Canonized Rewriting and Ground AC Completion Modulo Shostak Theories*

Sylvain Conchon Evelyne Contejean Mohamed Iguernelala

LRI, Univ Paris-Sud, CNRS, Orsay F-91405
INRIA Saclay – Ile-de-France, ProVal, Orsay, F-91893

Abstract. AC-completion efficiently handles equality modulo associative and commutative function symbols. When the input is ground, the procedure terminates and provides a decision algorithm for the word problem. In this paper, we present a modular extension of ground AC-completion for deciding formulas in the combination of the theory of equality with user-defined AC symbols, uninterpreted symbols and an arbitrary signature disjoint Shostak theory X. Our algorithm, called AC(X), is obtained by augmenting in a modular way ground AC-completion with the canonizer and solver present for the theory X. This integration rests on canonized rewriting, a new relation reminiscent to normalized rewriting, which integrates canonizers in rewriting steps. AC(X) is proved sound, complete and terminating, and is implemented to extend the core of the Alt-Ergo theorem prover.

Key words: decision procedure, associativity and commutativity, rewriting, AC-completion, SMT solvers, Shostak's algorithm.

1 Introduction

Many mathematical operators occurring in automated reasoning such as union and intersection of sets, or boolean and arithmetic operators, satisfy the following associativity and commutativity (AC) axioms

$$\forall x.\forall y.\forall z.\ u(x,u(y,z)) = u(u(x,y),z) \qquad \text{(A)}$$
$$\forall x.\forall y.\ u(x,y) = u(y,x) \qquad \text{(C)}$$

Automated AC reasoning is known to be difficult. Indeed, the mere addition of these two axioms to a prover will usually glut it with plenty of useless equalities which will strongly impact its performances[1]. In order to avoid this drawback, built-in procedures have been designed to efficiently handle AC symbols. For instance, SMT-solvers incorporate dedicated decision procedures for some *specific* AC symbols such as arithmetic or boolean operators. On the contrary, algorithms found in resolution-based provers such as AC-completion allow a powerful *generic* treatment of user-defined AC symbols.

* Work partially supported by the French ANR project ANR-08-005 Decert

[1] Given a term t of the form $u(c_1, u(c_2, \ldots, u(c_n, c_{n+1})\ldots)$, the axiomatic approach may have to explicitly handle the $(2n)!/n!$ terms equivalent to t.

Given a finite word problem $\bigwedge_{i\in I} s_i = t_i \vdash s = t$ where the function symbols are either uninterpreted or AC, AC-completion attempts to transform the conjunction $\bigwedge_{i\in I} s_i = t_i$ into a finitely terminating, confluent term rewriting system R whose reductions preserve identity. The rewriting system R serves as a decision procedure for validating $s = t$ modulo AC: the equation holds if and only if the normal forms of s and t w.r.t R are equal modulo AC. Furthermore, when its input contains only ground equations, AC-completion terminates and outputs a convergent rewriting system [16].

Unfortunately, AC reasoning is only a part of the automated deduction problem, and what we really need is to decide formulas combining AC symbols and other theories. For instance, in practice, we are interested in deciding finite ground word problems which contain a mixture of uninterpreted, interpreted and AC function symbols, as in the following assertion

$$\begin{array}{l} u(a, c_2 - c_1) \approx a \wedge u(e_1, e_2) - f(b) \approx u(d, d) \wedge \\ d \approx c_1 + 1 \wedge e_2 \approx b \wedge u(b, e_1) \approx f(e_2) \wedge c_2 \approx 2 * c_1 + 1 \end{array} \vdash a \approx u(a, 0)$$

where u is an AC symbol, $+$, $-$, $*$ and the numerals are from the theory of linear arithmetic, f is an uninterpreted function symbol and the other symbols are uninterpreted constants. A combination of AC reasoning with linear arithmetic and the free theory $\mathcal{E}$ of equality is necessary to prove this formula. Linear arithmetic is used to show that $c_2 - c_1 = c_1 + 1$ so that (i) $u(a, c_1 + 1) = a$ follows by congruence. Independently, $e_2 = b$ and $d = c_1 + 1$ imply (ii) $u(c_1 + 1, c_1 + 1) = 0$ by congruence, linear arithmetic and commutativity of u. AC reasoning can finally be used to conclude that (i) and (ii) imply that $u(a, c_1 + 1, c_1 + 1)$ is equal to both a and $u(a, 0)$.

There are two main methods for combining decision procedures for disjoint theories. First, the Nelson-Oppen approach [18] is based on a variable abstraction mechanism and the exchange of equalities between shared variables. Second, the Shostak's algorithm [21] extends a congruence closure procedure with theories equipped with canonizers and solvers, *i.e.* procedures that compute canonical forms of terms and solve equations, respectively. While ground AC-completion can be easily combined with other decision procedures by the Nelson-Oppen method, it cannot be directly integrated in the Shostak's framework since it actually does not provide a solver for the AC theory.

In this paper, we investigate the integration of Shostak theories in ground AC-completion. We first introduce a new notion of rewriting called *canonized* rewriting which adapts normalized rewriting to cope with canonization. Then, we present a modular extension of ground AC-completion for deciding formulas in the combination of the theory of equality with user-defined AC symbols, uninterpreted symbols and an arbitrary signature disjoint Shostak theory X. The main ideas of our integration are to substitute standard rewriting by canonized rewriting, using a global canonizer for AC and X, and to replace the equation orientation mechanism found in ground AC-completion with the solver for X.

AC-completion has been studied for a long time in the rewriting community [15, 20]. A generic framework for combining completion with a generic built-in equational theory E has been proposed in [10]. Normalized completion [17] is

designed to use a modified rewriting relation when the theory E is equivalent to the union of the AC theory and a convergent rewriting system $\mathcal{S}$. In this setting, rewriting steps are only performed on $\mathcal{S}$-normalized terms. $\mathsf{AC(X)}$ can be seen as an adaptation of ground normalized completion to efficiently handle the theory E when it is equivalent to the union of the AC theory and a Shostak theory X. In particular, $\mathcal{S}$-normalization is replaced by the application of the canonizer of X. This modular integration of X allows us to reuse proof techniques of ground AC-completion [16] to show the correctness of $\mathsf{AC(X)}$.

Kapur [11] used ground completion to demystify Shostak's congruence closure algorithm and Bachmair *et al.* [3] compared its strategy with other ones into an abstract congruence closure framework. While the latter approach can also handle AC symbols, none of these works formalized the integration of Shostak theories into (AC) ground completion.

Outline. Section 2 recalls standard ground AC completion. Section 3 is devoted to Shostak theories and global canonization. Section 4 presents the $\mathsf{AC(X)}$ algorithm and illustrates its use through an example. The correctness of $\mathsf{AC(X)}$ is sketched in Section 5 and experimental results are presented in Section 6. Conclusion and future works are presented in Section 7.

2 Ground AC-Completion

In this section, we first briefly recall the usual notations and definitions of [1, 7] for term rewriting modulo AC. Then, we give the usual set of inference rules for ground AC-completion procedure and we illustrate its use through an example.

