t \Longrightarrow u[s] >_E u[t]$ for any context $u[.]$*
- closure under instantiation*: $s >_E t \Longrightarrow s\sigma >_E t\sigma$ for any substitution σ.*

*Moreover, $\geq_E$ is called an E-*simplification ordering *in case it satisfies the* subterm property*, i.e., a term is greater than any of its subterms. In that case, termination is a consequence of the other properties [8].*

To prove AC1-termination of a rewriting system R, we need a rewrite ordering compatible with AC1. Assume that such an ordering $\geq_{\mathrm{AC1}}$ compares $s+x$ and s, that is $s+x \geq_{\mathrm{AC1}} s$. Then $(s+x)\sigma \geq_{\mathrm{AC1}} s\sigma$ for any substitution σ. Taking $\sigma = \{x \mapsto 0\}$ yields $s\sigma \geq_{\mathrm{AC1}} s\sigma$ since $(s+x)\sigma =_{\mathrm{AC1}} s\sigma$, a contradiction since $=_{\mathrm{AC1}}$ must be terminating. We show in the following that AC1-rewrite orderings need not be closed under arbitrary instantiations, but under those instantiations only allowed in constrained AC1-rewriting.

Let us consider the following constrained rule to avoid non-termination in the example of Abelian groups:

$$(x \neq 0) \wedge (y \neq 0) \mid -(x+y) \rightarrow (-x)+(-y)$$

One question is whether we can check termination of this constrained rule by showing that the left-hand side is greater than the right-hand side in some rewrite ordering. In other words, we want an ordering closed by all substitutions σ which satisfy the above constraint, but not necessarily by the other substitutions. Another question is how to compute the constraint of a rule when the ordering is given. The answer to this second question is simple, already suggested by Baird, Peterson and Wilkerson: for each substitution σ which assigns a identity to a variable, we must check whether the left-hand side is greater than the right-hand side in the ordering. If this is not the case, then the identity rule must have been applied to the instantiated left-hand side. This suggest that the ordering should satisfy the following property:

Definition 2.8 $\geq_{\mathrm{AC1}}$ *is closed under non-zero instantiation if*

$$\boxed{(P) \quad \left\{ \begin{array}{l} s \geq_{\mathrm{AC1}} t \\ (s\!\downarrow)(\sigma\!\downarrow) = (s\sigma)\!\downarrow \end{array} \right. \Longrightarrow s\sigma \geq_{\mathrm{AC1}} t\sigma}$$

Lemma 2.1 *Assume that $\geq_{\mathrm{AC1}}$ satisfies (P). Then $\forall s,t$, $\forall \sigma$ s.t. $x\sigma \neq_{\mathrm{AC1}} 0$ for every x and 0*

$$s \geq_{\mathrm{AC1}} t \Longrightarrow s\sigma \geq_{\mathrm{AC1}} t\sigma$$

Proof. by contradiction: $(s\!\downarrow)(\sigma\!\downarrow) \neq (s\sigma)\!\downarrow$ implies that σ instantiates some variable to zero, because $(s\!\downarrow)(\sigma\!\downarrow)$ contains a redex $x+0$ or $0+x$. □

We now assume known an ordering $\geq_{\mathrm{AC1}}$ satisfying (P). Given this ordering, our goal is to compute a constraint which eliminates the non-terminating chains from the graph of the rewrite relation. Given a finite set X of variables, let

$$S_{AC1}(X) = \{\rho = (x_1 \mapsto 0_1, \ldots, x_k \mapsto 0_k) \text{ s.t. } k \geq 0, x_1, \ldots, x_k \in X, 0_1, \ldots, 0_k \in \mathcal{Z}\}$$

be the finite set of all possible instantiations of some variables in X by an arbitrary identity. Our goal is to compute for all pairs (l,r) a formula $\Phi = \Phi(l,r)$ satisfying:

$$\boxed{(Q) \;\; \forall \rho \in S_{AC1}(\mathcal{V}ar(l)) \; \Phi\rho \text{ satisfiable} \Longrightarrow l\rho \geq_{\mathrm{AC1}} r\rho}$$

Φ is actually any condition on ρ such that $l\rho \geq_{\mathrm{AC1}} r\rho$. It can of course be computed by checking all possible cases of instantiation of a variable by an identity. Before seeing how to compute an optimized Φ, we show the adequacy of this approach.

Notations: Given a pair (l,r) of terms, let $C_{AC1}(l,r)$ be a logical formula satisfying (Q), and $\Theta_{AC1}(l,r) = \{\rho \in S_{AC1}(\mathcal{V}ar(l)) \text{ s.t. } C_{AC1}(l,r)\rho \text{ unsatisfiable}\}$. Note that $\Theta_{AC1}(l,r)$ is redundant as defined here. Only those substitutions which are maximal in the subsumption ordering are necessary. For example, if $C_{AC1}(l,r) = (x \neq 0) \wedge (y \neq 0)$ then the substitution $\{x \mapsto 0, y \mapsto 0\}$ is not necessary, hence we can take $\Theta_{AC1}(l,r) = \{\{x \mapsto 0\}, \{y \mapsto 0\}\}$.

The following theorem shows that the termination problem is solved provided an ordering satisfying (P) is given together with an algorithm computing a constraint satisfying (Q).

Theorem 2.1 *Let l and r be two terms and $\Phi = C_{AC1}(l,r)$. For an arbitrary substitution σ:*

$$\begin{cases} \text{if } \Phi\sigma \text{ satisfiable then } l\sigma \geq_{\mathrm{AC1}} r\sigma \\ \text{if } \Phi\sigma \text{ unsatisfiable then } \exists \rho \in \Theta_{AC1}(l,r), \exists \lambda \text{ s.t. } \sigma = \rho\lambda \end{cases}$$

Proof. write σ as $\rho\lambda$ where $\rho = \sigma|_X$, $X = \{x \text{ s.t. } x\sigma =_{\mathrm{AC1}} 0, 0 \in \mathcal{Z}\}$. □

We now give an algorithm for computing a formula satisfying (Q). Given an AC1-ordering $\geq_{\mathrm{AC1}}$ satisfying (P), let

$$\begin{array}{l} \varphi(l,r) = \psi(l\downarrow, r\downarrow) \\ \text{where } \psi(l,r) = \text{ if } \quad l \geq_{\mathrm{AC1}} r \\ \qquad\qquad\qquad \text{then} \quad \displaystyle\bigwedge_{\substack{(x,0)\in I(l) \\ \sigma=\{x\mapsto 0\}}} (x \neq 0 \vee \varphi(l\sigma, r\sigma)) \\ \qquad\qquad\qquad\qquad \text{where } I(l) = \{(x,0) \in \mathcal{V}ar(l) \times \mathcal{Z} \text{ s.t. } \exists p, s\ l|_p = x + s \text{ or } s + x\} \\ \qquad\qquad\qquad \text{else} \quad \text{False} \end{array}$$

Lemma 2.2 *Let $\Phi = \varphi(l,r)$. Then $\forall \rho \in S_{AC1}(\mathcal{V}ar(l))$, if $\Phi\rho$ is satisfiable then $l\rho \geq_{\mathrm{AC1}} r\rho$.*

Proof. First remark that we can assume that l and r are in normal forms: if the lemma is proved in that case, then it is true for arbitrary l and r since $\varphi(l,r) = \varphi(l\downarrow, r\downarrow)$ implies that if $\Phi\rho$ is satisfiable, $l\downarrow\rho \geq_{\mathrm{AC1}} r\downarrow\rho$ and by $l =_{\mathrm{AC1}} l\downarrow$, $r =_{\mathrm{AC1}} r\downarrow$ and AC1-compatibility of $\geq_{\mathrm{AC1}}$, $l\rho \geq_{\mathrm{AC1}} r\rho$. So let us assume that $l = l\downarrow$ and $r = r\downarrow$. We prove the lemma by induction on $n = |\mathcal{V}ar(l)|$:

- If $n = 0$ then $I(l) = \phi$, $S_{AC1}(\mathcal{V}ar(l)) = \{\mathrm{id}\}$, Φ is True if $l \geq_{\mathrm{AC1}} r$ and False otherwise, hence the result is trivial.

