

Ground Associative and Commutative 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. In the ground case, 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 . The main ideas of our algorithm are first to adapt the definition of rewriting in order to integrate the canonizer of X and second, to replace the equation orientation mechanism found in ground AC-completion with the solver for X .

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

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

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. 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.

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 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 [11].

Unfortunately, AC reasoning is only a part of the automated deduction problem, and what we need is to decide formulas combining AC symbols and other theories. There are two main methods for combining decision procedures for disjoint theories. The Nelson-Oppen approach [13] is based on a variable abstraction mechanism and the exchange of equalities between shared variables. The Shostak's algorithm [15] 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 does not provide a solver for AC.

In this paper, we investigate 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 Shostak-like algorithm $\text{AC}(X)$ is an extension of ground AC-completion. The main ideas consist in adapting the notion of rewriting to integrate the canonizer of X and in replacing the equation orientation mechanism found in ground AC-completion with the solver for X .

2 Preliminaries and Notations

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 $a \dots f$ denote 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 . 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 [3] 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 $\preceq \cap \not\succeq$.

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 equations that can be obtained by reflexivity, symmetry, transitivity, congruence and instances of equations in E^1 . 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 $l \rightarrow r$. A term s rewrites to a term t at position p by the rule $l \rightarrow r$, denoted by $s \rightarrow_{l \rightarrow r}^p t$, 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 $s \rightarrow_R t$ whenever there exists a rule $l \rightarrow r$ of R such that s rewrites to t by $l \rightarrow r$ at some position. A normal form of a term s w.r.t to R is a term t such that $s \rightarrow_R^* t$ 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 [7] 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 [9].

Let $=_{AC}$ be the equational theory defined by $\bigcup_{u \in \Sigma_{AC}} \{u(x, y) \approx u(y, x), u(x, u(y, z)) \approx u(u(x, y), z)\}$. 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 $E \cup AC$. 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 $u(a, u(b, c)) \rightarrow t$, we would like the reductions $f(u(c, u(b, a)), d) \rightarrow f(t, d)$ and $u(a, u(c, u(d, b))) \rightarrow u(t, d)$ to be possible. The AC property of u is needed in the first one for the subterm $u(c, u(b, a))$ to match the term $u(a, u(b, c))$, and in the second one 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.

Definition 1 (Ground rewriting modulo AC). *A term s rewrites to a term t modulo AC at position p by the rule $l \rightarrow r$, denoted by $s \rightarrow_{AC \setminus l \rightarrow r}^p t$, 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.

¹The equational theory of the free theory of equality \mathcal{E} , defined by the empty set of equations, is simply denoted $=$.

3 Shostak Theories and Global Canonization

Throughout the rest of the paper, we assume given a theory X with a signature Σ_X . A canonizer for X is a function $\text{can}_X : \mathcal{T}_{\Sigma_X}(\mathcal{X}) \rightarrow \mathcal{T}_{\Sigma_X}(\mathcal{X})$ that computes a unique normal form for every term such that $s =_X t$ iff $\text{can}_X(s) = \text{can}_X(t)$. A solver for X is a function solve_X that solves equations between Σ_X -terms. Given an equation $s \approx t$, $\text{solve}_X(s \approx t)$ either returns a special value \perp when $s \approx t \cup X$ 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 [8].

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 solve_X to handle heterogeneous equations. Following the technique described in [8], we can define a global canonizer can by combining can_X with the canonizer for AC defined in [4] and formally proved in [2]. This framework assumes a global one-to-one mapping $\alpha : \mathcal{T}_{\Sigma} \rightarrow \mathcal{X}$ and its inverse mapping ρ . It is based on a variable abstraction mechanism which computes the *pure* Σ_X -part $\llbracket t \rrbracket$ of a heterogeneous term t as follows:

$$\llbracket t \rrbracket = f(\llbracket \vec{s} \rrbracket) \text{ when } t = f(\vec{s}) \text{ and } f \in \Sigma_X \quad \text{and} \quad \llbracket t \rrbracket = \alpha(t) \text{ otherwise}$$