Terms are built from a signature $\Sigma = \Sigma_{AC} \uplus \Sigma_{\mathcal{E}}$ of AC and uninterpreted symbols, and a set of variables $\mathcal{X}$ yielding the term algebra $\mathcal{T}_{\Sigma}(\mathcal{X})$. The range of letters $a \ldots f$ denotes uninterpreted symbols, u denotes an AC function symbol, s, t, l, r denote terms, and x, y, z denote variables. Viewing terms as trees, subterms within a term s are identified by their positions. Given a position p, $s|_p$ denotes the subterm of s at position p, and $s[r]_p$ the term obtained by replacement of $s|_p$ by the term r. We will also use the notation $s(p)$ to denote the symbol at position p in the tree, and the root position is denoted by Λ. Given a subset Σ' of Σ, a subterm $t|_p$ of t is a Σ'-alien of t if $t(p) \notin \Sigma'$ and p is minimal *w.r.t* the prefix word ordering[2]. We write $\mathcal{A}_{\Sigma'}(t)$ the multiset of Σ'-aliens of t.

A substitution is a partial mapping from variables to terms. Substitutions are extended to a total mapping from terms to terms in the usual way. We write $t\sigma$ for the application of a substitution σ to a term t. A well-founded quasi-ordering [6] on terms is a reduction quasi-ordering if $s \preceq t$ implies $s\sigma \preceq t\sigma$ and $l[s]_p \preceq l[t]_p$, for any substitution σ, term l and position p. A quasi-ordering $\preceq$ defines an equivalence relation $\simeq$ as $\preceq \cap \succeq$ and a partial ordering $\prec$ as $\preceq \cap \not\simeq$.

An equation is an unordered pair of terms, written $s \approx t$. The variables contained in an equation, if any, are understood as being universally quantified. Given a set of equations E, the equational theory of E, written $=_E$, is the set of

[2] Notice that according to this definition, a variable may be a Σ'-alien.

equations that can be obtained by reflexivity, symmetry, transitivity, congruence and instances of equations in E[3]. The word problem for E consists in determining if, given two ground terms s and t, the equation $s \approx t$ is in $=_E$, denoted by $s =_E t$. The word problem for E is ground when E contains only ground equations. An equational theory $=_E$ is said to be *inconsistent* when $s =_E t$, for *any* s and t.

A rewriting rule is an oriented equation, usually denoted by $l \to r$. A term s rewrites to a term t at position p by the rule $l \to r$, denoted by $s \to^p_{l \to r} t$, iff there exists a substitution σ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. A rewriting system R is a set of rules. We write $s \to_R t$ whenever there exists a rule $l \to r$ of R such that s rewrites to t by $l \to r$ at some position. A normal form of a term s w.r.t to R is a term t such that $s \to^*_R t$ and t cannot be rewritten by R. The system R is said to be *convergent* whenever any term s has a unique normal form, denoted $s{\downarrow_R}$, and does not admit any infinite reduction. Completion [12] aims at converting a set E of equations into a convergent rewriting system R such that the sets $=_E$ and $\{s \approx t \mid s{\downarrow_R} = t{\downarrow_R}\}$ coincide. Given a suitable reduction ordering on terms, it has been proved that completion terminates when E is ground [14].

Rewriting modulo AC. Let $=_{AC}$ be the equational theory obtained from the set:

$$AC = \bigcup_{u \in \Sigma_{AC}} \{\, u(x,y) \approx u(y,x),\ u(x,u(y,z)) \approx u(u(x,y),z) \,\}$$

In general, given a set E of equations, it has been shown that no suitable reduction ordering allows completion to produce a convergent rewriting system for $E \cup AC$. When E is ground, an alternative consists in in-lining AC reasoning both in the notion of rewriting step and in the completion procedure.

Rewriting modulo AC is directly related to the notion of matching modulo AC as shown by the following example. Given a rule $u(a, u(b, c))) \to t$, we would like the following reductions to be possible:

$$(1)\ f(u(c,u(b,a)),d) \to f(t,d) \qquad (2)\ u(a,u(c,u(d,b))) \to u(t,d)$$

Associativity and commutativity of u are needed in (1) for the subterm $u(c,u(b,a))$ to match the term $u(a,u(b,c))$, and in (2) for the term $u(a,u(c,u(d,b)))$ to be seen as $u(u(a,u(b,c)),d)$, so that the rule can be applied. More formally, this leads to the following definition.

Definition 1 (Ground rewriting modulo AC). A term s rewrites to a term t modulo AC at position p by the rule $l \to r$, denoted by $s \to^p_{AC \setminus l \to r} t$, iff (1) $s|_p =_{AC} l$ and $t = s[r]_p$ or (2) $l(\Lambda) = u$ and there exists a term s' such that $s|_p =_{AC} u(l,s')$ and $t = s[u(r,s')]_p$

In order to produce a convergent rewriting system, ground AC-completion requires a well-founded reduction quasi-ordering $\preceq$ total on ground terms with an underlying equivalence relation which coincides with $=_{AC}$. Such an ordering will be called a total ground AC-reduction ordering.

[3] The equational theory of the free theory of equality $\mathcal{E}$, defined by the empty set of equations, is simply denoted $=$.

The inference rules for ground AC-completion are given in Figure 1. The rules describe the evolution of the state of a procedure, represented as a configuration $\langle\ E \mid R\ \rangle$, where E is a set of ground equations and R a ground set of rewriting rules. The initial state is $\langle\ E_0 \mid \emptyset\ \rangle$ where E_0 is a given set of ground equations. **Tri**vial removes an equation $u \approx v$ from E when u and v are equal modulo AC. **Ori**ent turns an equation into a rewriting rule according to a given total ground AC-reduction ordering $\preceq$. R is used to rewrite either side of an equation (**Sim**plify), and to reduce right hand side of rewriting rules (**Com**pose). Given a rule $l \to r$, **Col**lapse either reduces l at an inner position, or replaces l by a term smaller than r. In both cases, the reduction of l to l' may influence the orientation of the rule $l' \to r$ which is added to E as an equation in order to be re-oriented. Finally, **Ded**uce adds equational consequences of rewriting rules to E. For instance, if R contains two rules of the form $u(a,b) \to s$ and $u(a,c) \to t$, then the term $u(a,u(b,c))$ can either be reduced to $u(s,c)$ or to the term $u(t,b)$. The equation $u(s,c) \approx u(t,b)$, called *critical pair*, is thus necessary for ensuring convergence of R. Critical pairs of a set of rules are computed by the following function (a^μ stands for the maximal term w.r.t. size enjoying the assertion):

$$\mathtt{headCP}(R) = \left\{ u(b,r') \approx u(b',r) \;\middle|\; \begin{array}{l} l \to r \in R,\ \ l' \to r' \in R \\ \exists\, a^\mu :\ l =_{AC} u(a^\mu, b)\ \wedge\ l' =_{AC} u(a^\mu, b') \end{array} \right\}$$

$$\textbf{Tri}\text{vial}\ \frac{\langle\ E \cup \{\, s \approx t \,\}\ \mid\ R\ \rangle}{\langle\ E\ \mid\ R\ \rangle}\ s =_{AC} t$$