- If $|\mathcal{V}ar(l)| = n+1$ and the lemma is true until n: $\Phi\rho$ is satisfiable hence $l >_{\mathrm{AC}} 1r$ and $\Phi = \psi(l,r) = \displaystyle\bigwedge_{\substack{(x,0)\in I(l) \\ \sigma=(x\mapsto 0)}} (x \neq 0 \vee \varphi(l\sigma, r\sigma))$

  Hence since $\Phi\rho$ satisfiable we have $\forall (x,0) \in I(l)$, $\begin{cases} x\rho \neq 0 \\ \text{or} \\ \varphi(l\sigma, r\sigma)\rho \text{ satisfiable} \end{cases}$

  - If $\forall (x,0) \in I(l)$, $x\rho \neq 0$ then there are no redexes $0 + \ldots$ or $\ldots + 0$ in $l\rho$ hence $(l\rho)\downarrow = l\rho$. But l is in normal form and ρ too because $\rho \in S_{AC1}(\mathcal{V}ar(l))$ hence $(l\rho)\downarrow = (l\downarrow)(\rho\downarrow)$. But then $l \succ r \Longrightarrow l\rho \succ r\rho$ by (P).
  - If $\exists (x,0) \in I(l) \mid x\rho = 0$ then assume such a $(x,0)$ chosen and $\sigma = \{x \mapsto 0\}$. We have $\varphi(l\sigma, r\sigma)\rho$ satisfiable and $x\rho = 0$ hence $\rho = \{x \mapsto 0\}\rho' = \sigma\rho'$ where ρ' is defined by:

$$\begin{cases} x\rho' = x \\ y\rho' = y\rho \text{ if } y \neq x \end{cases}$$

But then $\rho' \in S_{AC1}(\mathcal{V}ar(l\sigma))$ because $\mathcal{V}ar(l\sigma) = \mathcal{V}ar(l)\backslash\{x\}$. Since $x \notin \mathcal{V}ar(\varphi(l\sigma, r\sigma))$ we have $\varphi(l\sigma, r\sigma)\rho = \varphi(l\sigma, r\sigma)\rho'$ satisfiable hence by induction hypothesis $(l\sigma)\rho' \succ (r\sigma)\rho'$ and since $\rho = \sigma\rho'$ we have $l\rho \succ r\rho$.

□

We are left with the problem of finding an ordering satisfying (P). We show now that any AC-simplification ordering can be made into an AC1-ordering satisfying (P). For a particular AC-simplification ordering, we may choose the associative path ordering [3], or an ordering defined by polynomial interpretations [5].

Definition 2.9 *Given an AC-simplification ordering $\geq_{\mathrm{AC}}$, let*

$$s \geq_{\mathrm{AC1}} t \textit{ if } s\!\downarrow \geq_{\mathrm{AC}} t\!\downarrow$$

Note that we have always $s \geq_{\mathrm{AC}} s\!\downarrow$ since applying an identity equation transforms a redex into one of its subterms.

Proposition 2.1 *$\geq_{\mathrm{AC1}}$ is an AC1-quasi-ordering satisfying (P), whose strict part is terminating. Furthermore, if all identities are minimal for $\geq_{\mathrm{AC}}$, then $\geq_{\mathrm{AC1}}$ is closed under context application.*

Proof. The monotonicity is based on the following technical lemma, which is a consequence of (P):

Lemma 2.3 *Let s be a term such that $\forall 0 \in \mathcal{Z}\ s \neq_{AC1} 0$. Then $(u[s]_p)\!\downarrow = (u\!\downarrow)[s\!\downarrow]_q$ where $q \in \mathcal{P}os(u\!\downarrow)$ depends upon u and p only (not on s).*

Proof. This lemma is in fact a consequence of (P): we can write $u[s]_p$ as $u[x]_p\sigma$ where $x \notin \mathcal{V}ar(u)$ and $\sigma = \{x \mapsto s\}$. Since s is not an identity, we have from lemma 2.1 $(u[s]_p)\!\downarrow = (u[x]_p\sigma)\!\downarrow = (u[x]_p\!\downarrow)(\sigma\!\downarrow) = (u\!\downarrow)[x]_q\sigma'$ where $\sigma' = \{x \mapsto s\!\downarrow\}$ and q, defined by $u[x]_p\!\downarrow = u\!\downarrow [x]_q$, does not depend on s. Hence $(u[s]_p)\!\downarrow = (u\!\downarrow)[s\!\downarrow]_q$. □

Using the lemma, since identities are minimal, if $s \geq_{\mathrm{AC1}} t$ then s is not an identity. Hence, $(u[s]_p)\!\downarrow = (u\!\downarrow)[s\!\downarrow]_q >_{\mathrm{AC}} (u\!\downarrow)[t\!\downarrow]_q$ since $>_{\mathrm{AC}}$ is monotonic and $s \geq_{\mathrm{AC1}} t \Longrightarrow s\!\downarrow >_{\mathrm{AC}} t\!\downarrow$. But $(u\!\downarrow)[t\!\downarrow]_q \geq_{\mathrm{AC}} ((u\!\downarrow)[t\!\downarrow]_q)\!\downarrow = (u[t]_q)\!\downarrow$, hence $u[s]_p \geq_{\mathrm{AC1}} u[t]_p$ by definition of the ordering. □

We conclude this section with two remarks:

- This result shows that (P) is the appropriate property replacing closure by instantiation. In the following, we assume that *AC1-rewrite orderings* satisfy (P) instead of the full closure by instantiation.
- The last hypothesis that identities are minimal is quite natural, since otherwise the AC1-ordering cannot be well-founded: $0 \geq_{\mathrm{AC1}} s =_{\mathrm{AC1}} s + 0 \geq_{\mathrm{AC1}} s + s =_{\mathrm{AC1}} s + s + 0$ etc...

3 AC1-completion

As now customary, we describe the completion process by a set of inference rules which mimics the inference rules for AC-completion. The main difference is the computation of constraints when a rule is added, and the use of the constraints when rules or equations are reduced. The problem of soundness of the rules is then addressed before dealing with the most difficult part, completeness.

3.1 Inference rules for AC1-completion

In the following, $\succ$ is an AC1-rewrite ordering, E is the set of equations, N is the set of non-protected rules, P is the set of protected rules (i.e. the extensions), and $R = N \cup P$.

Orient

$$E \cup \{l = r\}; N; P \vdash E \cup \{l\rho = r\rho \mid \rho \in \Theta_{AC1}(l, r)\}; N \cup \{\Phi \mid l \to r\}; P \text{ if } \begin{cases} l \succ r \\ \Phi = C_{AC1}(l, r) \end{cases}$$

Orient turns an equation into a rule, and computes its constraint. The "forbidden" instantiations of the equations are left as equations.

Extend

$$E; N; P \vdash E; N; P \cup \{\Psi \mid g\theta \to d\theta[r\theta]_q\} \text{ if } \begin{cases} \Phi \mid l \to r \in R \\ g = d \in \mathrm{AC1} \\ q \in \mathcal{FP}os(d) \text{ with } q \neq \Lambda \\ \theta \in CSU_{AC1}(d|_q, l) \\ \Psi = C_{AC1}(g\theta, d\theta[r\theta]_q) \end{cases}$$

Extend adds an extension as a protected rule, and computes the associated constraint. Actually, the same constraint could be kept in the AC1 case, but this form is more general.

Deduce

$$E; N; P \vdash E \cup \{r\theta = l\theta[d\theta]_q\}; N; P \text{ if } \begin{cases} \Phi \mid l \to r \in R \\ \Psi \mid g \to d \in R \\ q \in \mathcal{FP}os(l) \\ \theta \in CSU_{AC1}(l|_q, g) \\ \Phi\theta \text{ satisfiable} \\ \Psi\theta \text{ satisfiable} \end{cases}$$

Deduce adds $AC1$-*critical pairs* as new equations.