$$\textbf{Ori}\text{ent}\ \frac{\langle\ E \cup \{\, s \approx t \,\}\ \mid\ R\ \rangle}{\langle\ E\ \mid\ R \cup \{\, s \to t \,\}\ \rangle}\ t \prec s$$

$$\textbf{Sim}\text{plify}\ \frac{\langle\ E \cup \{\, s \approx t \,\}\ \mid\ R\ \rangle}{\langle\ E \cup \{\, s' \approx t \,\}\ \mid\ R\ \rangle}\ s \to_{AC \setminus R} s'$$

$$\textbf{Com}\text{pose}\ \frac{\langle\ E\ \mid\ R \cup \{\, l \to r \,\}\ \rangle}{\langle\ E\ \mid\ R \cup \{\, l \to r' \,\}\ \rangle}\ r \to_{AC \setminus R} r'$$

$$\textbf{Col}\text{lapse}\ \frac{\langle\ E\ \mid\ R \cup \{\, g \to d,\, l \to r \,\}\ \rangle}{\langle\ E \cup \{\, l' \approx r \,\}\ \mid\ R \cup \{\, g \to d \,\}\ \rangle}\ \begin{cases} l \to_{AC \setminus g \to d} l' \\ g \prec l\ \vee\ (g \simeq l \wedge d \prec r) \end{cases}$$

$$\textbf{Ded}\text{uce}\ \frac{\langle\ E\ \mid\ R\ \rangle}{\langle\ E \cup \{\, s \approx t \,\}\ \mid\ R\ \rangle}\ s \approx t \in \mathtt{headCP}(R)$$

Fig. 1. Inference rules for ground AC-completion

Example. To get a flavor of ground AC-completion, consider a modified version of the assertion given in the introduction, where the arithmetic part has been removed (and uninterpreted constant symbols renamed for the sake of simplicity)

$$u(a_1,a_4) \approx a_1, u(a_3,a_6) \approx u(a_5,a_5), a_5 \approx a_4, a_6 \approx a_2 \vdash a_1 \approx u(a_1,u(a_6,a_3))$$

The precedence $a_1 \prec_p \cdots \prec_p a_6 \prec_p u$ defines an AC-RPO ordering on terms [19] which is suitable for ground AC-completion. The table in Figure 2 shows the application steps of the rules given in Figure 1 from an initial configuration $\langle \{u(a_1, a_4) \approx a_1, u(a_3, a_6) \approx u(a_5, a_5), a_5 \approx a_4, a_6 \approx a_2\} \mid \emptyset \rangle$ to a final configuration $\langle \emptyset \mid R_f \rangle$, where R_f is the set of rewriting rules $\{1, 3, 5, 7, 10\}$. It can be checked that $a_1{\downarrow_{R_f}}$ and $u(a_1, u(a_6, a_3)){\downarrow_{R_f}}$ are identical.

1	$\mathbf{u(a_1, a_4) \to a_1}$	**Ori** $u(a_1, a_4) \approx a_1$
2	$u(a_3, a_6) \to u(a_5, a_5)$	**Ori** $u(a_3, a_6) \approx u(a_5, a_5)$
3	$\mathbf{a_5 \to a_4}$	**Ori** $a_5 \approx a_4$
4	$u(a_3, a_6) \to u(a_4, a_4)$	**Com** 2 and 3
5	$\mathbf{a_6 \to a_2}$	**Ori** $a_6 \approx a_2$
6	$u(a_3, a_2) \approx u(a_4, a_4)$	**Col** 4 and 5
7	$\mathbf{u(a_4, a_4) \to u(a_3, a_2)}$	**Ori** 6
8	$u(a_1, a_4) \approx u(a_1, u(a_3, a_2))$	**Ded** from 1 and 7
9	$a_1 \approx u(a_1, u(a_3, a_2))$	**Sim** 8 by 1
10	$\mathbf{u(a_1, u(a_3, a_2)) \to a_1}$	**Ori** 9

Fig. 2. Ground AC-completion example

3 Shostak Theories and Global Canonization

In this section, we recall the notions of canonizers and solvers underlying Shostak theories and show how to obtain a global canonizer for the combination of the theories $\mathcal{E}$ and AC with an arbitrary signature disjoint Shostak theory X.

From now on, we assume given a theory X with a signature Σ_{X}. A canonizer for X is a function $\mathtt{can}_{\mathsf{X}}$ that computes a unique normal form for every term such that $s =_{\mathsf{X}} t$ iff $\mathtt{can}_{\mathsf{X}}(s) = \mathtt{can}_{\mathsf{X}}(t)$. A solver for X is a function $\mathtt{solve}_{\mathsf{X}}$ that solves equations between Σ_{X}-terms. Given an equation $s \approx t$, $\mathtt{solve}_{\mathsf{X}}(s \approx t)$ either returns a special value $\bot$ when $s \approx t \cup \mathsf{X}$ is inconsistent, or an equivalent substitution. A Shostak theory X is a theory with a canonizer and a solver which fulfill some standard properties given for instance in [13].

Our combination technique is based on the integration of a Shostak theory X in ground AC-completion. From now on, we assume that terms are built from a signature Σ defined as the union of the disjoint signatures Σ_{AC}, $\Sigma_{\mathcal{E}}$ and Σ_{X}. We also assume a total ground AC-reduction ordering $\preceq$ defined on $\mathcal{T}_\Sigma(\mathcal{X})$ used later on for completion. The combination mechanism requires defining both a global canonizer for the union of $\mathcal{E}$, AC and X, and a wrapper of $\mathtt{solve}_{\mathsf{X}}$ to handle heterogeneous equations. These definitions make use of a global one-to-one mapping $\alpha : \mathcal{T}_\Sigma \to \mathcal{X}$ (and its inverse mapping ρ) and are based on a variable abstraction mechanism which computes the *pure* Σ_{X}-part $[\![t]\!]$ of a heterogeneous term t as follows:

$$[\![t]\!] = f([\![\boldsymbol{s}]\!]) \text{ when } t = f(\boldsymbol{s}) \text{ and } f \in \Sigma_{\mathsf{X}} \qquad \text{and} \qquad [\![t]\!] = \alpha(t) \text{ otherwise}$$

The canonizer for AC defined in [9] is based on flattening and sorting techniques which simulate associativity and commutativity, respectively. For instance, the term $u(u(u'(c,b),b),c)$ is first flattened to $u(u'(c,b),b,c)$ and then sorted[4] to get the term $u(b,c,u'(c,b))$. It has been formally proved that this canonizer solves the word problem for AC [5]. However, this definition implies a modification of the signature Σ_{AC} where arity of AC symbols becomes variadic. Using such canonizer would impact the definition of AC-rewriting given in Section 2. In order to avoid such modification we shall define an equivalent canonizer that builds degenerate trees instead of flattened terms. For instance, we would expect the normal form of $u(u(u'(c,b),b),c)$ to be $u(b,u(c,u'(c,b)))$. Given a signature Σ which contains Σ_{AC} and any total ordering $\trianglelefteq$ on terms, we define $\mathtt{can}_{AC}$ by:

$$\begin{array}{lll}
\mathtt{can}_{AC}(x) & = x & \text{when } x \in \mathcal{X} \\
\mathtt{can}_{AC}(f(\boldsymbol{v})) & = f(\mathtt{can}_{AC}(\boldsymbol{v})) & \text{when } f \notin \Sigma_{AC} \\
\mathtt{can}_{AC}(u(t_1,t_2)) & = u(s_1,u(s_2,\ldots,u(s_{n-1},s_n)\ldots)) & \\
& \quad \text{where } t'_i = \mathtt{can}_{AC}(t_i) \text{ for } i \in [1,2] & \\
& \quad \text{and } \{\!\{s_1,\ldots,s_n\}\!\} = \mathcal{A}_{\{u\}}(t'_1) \cup \mathcal{A}_{\{u\}}(t'_2) & \\
& \quad \text{and } s_i \trianglelefteq s_{i+1} \text{ for } i \in [1,n-1] & \text{when } u \in \Sigma_{\mathsf{AC}}
\end{array}$$

We can easily show that $\mathtt{can}_{AC}$ enjoys the standard properties required for a canonizer. The proof that $\mathtt{can}_{AC}$ solves the word problem for AC follows directly from the one given in [5].

Using the technique described in [13], we define our global canonizer $\mathtt{can}$ which combines $\mathtt{can}_{\mathsf{X}}$ with $\mathtt{can}_{AC}$ as follows:

$$\begin{array}{lll}
\mathtt{can}(x) & = x & \text{when } x \in \mathcal{X} \\
\mathtt{can}(f(\boldsymbol{v})) & = f(\mathtt{can}(\boldsymbol{v})) & \text{when } f \in \Sigma_{\mathcal{E}} \\
\mathtt{can}(u(s,t)) & = \mathtt{can}_{AC}(u(\mathtt{can}(s),\mathtt{can}(t))) & \text{when } u \in \Sigma_{\mathsf{AC}} \\
\mathtt{can}(f_{\mathsf{x}}(\boldsymbol{v})) & = \mathtt{can}_{\mathsf{X}}(f_{\mathsf{x}}([\![\mathtt{can}(\boldsymbol{v})]\!]))\rho & \text{when } f_{\mathsf{x}} \in \Sigma_{\mathsf{X}}
\end{array}$$

Again, the proofs that $\mathtt{can}$ solves the word problem for the union $\mathcal{E}$, AC and X and enjoys the standard properties required for a canonizer are similar to those given in [13]. The only difference is that $\mathtt{can}_{AC}$ directly works on the signature Σ, which avoids the use of a variable abstraction step when canonizing a mixed term of the form $u(t_1,t_2)$ such that $u \in \Sigma_{AC}$.

Using the same mappings α, ρ and the abstraction function, the wrapper $\mathtt{solve}$ can be easily defined by:

$$\mathtt{solve}(s \approx t) = \begin{cases} \bot & \text{if } \mathtt{solve}_{\mathsf{X}}([\![s]\!] \approx [\![t]\!]) = \bot \\ \{\ x_i\rho \to t_i\rho\ \} & \text{if } \mathtt{solve}_{\mathsf{X}}([\![s]\!] \approx [\![t]\!]) = \{x_i \approx t_i\} \end{cases}$$

In order to ensure termination of $\mathsf{AC(X)}$, the global canonizer and the wrapper must be compatible with the ordering $\preceq$ used by AC-completion, that is:

Lemma 1. $\forall t \in \mathcal{T}_\Sigma,\ \mathtt{can}(t) \preceq t$
$\forall s,t \in \mathcal{T}_\Sigma,\ \text{ if } \mathtt{solve}(s \approx t) = \bigcup\{p_i \to v_i\} \text{ then } v_i \prec p_i$

[4] For instance, using the AC-RPO ordering based on the precedence $b \prec_p c \prec_p u'$.

We can prove that the above properties hold when the theory X enjoys the following local compatibility properties:

Axiom 1. $\forall t \in \mathcal{T}_\Sigma,\ \mathtt{can}_\mathsf{X}([\![t]\!]) \preceq [\![t]\!]$
$\forall s,t \in \mathcal{T}_\Sigma,\ \text{if } \mathtt{solve}_\mathsf{X}([\![s]\!] \approx [\![t]\!]) = \bigcup\{x_i \approx t_i\} \text{ then } t_i\rho \prec x_i\rho$

To fulfil this axiom, AC-reduction ordering can be chosen as an AC-RPO ordering [19] based on a precedence relation $\prec_p$ such that $\Sigma_\mathsf{X} \prec_p \Sigma_\mathcal{E} \cup \Sigma_{AC}$. From now on, we assume that X is locally compatible with $\preceq$.

Example. To solve the equation $u(a,b) + a \approx 0$, we use the abstraction $\alpha = \{u(a,b) \mapsto x,\ a \mapsto y\}$ and call $\mathtt{solve}_\mathsf{X}$ on $x + y \approx 0$. Since $a \prec u(a,b)$, the only solution which fulfills the axiom above is $\{x \approx -y\}$. We apply ρ and get the set $\{u(a,b) \to -a\}$ of rewriting rules.

4 Ground AC-Completion Modulo X

In this section, we present the AC(X) algorithm which extends the ground AC-completion procedure given in Section 2. For that purpose, we first adapt the notion of ground AC-rewriting to cope with canonizers. Then, we show how to refine the inference rules given in Figure 1 to reason modulo the equational theory induced by a set E of ground equations and the theories $\mathcal{E}$, AC and X.

4.1 Canonized Rewriting

From rewriting point of view, a canonizer behaves like a convergent rewriting system: it gives an effective way of computing normal forms. Thus, a natural way for integrating `can` in ground AC-completion is to extend normalized rewriting [17].

Definition 2. Let `can` be a canonizer. A term s `can`-rewrites to a term t at position p by the rule $l \to r$, denoted by $s \rightsquigarrow^p_{l\to r} t$, iff

$$s \to^p_{AC\setminus l\to r} t' \qquad \text{and} \qquad \mathtt{can}(t') = t$$

Example. Using the usual canonizer $\mathtt{can}_\mathcal{A}$ for linear arithmetic and the rule $\gamma : u(a,b) \to a$, the term $f(a + 2 * u(b,a))$ $\mathtt{can}_\mathcal{A}$-rewrites to $f(3 * a)$ by $\rightsquigarrow_\gamma$ as follows: $f(a+2*u(b,a)) \to_{AC\setminus\gamma} f(a+2*a)$ and $\mathtt{can}_\mathcal{A}(f(a+2*a)) = f(3*a)$.