Delete

$$E \cup \{l = r\}; N; P \vdash E; N; P \text{ if } l =_{AC1} r$$

Delete removes trivial equations.

Compose rule

$$E; N \cup \{\Phi \mid l \to r\}; P \vdash E; N \cup \{\Phi' \mid l \to r'\} \cup \{\Phi\rho \mid l\rho \to r\rho \mid \rho \in \Theta_{AC1}(g\theta, d\theta) \text{ and } \Phi\rho \text{ satisfiable}\}; P$$

$$\text{if } \begin{cases} \Psi \mid g \to d \in R \\ r \xrightarrow[\Psi|g\to d/AC1]{\theta} r' \\ \Phi' = C_{AC1}(l, r') \end{cases}$$

Compose rule reduces the right-hand side of rules, and recomputes the constraint. Here, it is important (for efficiency) to compute the new constraint which can improve over the previous one (it may be empty for example).

Compose extension

$$E; N; P \cup \{\Phi \mid l \to r\} \vdash E; N; P \cup \{\Phi' \mid l \to r'\} \cup \{\Phi\rho \mid l\rho \to r\rho \mid \rho \in \Theta_{AC1}(g\theta, d\theta) \text{ and } \Phi\rho \text{ satisfiable}\}$$

$$\text{if } \begin{cases} \Psi \mid g \to d \in R \\ r \xrightarrow[\Psi|g\to d/AC1]{\theta} r' \\ \Phi' = C_{AC1}(l, r') \end{cases}$$

Compose extension is similar to *compose rule*.

Simplify

$$E \cup \{l = r\}; N; P \vdash E \cup \{l' = r\} \cup \{l\rho = r\rho \mid \rho \in \Theta_{AC1}(g\theta, d\theta)\}; N; P$$

$$\text{if } \begin{cases} \Psi \mid g \to d \in R \\ l \xrightarrow[\Psi \mid g \to d / AC1]{\theta} l' \end{cases}$$

Simplify reduces equations as usual.

Collapse

$$E; N \cup \{\Phi \mid l \to r\}; P \vdash E \cup \{l' = r\}; N \cup \{\Phi\rho \mid l\rho \to r\rho \mid \rho \in \Theta_{AC1}(g\theta, d\theta) \text{ and } \Phi\rho \text{ satisfiable}\}; P$$

$$\text{if } \begin{cases} \Psi \mid g \to d \in R \\ l \xrightarrow[\Psi \mid g \to d / AC1]{\theta} l' \\ (l, r) \succ\!\!\succ (g\theta, d\theta) \end{cases}$$

Collapse reduces the left-hand side of constrained rules resulting in new (unconstrained) equations. The ordering $\succ\!\!\succ$ used in the condition is $(\lhd, \succ)_{lex}$ (in fact, any terminating ordering would do) where $\lhd$ is defined as follows: $s \lhd t$ iff $|s\downarrow| >_{\mathbb{N}} |t\downarrow|$ ($|s|$ is the size of s, i.e. the cardinal of $\mathcal{P}os(s)$). If we assume that all rules are normalized, this is equivalent to $|s| >_{\mathbb{N}} |t|$ (we can use also strict-subterm modulo AC, which is more or less the same as comparing the sizes). $\succ\!\!\succ$ is terminating.

Lemma 3.1 *Assume that the equation $l = r$ is valid in $E \cup R \cup AC1$ for all $\Phi \mid l \to r \in R$. Then*

$$E; N; P \vdash E'; N'; P' \Longrightarrow =_{E \cup R \cup AC1} \equiv =_{E' \cup R' \cup AC1}$$

hence $l = r$ is valid in $E' \cup R' \cup AC1$ for all $\Phi \mid l \to r \in R'$.

Proof. $=_{E \cup R \cup AC1} \subseteq =_{E' \cup R' \cup AC1}$ is clear. $=_{E' \cup R' \cup AC1} \subseteq =_{E \cup R \cup AC1}$ follows from the hypothesis. □

3.2 Correctness

Definition 3.1 *An AC1-completion algorithm is an algorithm which takes as input a set of equations E_0 and an AC1-rewrite ordering $\succ$ and produces a (finite or infinite) sequence $(E_n; N_n; P_n)$ where $R_0 = \phi$ and for all i, $E_i; N_i; P_i \vdash E_{i+1}; N_{i+1}; P_{i+1}$. Let:*

$$\begin{cases} E_\infty = \bigcup_{n=0}^{\infty} \left(\bigcap_{i=n}^{\infty} E_i \right) \;, \; N_\infty = \bigcup_{n=0}^{\infty} \left(\bigcap_{i=n}^{\infty} N_i \right) \\ P_\infty = \bigcup_{n=0}^{\infty} \left(\bigcap_{i=n}^{\infty} P_i \right) \;, \; R_\infty = \bigcup_{n=0}^{\infty} \left(\bigcap_{i=n}^{\infty} R_i \right) = N_\infty \cup P_\infty \end{cases}$$

E_∞ and R_∞ are respectively the sets of persisting equations and the set of persisting rules. We say that the algorithm fails *if E_∞ is not empty,* diverges *if R_∞ is infinite, and* succeeds *otherwise.*

Theorem 3.1 (Correctness)

If completion does not fail, $=_{E_0 \cup AC1} \equiv =_{R_\infty \cup AC1}$.

This result follows from the previous lemma, but we must assume that the starting set of rules does not contain any constrained rule. This is true in practice since we usually start from an empty R_0. So, this method does not permit to handle fields' theory, for which we would want to start with the constrained rule $x \neq 0 \mid x * x^{-1} \to 1$.

3.3 Fairness and Completeness

Definition 3.2 *A derivation* $E_0; N_0; P_0 \vdash E_1; N_1; P_1 \vdash \ldots$ *is* fair *if all persisting critical pairs and extensions are computed, i.e.*

$$\begin{cases} CP_{AC1}(R_\infty) \subset \bigcup_{i=0}^{\infty} E_i \\ Ext_{AC1}(R_\infty) \subset \bigcup_{i=0}^{\infty} R_i \end{cases}$$

An AC1-completion algorithm is fair if all sequences that it produces are fair.

Fairness is fundamental in completion procedures, it expresses completeness of the search strategy. In practice, we also like that the simplification rules are used as much as possible. This yields sets of rules which are inter-reduced, an important property as far as the uniqueness of the completion result is concerned.

We will prove in 3.5 that as in AC-completion, extensions of extensions are redundant. This will prove that in the rule *Extend* we can consider only $\Phi \mid l \to r \in N$, and the second condition for fairness can be transformed to:

$$Ext_{AC1}(N_\infty) \subset \bigcup_{i=0}^{\infty} R_i$$

Theorem 3.2 (Completeness) *Assume that the completion is fair and succeeds. Then,*

$$\forall s, t \quad s =_{E_0 \cup AC1} t \Longrightarrow \exists u, v \quad s \xrightarrow[AC1\backslash R_\infty]{*} u =_{\mathrm{AC1}} v \xleftarrow[AC1\backslash R_\infty]{*} t$$

The proof of this result is by the proof normalization method. Such a proof needs previously a formal definition of the AC1-proofs algebra:

Definition 3.3 • *Elementary proofs are*

$$s \underset{g=d}{\overset{\sigma,p}{\longleftarrow_{AC1}\longrightarrow}} t \quad \textit{if} \quad \begin{cases} g = d \in AC1 \\ s|_p = g\sigma \\ t = s[d\sigma]_p \end{cases}$$

$$s \underset{l=r}{\overset{\sigma,p}{\longleftrightarrow}} t \quad \textit{if} \quad \begin{cases} s|_p = l\sigma \\ t = s[r\sigma]_p \end{cases}$$

$$s \xrightarrow[\Phi|l\to r_{(x)}]{\sigma,p} t \quad \textit{if} \quad \begin{cases} s|_p = l\sigma \\ t = s[r\sigma]_p \\ \Phi\sigma \textit{ satisfiable} \\ x \in \{N,P\} \text{ (rule status)} \end{cases}$$

$$s \xleftarrow[\Phi|l\to r_{(x)}]{\sigma,p} t \quad \textit{if} \quad \begin{cases} t|_p = l\sigma \\ s = t[r\sigma]_p \\ \Phi\sigma \textit{ satisfiable} \\ x \in \{N,P\} \text{ (rule status)} \end{cases}$$

- *AC1-proofs are sequences of elementary proofs: given two proofs P and Q, PQ is a proof if last(P) =first(Q) where last and first are given by*

$$\begin{cases} \textit{first}(s\ldots t) = s \textit{ and } \textit{last}(s\ldots t) = t \textit{ if } s\ldots t \textit{ is an elementary proof} \\ \textit{first}(PQ) = \textit{first}(P) \textit{ and } \textit{last}(PQ) = \textit{last}(Q) \end{cases}$$

- *We will use an abbreviation:*

$$s \overset{(\geq p)^*}{\longleftarrow_{AC1}\longrightarrow} t \text{ if } \begin{cases} k \geq 0 \\ s = s_0 \underset{g_1=d_1}{\overset{\sigma_1,p_1}{\longleftarrow_{AC1}\longrightarrow}} s_1 \ldots \underset{g_k=d_k}{\overset{\sigma_k,p_k}{\longleftarrow_{AC1}\longrightarrow}} s_k = t \\ \forall i, p_i \geq p \end{cases}$$

- *A proof P is an $E;R$-proof if all equations (resp. rules) that appear in its elementary subproofs are in E (resp. R).*

3.3.1 Rewriting rules on proofs

We now define a rewriting system that rewrites AC1-proofs. It is made of two kinds of rules. The first batch expresses properties of derivations: for example, rewriting at parallel redexes commutes. The second batch expresses the transformations on proofs resulting from the application of an inference rule. Rewrite rules on proofs obtained by symmetry from another rule are skipped.

We start with the rules of the algebra of derivations. These rules are usual, and derive from Jouannaud and Kirchner's E-critical pairs lemma [10]. They allow to rewrite "peaks" and "cliffs":

Cliff elimination 1 (non overlap)

$$s \underset{g=d}{\overset{\sigma,p}{\longleftarrow_{AC1}\longrightarrow}} t \overset{(\geq q)^*}{\longleftarrow_{AC1}\longrightarrow} t' \xrightarrow[\Phi|l\rightarrow r_{(x)}]{\sigma,q} u \Longrightarrow s \overset{(\geq q)^*}{\longleftarrow_{AC1}\longrightarrow} s' \xrightarrow[\Phi|l\rightarrow r_{(x)}]{\sigma,q} v \underset{g=d}{\overset{\sigma,p}{\longleftarrow_{AC1}\longrightarrow}} u$$

$$\text{if } p \wedge q$$

Cliff elimination 2 (overlap under a variable)

$$s \underset{g=d}{\overset{\sigma,p}{\longleftarrow_{AC1}\longrightarrow}} t \overset{(\geq pq)^*}{\longleftarrow_{AC1}\longrightarrow} t' \xrightarrow[\Phi|l\rightarrow r_{(x)}]{\sigma,pq} u \Longrightarrow$$

$$s \left(\overset{(\geq po'q')^*}{\longleftarrow_{AC1}\longrightarrow} \xrightarrow[\Phi|l\rightarrow r_{(x)}]{\sigma,po'q'} \right)^+_{o'\in P'} s' \underset{g=d}{\overset{\sigma',p}{\longleftarrow_{AC1}\longrightarrow}} u' \left(\xleftarrow[\Phi|l\rightarrow r_{(x)}]{\sigma,poq'} \overset{(\geq poq')^*}{\longleftarrow_{AC1}\longrightarrow} \right)^*_{o\in P\setminus\{p'\}} u$$

$$\text{where } \begin{cases} q \notin \mathcal{FP}os(d),\ q \neq \Lambda \\ q = p'q' \text{ with } d|_{p'} = x \in X \\ P = \{o \in \mathcal{P}os(d) | d|_o = x\} \\ P' = \{o' \in \mathcal{P}os(g) | g|_{o'} = x\} \\ \sigma' : \begin{cases} x\sigma' = x\sigma[r\sigma]_q \\ \text{if } y \neq x,\ y\sigma' = y\sigma \end{cases} \end{cases}$$

Peak elimination 1 (non overlap)

$$s \xleftarrow[\Phi|l\rightarrow r_{(x)}]{\sigma,p} t \overset{(\geq q)^*}{\longleftarrow_{AC1}\longrightarrow} t' \xrightarrow[\Psi|g\rightarrow d_{(y)}]{\sigma,q} u \Longrightarrow s \overset{(\geq q)^*}{\longleftarrow_{AC1}\longrightarrow} s' \xrightarrow[\Psi|q\rightarrow d_{(y)}]{\sigma,q} v \xleftarrow[\Phi|l\rightarrow r_{(x)}]{\sigma,p} u$$

$$\text{if } p \wedge q$$

Peak elimination 2 (variable overlap)

$$s \xleftarrow[\Phi|l\rightarrow r_{(x)}]{\sigma,p} t \overset{(\geq pq)^*}{\longleftarrow_{AC1}\longrightarrow} t' \xrightarrow[\Psi|g\rightarrow d_{(y)}]{\sigma,pq} u \Longrightarrow$$

$$s \left(\overset{(\geq po'q')^*}{\longleftarrow_{AC1}\longrightarrow} \xrightarrow[\Psi|g\rightarrow d_{(y)}]{\sigma,po'q'} \right)^*_{o'\in P'} s' \xleftarrow[\Phi|l\rightarrow r_{(x)}]{\sigma',p} u' \left(\xleftarrow[\Psi|g\rightarrow d_{(y)}]{\sigma,poq'} \overset{(\geq poq')^*}{\longleftarrow_{AC1}\longrightarrow} \right)^*_{o\in P\setminus\{p'\}} u$$

$$\text{if } \begin{cases} q \notin \mathcal{FP}os(l) \\ q = p'q' \text{ with } l|_{p'} = x \in X \\ P = \{o \in \mathcal{P}os(l) | l|_o = x\} \\ P' = \{o' \in \mathcal{P}os(r) | r|_{o'} = x\} \\ \sigma' : \begin{cases} x\sigma' = x\sigma[d\sigma]_{q'} \\ \text{if } y \neq x,\ y\sigma' = y\sigma \end{cases} \end{cases}$$

We now continue with the rules associated to the inferences performed on equations and rules:

Orient 1

$$s \overset{\sigma,p}{\underset{l=r}{\longleftrightarrow}} t \Longrightarrow s \xrightarrow[\Phi|l\to r_{(N)}]{\sigma,p} t$$

$$\text{if } \begin{cases} \Phi = C_{AC1}(l,r) \\ \Phi\sigma \text{ satisfiable} \end{cases}$$

Orient 2

$$s \overset{\sigma,p}{\underset{l=r}{\longleftrightarrow}} t \Longrightarrow s \overset{\lambda,p}{\underset{l\rho=r\rho}{\longleftrightarrow}} t$$

$$\text{if } \begin{cases} \Phi = C_{AC1}(l,r) \\ \Phi\sigma \text{ unsatisfiable then } \exists\rho \in \Theta_{AC1}(l,r), \exists\lambda \ / \ \sigma = \rho\lambda \end{cases}$$

We need two rules here because the substitution σ may or may not satisfy Φ .