Lemma 2. $\forall\ s,\ t.\quad s \rightsquigarrow_{l\to r} t \implies s =_{AC,\mathsf{X},l\approx r} t$

4.2 The AC(X) Algorithm

The first step of our combination technique consists in replacing the rewriting relation found in completion by canonized rewriting. This leads to the rules of AC(X) given in Figure 3. The state of the procedure is a pair $\langle\ E \mid R\ \rangle$ of equations and rewriting rules. The initial configuration is $\langle\ E_0 \mid \emptyset\ \rangle$ where E_0 is supposed to be a set of equations between canonized terms. Since AC(X)'s rules only involve

canonized rewriting, the algorithm maintains the invariant that terms occurring in E and R are in canonical forms. **Tri**vial thus removes an equation $u \approx v$ from E when u and v are syntactically equal. A new rule **Bot**tom is used to detect inconsistent equations. Similarly to normalized completion, integrating the global canonizer `can` in rewriting is not enough to fully extend ground AC-completion with the theory X: in both cases the orientation mechanism has to be adapted . Therefore, the second step consists in integrating the wrapper `solve` in the **Ori**ent rule. The other rules are much similar to those of ground AC-completion except that they use the relation $\rightsquigarrow_R$ instead of $\rightarrow_{AC\setminus R}$.

$$\textbf{Tri}\text{vial } \frac{\langle\, E \cup \{\, s \approx t \,\} \mid R \,\rangle}{\langle\, E \mid R \,\rangle}\ s = t \qquad \textbf{Bot}\text{tom } \frac{\langle\, E \cup \{\, s \approx t \,\} \mid R \,\rangle}{\bot}\ \mathtt{solve}(s,t) = \bot$$

$$\textbf{Ori}\text{ent } \frac{\langle\, E \cup \{\, s \approx t \,\} \mid R \,\rangle}{\langle\, E \mid R \cup \mathtt{solve}(s,t) \,\rangle}\ \mathtt{solve}(s,t) \neq \bot$$

$$\textbf{Sim}\text{plify } \frac{\langle\, E \cup \{\, s \approx t \,\} \mid R \,\rangle}{\langle\, E \cup \{\, s' \approx t \,\} \mid R \,\rangle}\ s \rightsquigarrow_R s' \qquad \textbf{Com}\text{pose } \frac{\langle\, E \mid R \cup \{\, l \to r \,\} \,\rangle}{\langle\, E \mid R \cup \{\, l \to r' \,\} \,\rangle}\ r \rightsquigarrow_R r'$$

$$\textbf{Col}\text{lapse } \frac{\langle\, E \mid R \cup \{\, g \to d, l \to r \,\} \,\rangle}{\langle\, E \cup \{\, l' \approx r \,\} \mid R \cup \{\, g \to d \,\} \,\rangle}\ \begin{cases} l \rightsquigarrow_{g \to d} l' \\ g \prec l \ \vee\ (g \simeq l \wedge d \prec r) \end{cases}$$

$$\textbf{Ded}\text{uce } \frac{\langle\, E \mid R \,\rangle}{\langle\, E \cup \{\, s \approx t \,\} \mid R \,\rangle}\ s \approx t \in \mathtt{headCP}(R)$$

Fig. 3. Inference rules for ground AC-completion modulo X

Example. We illustrate $\mathsf{AC(X)}$ on the example given in the introduction:

$$\begin{array}{l} u(a, c_2 - c_1) \approx a \wedge u(e_1, e_2) - f(b) \approx u(d,d) \wedge \\ d \approx c_1 + 1 \wedge e_2 \approx b \wedge u(b, e_1) \approx f(e_2) \wedge c_2 \approx 2 * c_1 + 1 \end{array} \vdash a \approx u(a, 0)$$

The table given in Figure 4 shows the application of the rules of $\mathsf{AC(X)}$ on the example when X is instantiated by linear arithmetic. We use an AC-RPO ordering based on the precedence $1 \prec_p 2 \prec_p a \prec_p b \prec_p c_1 \prec_p c_2 \prec_p d \prec_p e_1 \prec_p e_2 \prec_p f \prec_p u$. The procedure terminates and produces a convergent rewriting system $R_f = \{3, 5, 9, 10, 11, 13, 16\}$. Using R_f, we can check that a and $u(a, 0)$ `can`-rewrite to the same normal form.

5 Correctness

As usual, in order to enforce correctness, we cannot use any (unfair) strategy. We say that a strategy is *strongly fair* when no possible application of an inference rule is infinitely delayed and **Ori**ent is only applied over fully reduced terms.

1	$u(a, c_2 - c_1) \to a$	**Ori** $u(a, c_2 - c_1) \approx a$
2	$u(e_1, e_2) \to u(d, d) + f(b)$	**Ori** $u(e_1, e_2) - f(b) \approx u(d, d)$
3	$\mathbf{d \to c_1 + 1}$	**Ori** $d \approx c_1 + 1$
4	$u(e_1, e_2) \to u(c_1 + 1, c_1 + 1) + f(b)$	**Com** 2 and 3
5	$\mathbf{e_2 \to b}$	**Ori** $e_2 \approx b$
6	$u(b, e_1) \approx u(c_1 + 1, c_1 + 1) + f(b)$	**Col** 4 and 5
7	$u(b, e_1) \to u(c_1 + 1, c_1 + 1) + f(b)$	**Ori** $u(b, e_1) \approx u(c_1 + 1, c_1 + 1) + f(b)$
8	$u(c_1 + 1, c_1 + 1) + f(b) \approx f(b)$	**Sim** $u(b, e_1) \approx f(e_2)$ by 5 and 7
9	$\mathbf{u(c_1 + 1, c_1 + 1) \to 0}$	**Ori** $u(c_1 + 1, c_1 + 1) + f(b) \approx f(b)$
10	$\mathbf{u(b, e_1) \to f(b)}$	**Com** 7 and 9
11	$\mathbf{c_2 \to 2 * c_1 + 1}$	**Ori** $c_2 \approx 2 * c_1 + 1$
12	$u(a, c_1 + 1) \approx a$	**Col** 1 and 11
13	$\mathbf{u(a, c_1 + 1) \to a}$	**Ori** $u(a, c_1 + 1) \approx a$
14	$u(0, a) \approx u(a, c_1 + 1)$	**Ded** from 9 and 13
15	$u(0, a) \approx a$	**Sim** 14 by 13
16	$\mathbf{u(0, a) \to a}$	**Ori** 15

Fig. 4. AC(X) on the running example.

Theorem 1. Given a set E of ground equations, the application of the rules of AC(X) under a strongly fair strategy terminates and either produces $\bot$ when $E \cup AC \cup \mathsf{X}$ is inconsistent, or yields a final configuration $\langle\ \emptyset \mid R\ \rangle$ such that: $\forall s, t \in \mathcal{T}_\Sigma.\ s =_{E,AC,X} t \iff \mathtt{can}(s)\!\downarrow_R = \mathtt{can}(t)\!\downarrow_R$

The proof [5] is based on three intermediate theorems, stating respectively soundness, completeness and termination. In the following, we shall consider a fixed run of the completion procedure,

$$\langle\ E_0 \mid \emptyset\ \rangle \to \langle\ E_1 \mid R_1\ \rangle \to \ldots \to \langle\ E_n \mid R_n\ \rangle \to \langle\ E_{n+1} \mid R_{n+1}\ \rangle \to \ldots$$

starting from the initial configuration $\langle\ E_0 \mid \emptyset\ \rangle$. We denote R_∞ (resp. E_∞) the set of all encountered rules $\bigcup_n R_n$ (resp. equations $\bigcup_n E_n$) and $\mathcal{R}_\omega$ (resp. E_ω) the set of persistent rules $\bigcup_n \bigcap_{i \geq n} R_i$ (resp. equations $\bigcup_n \bigcap_{i \geq n} E_i$).