Extend

$$s \overset{\sigma,p}{\underset{g=d}{\longleftarrow_{AC1}\!\!\longrightarrow}} t \overset{(\geq pq)^*}{\longleftarrow_{AC1}\!\!\longrightarrow} t' \xrightarrow[\Phi|l\to r_{(x)}]{\sigma,pq} u \Longrightarrow s \overset{(\geq p)^*}{\longleftarrow_{AC1}\!\!\longrightarrow} s' \xrightarrow[\Psi|g\theta\to d\theta[r\theta]_{q(P)}]{\sigma',p} u' \overset{(\geq p)^*}{\longleftarrow_{AC1}\!\!\longrightarrow} u$$

$$\text{if } \begin{cases} q \in \mathcal{FP}os(d) \text{ with } q \neq \Lambda \\ d|_q\sigma =_{AC1} l\sigma \text{ then} \\ \exists\theta \in CSU_{AC1}(d|_q, l) \mid \sigma =_{\mathrm{AC1}} \theta\sigma' \end{cases}$$

We may use the same σ in the left proof without loss of generality, since we can always assume that d and l have no variable in common.

Deduce

$$s \xleftarrow[\Phi|l\to r_{(x)}]{\sigma,p} t \overset{(\geq pq)^*}{\longleftarrow_{AC1}\!\!\longrightarrow} t' \xrightarrow[\Psi|g\to d_{(y)}]{\sigma,pq} u \Longrightarrow s \overset{(\geq p)^*}{\longleftarrow_{AC1}\!\!\longrightarrow} s' \overset{\sigma',p}{\underset{r\theta=l\theta[d\theta]_q}{\longleftrightarrow}} u' \overset{(\geq p)^*}{\longleftarrow_{AC1}\!\!\longrightarrow} u$$

$$\text{if } \begin{cases} q \in \mathcal{FP}os(l) \\ l|_q\sigma =_{AC1} g\sigma \text{ then} \\ \exists\theta \in CSU_{AC1}(l|_q, g) \mid \sigma =_{\mathrm{AC1}} \theta\sigma' \end{cases}$$

Delete

$$s \overset{\sigma,p}{\underset{l=r}{\longleftrightarrow}} t \Longrightarrow s \overset{(\geq p)^*}{\longleftarrow_{AC1}\!\!\longrightarrow} t$$

$$\text{if } l =_{AC1} r$$

Compose 1

$$s \xrightarrow[\Phi|l\to r_{(x)}]{\sigma,p} t \Longrightarrow s \xrightarrow[\Phi'|l\to r'_{(x)}]{\sigma,p} u \xleftarrow[\Psi|g\to d_{(y)}]{\theta\sigma,pq} t' \xleftrightarrow[AC1]{(\geq p)^*} t$$

$$\text{if } \begin{cases} r \xrightarrow[\Psi|g\to d_{(y)}/AC1]{q} r' \\ \Psi\theta\sigma \text{ satisfiable} \end{cases}$$

Compose 2

$$s \xrightarrow[\Phi|l\to r_{(x)}]{\sigma,p} t \Longrightarrow s \xrightarrow[\Phi\rho|l\rho\to r\rho_{(x)}]{\lambda,p} t$$

$$\text{if } \begin{cases} r \xrightarrow[\Psi|g\to d_{(y)}/AC1]{q} r' \\ \Psi\theta\sigma \text{ unsatisfiable then } \exists\rho \in \Theta_{AC1}(g\theta, d\theta) \mid \sigma = \rho\lambda \text{ (because } \Psi\theta \text{ satisfiable)} \end{cases}$$

We again have two cases here, depending on whether $\theta\sigma$ satisfies Φ or not. If it does, then the right-hand side of the rule $l \to r$ is simplified, otherwise, the rule itself is instantiated by ρ.

Simplify 1

$$s \xleftrightarrow[l=r]{\sigma,p} t \Longrightarrow s \xleftrightarrow[AC1]{(\geq p)^*} s' \xrightarrow[\Phi|g\to d_{(y)}]{\theta\sigma,pq} u \xleftrightarrow[l'=r]{\sigma,p} t$$

$$\text{if } \begin{cases} l \xrightarrow[\Psi|g\to d_{(y)}/AC1]{q} l' \\ \Psi\theta\sigma \text{ satisfiable} \end{cases}$$

Simplify 2

$$s \xleftrightarrow[l=r]{\sigma,p} t \Longrightarrow s \xleftrightarrow[l\rho=r\rho]{\lambda,p} t$$

$$\text{if } \begin{cases} l \xrightarrow[\Psi|g\to d_{(y)}/AC1]{q} l' \\ \Psi\theta\sigma \text{ unsatisfiable then } \exists\rho \in \Theta_{AC1}(g\theta, d\theta) \mid \sigma = \rho\lambda \text{ (because } \Psi\theta \text{ satisfiable)} \end{cases}$$

The situation here is very similar to the previous one.

Collapse 1

$$s \xrightarrow[\Phi|l\to r_{(N)}]{\sigma,p} t \Longrightarrow s \xleftrightarrow[AC1]{(\geq p)^*} s' \xrightarrow[\Psi|g\to d_{(y)}]{\theta\sigma,pq} u \xleftrightarrow[l'=r]{\sigma,p} t$$

$$\text{if } \begin{cases} l \xrightarrow[\Psi|g\to d_{(y)}/AC1]{q} l' \\ (l, r) \succ (g\theta, d\theta) \\ \Psi\theta\sigma \text{ satisfiable} \end{cases}$$

Collapse 2

$$s \xrightarrow[\Phi|l \to r_{(N)}]{\sigma,p} t \Longrightarrow s \xrightarrow[\Phi\rho|l\rho \to r\rho_{(N)}]{\lambda,p} t$$

$$\text{if} \left\{ \begin{array}{l} l \xrightarrow[\Psi|g \to d_{(y)}/AC1]{q} l' \\ (l,r) \succ (g\theta, d\theta) \\ \Psi\theta\sigma \text{ unsatisfiable then } \exists\rho \in \Theta_{AC1}(g\theta, d\theta) \mid \sigma = \rho\lambda \text{ (because } \Psi\theta \text{ satisfiable)} \end{array} \right.$$

3.3.2 Correctness of the rewriting system on proofs

We show here that each rewrite rule on proofs is valid, i.e. transforms an existing proof into a new proof using the inferred rules or equations. This part of the proof is very tedious, but helps in understanding some critical points in the rules, like the use of certain $AC1$-equalities. We give here only the proof of correctness of *Extend*:

We have the hypothesis $\left\{ \begin{array}{l} s|_p = g\sigma \\ t = s[d\sigma]_p \end{array} \right.$, $\left\{ \begin{array}{l} t|_{pq} =_{\mathrm{AC1}} l\sigma \\ u = t[r\sigma]_{pq} \\ \Phi\sigma \text{ satisfiable} \end{array} \right.$ and $\left\{ \begin{array}{l} q \in \mathcal{FP}os(d) \backslash \{\Lambda\} \\ d|_q\theta =_{\mathrm{AC1}} l\theta \\ \sigma =_{\mathrm{AC1}} \theta\sigma' \end{array} \right.$

Expected conclusions: $\left\{ \begin{array}{ll} (1) & s|_p =_{\mathrm{AC1}} (g\theta)\sigma' \\ (2) & u' = s[d\theta[r\theta]_q\sigma']_p \\ (3) & \Psi\sigma' \text{ satisfiable} \end{array} \right.$ and $\left\{ \begin{array}{ll} (4) & u'|_p =_{\mathrm{AC1}} u|_p \\ (5) & u = u'[u|_p]_p \end{array} \right.$

proofs:

(1) $s|_p = g\sigma =_{\mathrm{AC1}} g\theta\sigma'$

(2) we take this as definition of u'

(3) $\Phi\sigma$ satisfiable so $l\sigma \succ r\sigma$
$(g\theta)\sigma' =_{\mathrm{AC1}} (d\theta)\sigma' =_{\mathrm{AC1}} d\theta[l\theta]_q\sigma' =_{\mathrm{AC1}} d\sigma[l\sigma]_q \succ d\sigma[r\sigma]_q =_{\mathrm{AC1}} d\theta[r\theta]_q\sigma'$
so $\Psi\sigma'$ satisfiable by definition of Ψ