5.1 Soundness

Soundness is ensured by the following invariant:

Theorem 2. For any configuration $\langle\ E_n \mid R_n\ \rangle$ reachable from $\langle\ E_0 \mid \emptyset\ \rangle$,

$$\forall\ s,\ t, \quad (s, t) \in E_n \cup R_n \implies s =_{AC,\mathsf{X},E_0} t$$

Proof. The invariant obviously holds for the initial configuration and is preserved by all the inference rules. The rules **Sim**plify, **Com**pose, **Col**lapse and **Ded**uce preserve the invariant since for any rule $l \to r$, if $l =_{AC,\mathsf{X},E_0} r$, for any term s rewritten by $\rightsquigarrow_{l \to r}$ into t, then $s =_{AC,\mathsf{X},E_0} t$. If **Ori**ent is used to turn an equation $s \approx t$ into a set of rules $\{p_i \to v_i\}$, by definition of `solve`, $p_i = x_i\rho$ and $v_i = t_i\rho$, where $\mathtt{solve}_\mathsf{X}([\![s]\!] \approx [\![t]\!]) = \{x_i \approx t_i\}$. By soundness of $\mathtt{solve}_\mathsf{X}$ $x_i =_{\mathsf{X},[\![s]\!] \approx [\![t]\!]} t_i$. An equational proof of $x_i =_{\mathsf{X},[\![s]\!] \approx [\![t]\!]} t_i$ can be instantiated by ρ, yielding an equational proof $p_i =_{\mathsf{X},s \approx t} v_i$. Since by induction $s =_{AC,\mathsf{X},E_0} t$ holds, we get $p_i =_{AC,\mathsf{X},E_0} v_i$.

[5] All the details of the proof can be found in a research report [4]

In the rest of this section, we assume that the strategy is strongly fair. This implies in particular that $\mathtt{headCP}(R_\omega) \subseteq E_\infty$, $E_\omega = \emptyset$ and R_ω is inter-reduced, that is none of its rules can be collapsed or composed by another one. We also assume that $\bot$ is not encountered, otherwise, termination is obvious.

5.2 Completeness

Completeness is established by using a variant of the technique introduced by Bachmair *et al.* in [2] for proving completeness of completion. It transforms a proof between two terms which is not under a suitable form into a smaller one, and the smallest proofs are the desired ones. The proofs we are considering are made of elementary steps, either equational steps, with AC, X and E_∞, or rewriting steps, with R_∞ and the additional (possibly infinite) rules $R_{\mathtt{can}} = \{t \to \mathtt{can}(t) \mid \mathtt{can}(t) \neq t\}$. Rewriting steps with R_∞ can be either $\rightsquigarrow_{R_\infty}$ or $\to_{R_\infty}$[6].

The measure of a proof is the multiset of the elementary measures of its elementary steps. The measure of an elementary step takes into account the number of terms which are in a canonical form in an elementary proof: the canonical weight of a term t, $w_{\mathtt{can}}(t)$ is equal to 0 if $\mathtt{can}(t) =_{AC} t$ and to 1 otherwise. Notice that if $w_{\mathtt{can}}(t) = 1$, then $\mathtt{can}(t) \prec t$, and if $w_{\mathtt{can}}(t) = 0$, then $\mathtt{can}(t) \simeq t$. The measure of an elementary step between t_1 and t_2 performed thanks to:

- an equation is equal to $(\{t_1, t_2\}, \_, \_, \_, \_)$
- a rule $l \to r \in R_\infty$ is equal to $(\{t_1\}, 1, w_{\mathtt{can}}(t_1) + w_{\mathtt{can}}(t_2), l, r)$ if $t_1 \rightsquigarrow_{l\to r} t_2$ or $t_1 \to_{l\to r} t_2$.
- a rule of $R_{\mathtt{can}}$ is equal to $(\{t_1\}, 0, w_{\mathtt{can}}(t_1) + w_{\mathtt{can}}(t_2), t_1, t_2)$ if $t_1 \to_{R_{\mathtt{can}}} t_2$.

As usual the measure of a step $s \leftarrow t$ is the measure of $t \to s$. Elementary steps are compared lexicographically using the multiset extension of $\preceq$ for the first component, the usual ordering over natural numbers for the components 2 and 3, and $\preceq$ for last ones. Since $\preceq$ is an AC-reduction ordering, the ordering defined over proofs is well-founded.

The general methodology is to show that a proof which contains some unwanted elementary steps can be replaced by a proof with a strictly smaller measure. Since the ordering over measures is well-founded, there exists a minimal proof, and such a minimal proof is of the desired form.

Lemma 3. A proof containing

- an elementary step $\longleftrightarrow_{s\approx t}$, where $s \approx t \in AC \cup \mathsf{X} \cup E_\infty$,
- or an elementary rewriting step truly of the form $\longrightarrow_{R_\infty}$ or $\longleftarrow_{R_\infty}$, or $\rightsquigarrow_{l\to r}$ or $\leftsquigarrow_{r\leftarrow l}$, where $l \to r \in R_\infty \setminus R_\omega$
- or a peak $s \leftarrow_{R_{\mathtt{can}}} t \to_{R_{\mathtt{can}}} s'$, $s \leftsquigarrow_{R_\omega} t \rightsquigarrow_{R_\omega} s'$, or $s \leftsquigarrow_{R_\omega} t \longrightarrow_{R_{\mathtt{can}}} s'$

is not minimal.

[6] Here,$s \longrightarrow_{R_\infty} t$ actually means $s \longrightarrow_{AC\setminus R_\infty} t'$ and $t = \mathtt{can}_{AC}(t')$.

Theorem 3. If s and t are two terms such that $s \longleftrightarrow^*_{AC,\mathsf{X},E_\infty,R_\infty} s'$ then $\mathtt{can}(s)\!\downarrow_{R\omega} = \mathtt{can}(t)\!\downarrow_{R\omega}$.

Proof. If s and s' are equal modulo $\longleftrightarrow^*_{AC,\mathsf{X},E_\infty,R_\infty}$, so are $\mathtt{can}(s)$ and $\mathtt{can}(s')$. By the above lemma, a minimal proof between $\mathtt{can}(s)$ and $\mathtt{can}(s')$ is necessary of the form $\mathtt{can}(s)(\rightsquigarrow_{R_\omega} \cup \rightarrow_{R_{\mathtt{can}}})^*(\leftsquigarrow_{R_\omega} \cup \leftarrow_{R_{\mathtt{can}}})^*\mathtt{can}(s')$. This sequence of steps can also be seen as $\mathtt{can}(s) \rightarrow^*_{R_{\mathtt{can}}} (\rightsquigarrow_{R_\omega} \rightarrow^*_{R_{\mathtt{can}}})^*(\leftarrow^*_{R_{\mathtt{can}}} \leftsquigarrow_{R_\omega})^* \leftarrow^*_{R_{\mathtt{can}}} \mathtt{can}(s')$. By definition, $\rightarrow_{R_{\mathtt{can}}}$ cannot follow a $\rightsquigarrow_{R_\omega}$-step, and $\mathtt{can}(s)$ and $\mathtt{can}(s')$ cannot be reduced by $\rightarrow_{R_{\mathtt{can}}}$, hence the wanted result.