(4) $u'|_p = d\theta[r\theta]_q\sigma' = d\theta\sigma'[r\theta\sigma']_q =_{\mathrm{AC1}} d\sigma[r\sigma]_q = t|_p[r\sigma]_q = u|_p$

(7) clear

3.3.3 Termination of the rewrite system on proofs

- We first define an ordering on proofs as follows (this is a variant of the general method described in [1]):

  - We first define the notion of boundedness of a AC1-step in a proof: in a proof $t_0 \vdash\!\dashv t_1 \vdash\!\dashv .. \vdash\!\dashv t_n$, an AC1-step $t_{i-1} \vdash_{AC1} \dashv t_i$ is said to be bound on the right (resp. left) if $\exists j > i$ (resp $j < i-1$) such that the subproof $t_{i-1} \vdash\!\dashv .. \vdash\!\dashv t_j$ (resp. $t_j \vdash\!\dashv .. \vdash\!\dashv t_i$) is an AC1-extended rewrite proof $t_{i-1} \overset{(\geq p)^*}{\longleftarrow_{AC1}\longrightarrow} t_{j-1} \xrightarrow[\Phi|l \to r]{\sigma,p} t_j$ (resp $t_j \xleftarrow[\Phi|l \to r]{\sigma,p} t_{j+1} \overset{(\geq p)^*}{\longleftarrow_{AC1}\longrightarrow} t_i$).
  - The complexity $C(P)$ of a proof $P = t_0 \vdash\!\dashv .. \vdash\!\dashv t_n$ is the multiset of elementary complexities

$\{C_P(t_0 \vdash\!\dashv t_1), .., C_P(t_{n-1} \vdash\!\dashv t_n)\}$ where C_P is a quintuple defined below

$$\begin{array}{rcl}
C_P(t_{i-1} \underset{l=r}{\overset{\sigma,p}{\longleftarrow_{AC1}\longrightarrow}} t_i) & = & \langle \{t_{i-1}\}, \quad \bot, \quad b^i_P, \quad \bot, \quad \bot \rangle \\
C_P(t_{i-1} \underset{l=r}{\overset{\sigma,p}{\longleftrightarrow}} t_i) & = & \langle \{t_{i-1}, t_i\}, \quad \bot, \quad \bot, \quad \bot, \quad |\mathcal{V}ar(l) \cup \mathcal{V}ar(r)| \rangle \\
C_P(t_{i-1} \xrightarrow[\Phi|l\to r_{(N)}]{\sigma,p} t_i) & = & \langle \{t_{i-1}\}, \quad l\sigma, \quad \bot, \quad r\sigma, \quad \bot \rangle \\
C_P(t_{i-1} \xrightarrow[\Phi|l\to r_{(P)}]{\sigma,p} t_i) & = & \langle \{t_{i-1}\}, \quad \bot, \quad \bot, \quad r\sigma, \quad \bot \rangle \\
C_P(t_{i-1} \xleftarrow[\Phi|l\to r_{(N)}]{\sigma,p} t_i) & = & \langle \{t_i\}, \quad l\sigma, \quad \bot, \quad r\sigma, \quad \bot \rangle \\
C_P(t_{i-1} \xleftarrow[\Phi|l\to r_{(P)}]{\sigma,p} t_i) & = & \langle \{t_i\}, \quad \bot, \quad \bot, \quad r\sigma, \quad \bot \rangle
\end{array}$$

where $\bot$ is a minimum element, $b^i_P = 0$ (resp. $1, 2$) if $t_{i-1} \vdash\!\dashv t_i$ is bound on the left and on the right (resp. on the left or on the right, neither on the left nor on the right). One can verify that the complexity of symmetric proofs are identical. Two elementary complexities are compared in the lexicographic extension of the orderings $>_{\text{mul}}$ for the first component, $\lhd$ for the second, $>_{\mathbb{N}}$ for the third, the fourth and the last, $\succ$ for the fifth.

– Proofs are compared in the multiset extension $\succ_P$ of the above ordering, i.e.:

$$\succ_P = \{(>_{\text{mul}}, \lhd, >_{\mathbb{N}}, \succ, >_{\mathbb{N}})_{lex}\}_{mul}$$

This ordering is terminating, since it is built up from terminating orderings with the functionals *lex* and *mul* which preserve terminating orderings. Because of the presence of b^i_P this ordering is not monotonic.

- We are left with the proof that each rule $P \Longrightarrow Q$ satisfies $R[P] \succ_P R[Q]$ where R is any context.

  This is again a bit tedious, we do it here for the only two rules *Extend*, *Collapse 1* and Cliff Elimination 2:

Extend If $x =_N$ then it is straightforward.

If $x =_P$: $q \neq \Lambda$ so the first AC1-step is not bounded on the right, then

$$C(R[s \underset{g=d}{\overset{\sigma,p}{\longleftarrow_{AC1}\longrightarrow}} t \overset{(\geq pq)^*}{\longleftarrow_{AC1}\longrightarrow} t' \xrightarrow[\Phi|l\to r_{(x)}]{\sigma,pq} u]) = \{...; \langle\{s\}, \bot, 2(1), ...\rangle; ...\}$$

and

$$C(R[s \overset{(\geq p)^*}{\longleftarrow_{AC1}\longrightarrow} s' \xrightarrow[\Psi|g\theta\to d\theta[r\theta]_{q(P)}]{\sigma',p} u' \overset{(\geq p)^*}{\longleftarrow_{AC1}\longrightarrow} u]) =$$

$$\{...; \langle\{s\}, \bot, 1(0), ...\rangle^* \langle\{s'\}, \bot, \bot, ...\rangle; \langle\{u\}, ...\rangle^*; ...\}$$

We conclude here by $s =_{\text{AC1}} t =_{\text{AC1}} s'$, $2 >_{\mathbb{N}} 1 (1 >_{\mathbb{N}} 0)$ and $s =_{\text{AC1}} s' \succ u' =_{\text{AC1}} u$

Collapse 1 In this case, we must take care of the context because the rewrite proof which is collapsed can participate to the boundedness of some AC1-steps in the context. Fortunately, we allow to collapse only non-protected rules:

$$C(R[s \xrightarrow[\Phi|l\to r_{(N)}]{\sigma,p} t]) \quad = \quad \{...; \langle\{s\}, l\sigma, \bot, r\sigma, \bot\rangle; ...\}$$

If $x =_N$ then

$$C(R[s \overset{(\geq p)^*}{\longleftarrow_{AC1}\longrightarrow} s' \xrightarrow[\Psi|g\to d_{(N)}]{\theta\sigma,pq} u \underset{l'=r}{\overset{\sigma,p}{\longleftrightarrow}} t]) =$$

$$\{...;\langle\{s\},\perp,...\rangle^*,\langle\{s'\},g\theta\sigma,\perp,d\theta\sigma,\perp\rangle,\langle\{u,t\},...\rangle;...\}$$

We conclude here as follows: $l \lhd \perp$, $(l,r) \succ(g\theta,d\theta)$ (because $l|q =_{\mathrm{AC1}} g\theta$) and $\begin{cases} s \succ u \\ s \succ t \end{cases}$ hence $\{s\} >_{\mathrm{mul}} \{u,t\}$, and $\langle\{s\},l\sigma,\perp,r\sigma,\perp,\rangle$ is bigger than all complexities of the AC1-steps of R. If $x =_P$ then it is straightforward (replace $g\theta\sigma$ by $\perp$).

Remark: The other rules where this phenomenon can happen are *Compose 1*, *Compose 2* and *Collapse 2* but in these cases, the complexity of the context remains the same, because in these rules, the rewrite step is replaced by an other rewrite step at the same position.

Cliff Elimination 2 We have at least one rewrite step at the left of the right-hand side proof, so it can be written as:

$$s \xleftrightarrow[AC1]{(\geq po_1'q'')^*} s_1' \xrightarrow[\Phi|l\to r_{(x)}]{\sigma,po_1'q''} s_1...s_n = s' \xleftrightarrow[AC1]{\sigma,p} \underset{g=d}{} u' = u_k...u_0 = u$$

If $x =_N$ (note that $q \neq \Lambda$) then

$$C(R[s \underset{g=d}{\xleftrightarrow[AC1]{\sigma,p}} t \xleftrightarrow[AC1]{(\geq pq)^*} t' \xrightarrow[\Phi|l\to r_{(x)}]{\sigma,pq} u]) = \{...;\langle\{s\},\perp,2(1),...\rangle;...;\langle\{t\},l\sigma,\perp,r\sigma,\perp\rangle;...\}$$

and

$$C(R[s \xleftrightarrow[AC1]{(\geq po_1'q'')^*} s_1' \xrightarrow[\Phi|l\to r_{(x)}]{\sigma,po_1'q''} s_1...s_n = s' \underset{g=d}{\xleftrightarrow[AC1]{\sigma,p}} u' = u_k...u_0 = u]) =$$

$$\left\{...;\langle\{s\},\perp,1(0),...\rangle^*;\langle\{s_1'\},l\sigma,\perp,r\sigma,\perp\rangle;\langle\{s_i\},...\rangle^*_{1\leq i\leq n};\langle\{s'\},...\rangle;\langle\{u_j\},...\rangle^*_{0\leq j\leq k};...\right\}$$

We conclude here by $\langle\{t\},l\sigma,\perp,r\sigma,\perp\rangle \simeq \langle\{s_1'\},l\sigma,\perp,r\sigma,\perp\rangle$, $2 >_{\mathbb{N}} 1(1 >_{\mathbb{N}} 0)$ and $s \succ s_i (i \geq 1)$, $s \succ s'$, $s \succ u_j$.

If $x =_P$ then

$$C(R[s \underset{g=d}{\xleftrightarrow[AC1]{\sigma,p}} t \xleftrightarrow[Ac1]{(\geq pq)^*} t' \xrightarrow[\Phi|l\to r_{(x)}]{\sigma,pq} u]) = \{...;\langle\{s\},\perp,2(1),...\rangle;...\}$$

and

$$C(R[s \xleftrightarrow[AC1]{(\geq po_1'q'')^*} s_1' \xrightarrow[\Phi|l\to r_{(x)}]{\sigma,po_1'q''} s_1...s_n = s' \xleftrightarrow[g=d]{\sigma,p} u' = u_k...u_0 = u]) =$$

$$\left\{...;\langle\{s\},\perp,1(0),...\rangle^*;\langle\{s_1'\},\perp,\perp,...\rangle;\langle\{s_i\},...\rangle^*_{1\leq i\leq n};\langle\{s'\},...\rangle;\langle\{u_j\},...\rangle^*_{0\leq j\leq k};...\right\}$$

We conclude here by $2 >_{\mathbb{N}} 1(0)$, $(1 >_{\mathbb{N}} 0)$, $2(1) >_{\mathbb{N}} \perp$ and $s \succ s_i(i \geq 2)$, $s \succ s'$, $s \succ u_j$.

3.3.4 Completeness proof

The rewrite system $\Longrightarrow$ schematizes a class of rewrite systems $\underset{E;R}{\Longrightarrow}$ (for all $E;R$ $P \underset{E;R}{\Longrightarrow} Q$ if P,Q are $E;R$-proofs and $P \Longrightarrow Q$). We can now prove our completeness result as follows: let

$$\mathcal{E}_i = \bigcup_{j=0}^{i} E_j \text{ and } \mathcal{R}_i = \bigcup_{j=0}^{i} R_j$$

The sequence of inferences from $E_0;R_0$ can be reflected on proofs in the sense that an $E_0;R_0$-proof is rewritten by $\underset{\mathcal{E}_0;\mathcal{R}_0}{\Longrightarrow}$ yielding an $E_0;R_0$-proof, which is also an $\mathcal{E}_1;\mathcal{R}_1$-proof and can be in turn rewritten by $\underset{\mathcal{E}_1;\mathcal{R}_1}{\Longrightarrow}$, etc.... If P is a $\mathcal{E}_i;\mathcal{R}_i$-proof, let $P\downarrow_i$ denote a normal form of P for $\underset{\mathcal{E}_i;\mathcal{R}_i}{\Longrightarrow}$, which is also an $\mathcal{E}_i;\mathcal{R}_i$-proof.

Lemma 3.2 *$P\downarrow_i$ is actually an $E_i; R_i$-proof.*

Assume now that $s =_{E_0 \cup AC1} t$. Then there exists an $E_0; R_0$-proof P such that $s = \mathrm{first}(P)$ et $t = \mathrm{last}(P)$. The sequence of inferences performed by completion can be reflected in a sequence of proofs

$$P_0 = P\downarrow_0 \stackrel{*}{\Longrightarrow} P_1 = P_0\downarrow_1 \stackrel{*}{\Longrightarrow} P_2 = P_1\downarrow_2 \stackrel{*}{\Longrightarrow} \dots$$

Since $\Longrightarrow$ is terminating, this sequence is stationary, hence there exists some natural n such that

$$P_n = P_{n+1} = P_{n+2} = \dots$$

Since P_n is an $E_k; R_k$-proof for all $k \geq n$ by application of the lemma, it is also an $E_\infty; R_\infty$-proof. Since completion does not fail, E_∞ is empty, hence P_n contains only rewrites and AC1-equational steps. Since it is in normal form for $\underset{\mathcal{E}_n;\mathcal{R}_n}{\Longrightarrow}$, the only possible cliffs and peaks in P_n are critical ones. Because of fairness, such peaks and cliffs must have been computed, either at some previous step $j < n$, but then P_n would not be in normal form for $\underset{\mathcal{E}_n;\mathcal{R}_n}{\Longrightarrow}$, a contradiction, or at some further step $j \geq n$, but then P_n would not be in normal form for $\underset{\mathcal{E}_j;\mathcal{R}_j}{\Longrightarrow}$, another contradiction.

Hence, P_n has the form

$$\begin{array}{ccccc} s & & & & t \\ & \underset{AC1\backslash R_\infty}{\searrow^{*}} & & \underset{AC1\backslash R_\infty}{{}^{*}\swarrow} & \\ & & u =_{AC1} v & & \end{array}$$

3.4 Remarks

- In practice, extended rewriting is used instead of rewriting modulo. Actually, implementations use flattened terms, which gives a mixture of both relations.
- *Collapse* can be improved in order to get a set of rules R_∞ which is "more" inter-reduced: if extensions of the collapsed rule $l \to r$ have already been calculated, they can be removed. This preserves correctness since extensions are equational consequences of the rule and the theory AC1, and it preserves completeness because a rewrite proof with an extension can be transformed to a proof using AC1-steps and an equationnal step $l' = r$, and a rewrite step with $g \to d$. It means that in an implementation, extensions of a rule must be linked to the rule in some way (labeling, numbering,...).
- An example of a fair completion procedure is described in [10].