5.3 Termination

We shall prove that, under a strongly fair strategy, R_ω is finite and obtained in a finite time (by cases on the head function symbol of the rule's left-hand side), and then we show that R_ω will clean up the next configurations and the completion process eventually halts on $\langle\ \emptyset \mid R_\omega\ \rangle$. In order to make our case analysis on rules, and to prove the needed invariants, we define several sets of terms (assuming without loss of generality that $E_0 = \mathtt{can}(E_0)$):

$$
\begin{array}{l}
T_0 = \{t \mid \exists t_0, e_1, e_2 \in \mathcal{T}_\Sigma(\mathcal{X}), e_1 \approx e_2 \in E_0 \ \text{ and } \ t_0 = e_i|_p \ \text{ and } \ t_0 \rightsquigarrow^*_{R_\infty} t\} \\
T_{0\mathsf{X}} = T_0 \cup \{f_{\mathsf{x}}(t_1, \ldots, t_n) \mid f_{\mathsf{x}} \in \Sigma_{\mathsf{X}} \ \text{ and } \ \forall i, t_i \in T_{0\mathsf{X}}\} \\
T_1 = \{t \mid t \in T_0 \ \text{ and } \ \forall p, t|_p \in T_{0\mathsf{X}}\} \\
T_2 = \{u(t_1, \ldots, t_n) \mid u \in \Sigma_{AC} \ \text{ and } \ \forall i, t_i \in T_1\}
\end{array}
$$

T_0 is the set of all terms and subterms in the original problem as well as their reducts by R_∞. The set $T_{0\mathsf{X}}$ moreover contains terms with X-aliens in T_0. T_1 is the set of terms that can be introduced by X from terms of T_0 (by solving or canonizing). T_2 is a superset of the terms built by critical pairs.

We first establish by structural induction over terms that:

$$\forall \gamma, t, s,\ \gamma \in R_\infty \cap T_j^2 \wedge t \in T_i \wedge t \rightsquigarrow_\gamma s \implies s \in T_i, \text{ for } i, j = 1, 2$$

Then, by induction over n, we show that any configuration $\langle\ E_n \mid R_n\ \rangle$ accessible from $\langle\ E_0 \mid \emptyset\ \rangle$ after n steps is such that $E_n \cup R_n \subseteq T_1^2 \cup T_2^2$.

The fact that R_∞ is finitely branching is a corollary of

Lemma 4. If $l \rightarrow r_n$ is created at step n in R_n and $l \rightarrow r_m$ at step m in R_m, with $n < m$, then r_m is a reduct of r_n by $\rightsquigarrow_{R_\infty}$.

The proof of this lemma is by induction over the length of the derivation, and by a case analysis over the applied inference rule.

Theorem 4. Under a strongly fair strategy, $\mathsf{AC(X)}$ terminates.

By the above properties, $\mathcal{R}_\omega$ can be divided in $\mathcal{R}_\omega \cap T_1^2$ and $\mathcal{R}_\omega \cap T_2^2$. $\mathcal{R}_\omega \cap T_1^2$ is finite, since all its left-hand sides are reducts of a finite number of terms by R_∞ which is well-founded and finitely branching. $\mathcal{R}_\omega \cap T_2^2$ is finite by using the same argument as in the ground AC-completion proof, based on the Higman's lemma. Hence $\mathcal{R}_\omega$ is finite and obtained in a finite number of steps, that is, there exists n such that $\mathcal{R}_\omega \subseteq R_n$. Then $\mathcal{R}_\omega$ will clean the rest of E_n, and the newly generated critical pairs will be discarded as trivial ones.

6 Experimental Results

AC(X) has been implemented in the Alt-Ergo [8] theorem prover. In this section, we show that this extension has strong impact both on performances and memory allocation *w.r.t.* an axiomatic approach. For that purpose, we benchmarked our implementation and compared its performances with state-of-the-art SMT solvers (Z3 v2.8, CVC3 v2.2, Simplify v1.5.4). All measures were obtained on a laptop running Linux equipped with a $2.58\,GHz$ dual-core Intel processor and with $4\,Gb$ main memory. Provers were given a time limit of five minutes for each test and memory limitation was managed by the system. The results are given in seconds; we write TO for *timeout* and OM for *out of memory*.

Our test suite is made of formulas which are valid in the combination of the theory of linear arithmetic $\mathcal{A}$, the free theory of equality[7] $\mathcal{E}$ and a small part of the theory of sets defined by the symbols $\cup$, $\subseteq$, the singleton constructor $\{\cdot\}$, and the following set of axioms:

$$AC \begin{cases} Assoc: & \forall x,y,z.\ \ x \cup (y \cup z) \approx (x \cup y) \cup z \\ Commut: & \forall x,y.\ \ x \cup y \approx y \cup x \end{cases}$$

$$\mathcal{S} \begin{cases} SubTrans: & \forall x,y,z.\ \ x \subseteq y \to y \subseteq z \to x \subseteq z \\ SubSuper: & \forall x,y,z.\ \ x \subseteq y \to x \subseteq y \cup z \\ SubUnion: & \forall x,y,z.\ \ x \subseteq y \to x \cup z \subseteq y \cup z \\ SubRefl: & \forall x.\ \ x \subseteq x \end{cases}$$

In order to get the most accurate information from our benchmarks, we classify formulas in three categories according to the subset of axioms needed to prove their validity. We use the standard mathematical notation $\bigcup_{i=1}^{d} a_i$ for the terms of the form $a_1 \cup (a_2 \cup (\cdots \cup a_d)) \cdots)$ and we write $\bigcup_{i=1}^{d} a_i; b$ for terms of the form $a_1 \cup (a_2 \cup (\cdots \cup (a_d \cup b))) \cdots)$. Formulas in the first category are of the form:

$$\bigwedge_{p=1}^{n} (\{e\} \cup \bigcup_{i=1}^{d} a_i^p) \approx b^p \ \to \ \underbrace{\bigwedge_{p=1}^{n} \bigwedge_{q=p+1}^{n} \ \bigcup_{i=d}^{1} a_i^p; b^q \approx \bigcup_{i=d}^{1} a_i^q; b^p}_{G}$$

and proving their validity only requires the theory $\mathcal{E}$ and the AC properties of $\cup$. The second category contains formulas additionally involving linear arithmetic:

$$\bigwedge_{p=1}^{n} (\{t_p - p\} \cup \bigcup_{i=1}^{d} a_i^p) \approx b^p \wedge \bigwedge_{p=1}^{n-1} t_p + 1 \approx t_{p+1} \to G$$

Formulas in the third category involve the $\subseteq$ symbol and are of the form:

$$\bigwedge_{p=1}^{n} \bigcup_{i=1}^{d} \{e_i^p\} \approx b^p \wedge \bigcup_{i=d}^{d} \{e + e_i^p\} \approx c^p \wedge e \approx 0 \to \bigwedge_{p=1}^{n} c^p \subseteq (b^p \cup \{e_d^p\}) \cup \{e\}$$

In order to prove their validity, we additionally need some axioms of $\mathcal{S}$. The results of the benchmarks are shown in Fig. 5, Fig. 6 and Fig 7. The first column contains the results for Alt-Ergo when we *explicitly* declare $\cup$ as an AC symbol and remove the AC axioms from the problem. In the second column, we do not take advantage of AC(X) and keep the AC axioms in the context. Figures 5 and

[7] These two theories are built-in for all SMT solvers we used for our benchmarks.

n-d	AC(X)	A-E	Z3	CVC3	SIMP.
3-3	**0.01**	0.19	0.22	0.40	0.18
3-6	**0.01**	32.2	OM	132	OM
3-12	**0.01**	TO	OM	OM	OM
6-3	**0.01**	11.2	1.10	13.2	2.20
6-6	**0.02**	TO	OM	OM	OM
6-12	**0.02**	TO	OM	OM	OM
12-3	**0.16**	TO	5.64	242	11.5
12-6	**0.24**	TO	OM	OM	OM
12-12	**0.44**	TO	OM	OM	OM

Fig. 5. AC + $\mathcal{E}$

n-d	AC(X)	A-E	Z3	CVC3	SIMP.
3-3	**0.00**	1.10	0.03	0.11	0.19
3-6	**0.00**	TO	3.67	4.21	OM
3-12	**0.00**	TO	OM	OM	OM
6-3	**0.02**	149	0.10	2.26	2.22
6-6	**0.02**	TO	17.7	99.3	OM
6-12	**0.04**	TO	OM	OM	OM
12-3	**0.27**	TO	0.35	44.5	11.2
12-6	**0.40**	TO	76.7	TO	OM
12-12	**0.72**	TO	OM	OM	OM

Fig. 6. AC + $\mathcal{E}$ + $\mathcal{A}$

n-d	AC(X)	A-E	Z3	CVC3	SIMP.
3-3	**0.02**	3.16	0.09	10.2	OM
3-6	**0.04**	TO	60.6	OM	OM
3-12	**0.12**	TO	OM	OM	OM
6-3	**0.07**	188	0.18	179	OM
6-6	**0.12**	TO	TO	OM	OM
6-12	**0.66**	TO	OM	OM	OM
12-3	**0.20**	TO	0.58	OM	OM
12-6	**0.43**	TO	TO	OM	OM
12-12	**1.90**	TO	OM	OM	OM

Fig. 7. AC + $\mathcal{E}$ + $\mathcal{A}$ + $\mathcal{S}$

6 show that, contrary to the axiomatic approach, built-in AC reasoning is little sensitive to the depth of terms: given a number n of equations, the running time is proportional to the depth d of terms. However, we notice a slowdown when n increases. This is due to the fact that AC(X) has to process a quadratic number of critical pairs generated from the equations in the hypothesis. From Fig. 7, we notice that Alt-Ergo with AC(X) performs better than the other provers. The main reason is that its instantiation mechanism is not spoiled by the huge number of intermediate terms the other provers generate when they instantiate the AC axioms.

7 Conclusion and Future Works

We have presented a new algorithm AC(X) which efficiently combines, in the ground case, the AC theory with a Shostak theory X and the free theory of equality. Our combination consists in a tight embedding of the canonizer and the solver for X in ground AC-completion. The integration of the canonizer relies on a new rewriting relation, reminiscent to normalized rewriting, which interleaves canonization and rewriting rules. We proved correctness of AC(X) by reusing standard proof techniques. Completeness is established thanks to a proofs' reduction argument, and termination follows the lines of the proof of ground AC-completion where the finitely branching result is adapted to account for the theory X.

AC(X) has been implemented in the Alt-Ergo theorem prover. The first experiments are very promising and show that a built-in treatment of AC, in the combination of the free theory of equality and a Shostak theory, is more efficient than an axiomatic approach. Although effective, the integration of AC(X) in Alt-Ergo fails to prove the formula

$$(\forall x, y, z. P((x \cup y) \cup z)) \wedge b \approx c \cup d \rightarrow P(a \cup b)$$

since the trigger for the internal quantified formula (the term $(x \cup y) \cup z$)) does not match the term $a \cup b$, even when exploiting the ground equation $b = c \cup d$ which allows to match the term $a \cup (c \cup d)$). Introducing explicitly the AC axioms for $\cup$ would allow the matcher to generate the ground term $(a \cup c) \cup d$ that could be matched. However, as shown by our benchmarks, too many terms are generated with these axioms in general. In order to fix this problem, we intend to extend

the pattern-matching algorithm of Alt-Ergo to exploit both ground equalities and properties of AC symbols. In the near future, we also plan to extend AC(X) to handle the AC theory with unit or idempotence. This will be a first step towards a decision procedure for a substantial part of the finite sets theory.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
2. L. Bachmair, N. Dershowitz, and J. Hsiang. Orderings for equational proofs. In *Proc. 1st LICS, Cambridge, Mass.*, pages 346–357, June 1986.
3. L. Bachmair, A. Tiwari, and L. Vigneron. Abstract congruence closure. *Journal of Automated Reasoning*, 31(2):129–168, 2003.
4. S. Conchon, E. Contejean, and M. Iguernelala. Canonized Rewriting and Ground AC Completion Modulo Shostak Theories. Research Report 1538, LRI, Dec. 2010.
5. E. Contejean. A certified AC matching algorithm. In V. van Oostrom, editor, *15th RTA*, volume 3091 of *LNCS*, pages 70–84, Aachen, Germany, June 2004. Springer.
6. N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17(3):279–301, Mar. 1982.
7. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 243–320. North-Holland, 1990.
8. S. Conchon and E. Contejean and F. Bobot and S. Lescuyer and M. Iguernelala. The Alt-Ergo theorem prover. http://alt-ergo.lri.fr
9. J.-M. Hullot. Associative commutative pattern matching. In *Proc. 6th IJCAI (Vol. I), Tokyo*, pages 406–412, Aug. 1979.
10. J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15(4), Nov. 1986.
11. D. Kapur. Shostak's congruence closure as completion. In H. Comon, editor, *Proc. 8th RTA*, volume 1232. Springer-Verlag, 1997.
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. S. Krstić and S. Conchon. Canonization for disjoint unions of theories. *Information and Computation*, 199(1-2):87–106, May 2005.
14. D. S. Lankford. Canonical inference. Memo ATP-32, University of Texas at Austin, Mar. 1975.
15. D. S. Lankford and A. M. Ballantyne. Decision procedures for simple equational theories with permutative axioms: Complete sets of permutative reductions. Memo ATP-37, University of Texas, Austin, Texas, USA, Aug. 1977.
16. C. Marché. On ground AC-completion. In R. V. Book, editor, *4th RTA*, volume 488 of *LNCS*, Como, Italy, Apr. 1991. Springer.
17. C. Marché. Normalized rewriting: an alternative to rewriting modulo a set of equations. *Journal of Symbolic Computation*, 21(3):253–288, 1996.
18. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. on Programming, Languages and Systems*, 1(2):245–257, Oct. 1979.
19. R. Nieuwenhuis and A. Rubio. A precedence-based total AC-compatible ordering. In C. Kirchner, editor, *Proc. 5th RTA, Montréal, LNCS 690*. Springer, June 1993.
20. G. E. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *J. ACM*, 28(2):233–264, Apr. 1981.
21. R. E. Shostak. Deciding combinations of theories. *J. ACM*, 31:1–12, 1984.