3.5 Redundancy of extensions of extensions

Lemma 3.3 *The only extensions needed are extensions of rules of the form $\Phi \mid l_1 + l_2 \to r$, which are extended in $\Phi \mid (x + l_1) + l_2 \to x + r$ and $\Phi \mid l_1 + (l_2 + x) \to r + x$ ($x \notin \mathcal{V}ar(l)$).*

Proof. The axiom $g = d$ used to calculate an extension must possess a ground position $q \neq \Lambda$. As in the AC case, in the AC1-theory only the associativity $x + (y + z) = (x + y) + z$ can be used. All critical cliffs have the form (or the similar form with $(x + y) + z = x + (y + z)$):

$$s \xleftarrow[x+(y+z)=(x+y)+z]{\sigma,p \quad AC1} t \xrightarrow[\Phi|l\to r]{\sigma,p.1} u$$

Then we have $\begin{cases} s = w[s_1 + (s_2 + s_3)]_p \\ t = w[(s_1 + s_2) + s_3]_p \\ s_1 + s_2 =_{\mathrm{AC1}} l\sigma \\ u = w[r\sigma + s_3]_p \end{cases}$. Three cases can arise here:

- $s_1 =_{\mathrm{AC1}} 0$ then $l\sigma =_{\mathrm{AC1}} s_1 + s_2 =_{\mathrm{AC1}} s_2$. The critical cliff commutes without adding any extension in that way:

$$s = w[s_1 + (s_2 + s_3)]_p \xleftrightarrow[AC1]{(\geq p.2.1)^*} w[s_1 + (l\sigma + s_3)]_p \xrightarrow[\Phi|l\to r]{\sigma,p.2.1} w[s_1 + (r\sigma + s_3)]_p$$

$$\xleftrightarrow[AC1]{(\geq p)^*} w[r\sigma + s_3]_p = u$$

- $s_2 =_{\mathrm{AC1}} 0$ then $l\sigma =_{\mathrm{AC1}} s_1 + s_2 =_{\mathrm{AC1}} s_1$. The critical cliff commutes without adding any extension in that way:

$$s = w[s_1 + (s_2 + s_3)]_p \xleftrightarrow[AC1]{(\geq p.1)^*} w[l\sigma + (s_2 + s_3)]_p \xrightarrow[\Phi|l\to r]{\sigma,p.1} w[r\sigma + (s_2 + s_3)]_p$$

$$\xleftrightarrow[AC1]{(\geq p.2)^*} w[r\sigma + s_3]_p = u$$

- else, the identity can not be applied at the top in the proof of $s_1 + s_2 =_{\mathrm{AC1}} l\sigma$ so $l = l_1 + l_2$. The cliff can be extended now as:

$$s = w[s_1 + (s_2 + s_3)]_p \xleftrightarrow[AC1]{(\geq p)^*} w[l\sigma + s_3]_p \xrightarrow[\Phi|l_1+(l_2+x)\to r+x]{\sigma,p} w[r\sigma + s_3]_p = u$$

Notice that extensions have the same constraint as the rule, because if $(l_1 + l_2)\sigma \succ r\sigma$ then $((x+l_1)+l_2)\sigma \succ (x + r)\sigma$ even if $x\sigma =_{\mathrm{AC1}} 0$. □

Corollary 3.1 *Extensions of extensions are redundant.*

Proof. Suppose we have a critical cliff where the rule is an extension, for example $\Phi \mid (x + l_1) + l_2 \to x + r$ (the other case is similar):

$$s \xleftrightarrow[(x+y)+z=x+(y+z)]{\sigma,p \; AC1} t \xrightarrow[\Phi|(x+l_1)+l_2\to x+r]{\sigma,p.2} u$$

Then $\begin{cases} s|_p = (s_1 + s_2) + s_3 \\ t|_p = s_1 + (s_2 + s_3) \\ t|_{p.2} =_{\mathrm{AC1}} (x\sigma + l_1\sigma) + l_2\sigma \\ u|_p = s_1 + (x\sigma + r\sigma) \end{cases}$ But then $s_2 + s_3 =_{\mathrm{AC1}} (x\sigma + l_1\sigma) + l_2\sigma$, so $(s_1 + s_2) + s_3 =_{\mathrm{AC1}} ((s_1 + x\sigma) + l_1\sigma) + l_2\sigma$, so the critical cliff commutes in:

$$s = w[(s_1 + s_2) + s_3]_p \xleftrightarrow[AC1]{(\geq p)^*} w[((s_1 + x\sigma) + s_2) + s_3]_p \xrightarrow[\Phi|(X+l_1)+l_2\to X+r]{\sigma\cup\{X\mapsto s_1+x\sigma\},p}$$

$$w[(s_1 + x\sigma) + r\sigma]_p \xleftrightarrow[(X+Y)+Z=X+(Y+Z)]{\sigma',p \; AC1} w[s_1 + (x\sigma + r\sigma)]_p = u$$

□

Now the inference rule *Extend* can be replaced by

$$E; N; P \vdash E; N; P \cup \{\Phi \mid (x + l_1) + l_2 \to x + r, \Phi \mid l_1 + (l_2 + x) \to r + x\}$$

$$\text{if } \Phi \mid l_1 + l_2 \to r \in N \text{ and } x \notin \mathcal{V}ar(l_1 + l_2)$$

The rewriting rule *Extend* on proofs can be replaced then by the rules of the previous lemma and corollary. One can check that these new rules also decrease the complexity (in the lemma, rules are non-protected, and in the corollary they are protected).

4 Conclusion

Based on Baird, Peterson and Wilkerson's idea of constrained rewriting, we have given a complete set of rules for AC1-completion together with its correctness and completeness proofs. Moreover, we have shown how to construct rewrite orderings for checking termination of constrained AC1-rewriting. One may ask whether the techniques described here extend to other similar cases, for example to ACI-rewriting (I=idempotency). The answer is negative: in ACI-rewriting, not only the substitutions but also contexts must be constrained, a much harder task.

Acknowledgements This work was partly supported by the "Greco de programmation du CNRS" and the Basic Research Working Group COMPASS.

We thank Leo Bachmair who pointed out an error well hidden in a previous version of this work.

References

[1] L. Bachmair and N. Dershowitz. Completion for rewriting modulo a congruence. *Theoretical Comput. Sci.*, 67(2&3):173–201, Oct. 1989.

[2] L. Bachmair, N. Dershowitz, and J. Hsiang. Orderings for equational proofs. In *Proc. 1st IEEE Symp. Logic in Computer Science, Cambridge, Mass.*, pages 346–357, June 1986.

[3] L. Bachmair and D. A. Plaisted. Termination orderings for associative-commutative rewriting systems. *Journal of Symbolic Computation*, 1(4):329–349, Dec. 1985.

[4] T. Baird, G. Peterson, and R. Wilkerson. Complete sets of reductions modulo Associativity, Commutativity and Identity. In *Proc. 3rd Rewriting Techniques and Applications, Chapel Hill, LNCS 355*, pages 29–44. Springer-Verlag, Apr. 1989.

[5] A. Ben Cherifa and P. Lescanne. An actual implementation of a procedure that mechanically proves termination of rewriting systems based on inequalities between polynomial interpretations. In *Proc. 8th Int. Conf. on Automated Deduction, Oxford, England, LNCS 230*, pages 42–51. Springer-Verlag, July 1986.

[6] A. Boudet, J.-P. Jouannaud, and M. Schmidt-Schauß. Unification in free extensions of Boolean rings and Abelian groups. In *Proc. 3rd IEEE Symp. Logic in Computer Science, Edinburgh*, July 1988.

[7] H.-J. Bürckert, A. Herold, D. Kapur, J. H. Siekmann, M. E. Stickel, M. Tepp, and H. Zhang. Opening the AC-unification race. *Journal of Automated Reasoning*, 4(4):465–474, Dec. 1988.

[8] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1):69–115, Feb. 1987.

[9] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.

[10] J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM J. Comput.*, 15(4):1155–1194, 1986.

[11] C. Kirchner and H. Kirchner. Constraint equational reasoning. In *Proc. of the 20th Int. Symp. on Symbolic and Algebraic Computation, Portland, Oregon*, 1989.

[12] D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[13] D. S. Lankford and A. M. Ballantyne. Decision procedures for simple equational theories with commutative-associative axioms: Complete sets of commutative-associative reductions. Research Report Memo ATP-39, Department of Mathematics and Computer Science, University of Texas, Austin, Texas, USA, Aug. 1977.

[14] M. Livesey and J. H. Siekmann. Unification of bags and sets. Research report, Institut fur Informatik I, Universität Karlsruhe, West Germany, 1976.

[15] G. E. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *J. ACM*, 28(2):233–264, Apr. 1981.