Using the same mappings α , ρ and the abstraction function, the wrapper solve can be easily defined:

$$\text{solve}(s \approx t) = \begin{cases} \perp & \text{if } \text{solve}_X(\llbracket s \rrbracket \approx \llbracket t \rrbracket) = \perp \\ \{x_i \rho \mapsto t_i \rho\} & \text{if } \text{solve}_X(\llbracket s \rrbracket \approx \llbracket t \rrbracket) = \{x_i \approx t_i\} \end{cases}$$

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

$$\begin{aligned} & \forall t \in \mathcal{T}_{\Sigma}, \text{can}(t) \preceq t \\ & \forall s, t \in \mathcal{T}_{\Sigma}, \text{if } \text{solve}(s \approx t) = \cup \{p_i \mapsto v_i\} \text{ then } v_i \prec p_i \end{aligned}$$

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

$$\begin{aligned} & \forall t \in \mathcal{T}_{\Sigma}, \text{can}_X(\llbracket t \rrbracket) \preceq \llbracket t \rrbracket \\ & \forall s, t \in \mathcal{T}_{\Sigma}, \text{if } \text{solve}_X(\llbracket s \rrbracket \approx \llbracket t \rrbracket) = \cup \{x_i \approx t_i\} \text{ then } t_i \rho \prec x_i \rho \end{aligned}$$

Requiring such properties implies that the AC-reduction ordering \preceq must be based on a precedence relation \prec_p such that $\Sigma_X \prec_p \Sigma_{\mathcal{E}} \cup \Sigma_{AC}$. From now on, we assume that X is locally compatible with \preceq .

4 Ground AC-Completion Modulo X

We adapt the notion of ground AC-rewriting to cope with canonizers. 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 [12] by replacing normalization with canonization.

Definition 2 (Canonized rewriting). *A term s can-rewrites to a term t at position p by the rule $l \rightarrow r$, denoted by $s \rightsquigarrow_{l \rightarrow r}^p t$, iff (1) $s \rightarrow_{AC \setminus l \rightarrow r}^p t'$ and (2) $\text{can}(t') = t$.*

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 $\text{AC}(X)$ given in Figure 1. The state of the procedure is a pair $\langle E \mid R \rangle$ of equations and rewriting rules. The initial configuration is a pair $\langle E_0 \mid \emptyset \rangle$ where E_0 is supposed to be a set of equations between canonized terms. Since $\text{AC}(X)$'s rules only involve canonized rewriting, the algorithm maintains the invariant that terms occurring in E and R are in canonical forms; **Trivial** thus removes an equation $u \approx v$ from E when u and v are syntactically equal. A new rule **Bottom** 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 **Orient** rule. The other rules are much similar to those of ground AC-completion except that the rewriting relation is \rightsquigarrow_R . R is used to rewrite either side of an equation (**Simplify**), and to reduce right hand side of rewriting rules (**Compose**). Given a rule $l \rightarrow r$, **Collapse** 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 $l' \rightarrow r$ which is added to E as an equation in order to be re-oriented. Finally, **Deduce** adds equational consequences of rewriting rules to E . For instance, if R contains two rules of the form $u(a,b) \rightarrow s$ and $u(a,c) \rightarrow t$, 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 rewriting rules are computed by the following function (where a^μ stands for the maximal term (w.r.t. size) enjoying the assertion):

$$\text{headCP}(R) = \{u(b, r') \approx u(b', r) \mid l \rightarrow r \in R, l' \rightarrow r' \in R, \exists a^\mu. l =_{\text{AC}} u(a^\mu, b) \wedge l' =_{\text{AC}} u(a^\mu, b') \}$$

$\text{Trivial} \frac{\langle E \cup \{s \approx t\} \mid R \rangle}{\langle E \mid R \rangle} s = t$	$\text{Bottom} \frac{\langle E \cup \{s \approx t\} \mid R \rangle}{\langle \emptyset \mid \cup_{t \in \mathcal{T}_E} \{t \rightarrow \perp\} \rangle} \text{solve}(s, t) = \perp$
$\text{Orient} \frac{\langle E \cup \{s \approx t\} \mid R \rangle}{\langle E \mid R \cup \text{solve}(s, t) \rangle} \text{solve}(s, t) \neq \perp$	
$\text{Simplify} \frac{\langle E \cup \{s \approx t\} \mid R \rangle}{\langle E \cup \{s' \approx t\} \mid R \rangle} s \rightsquigarrow_R s'$	$\text{Compose} \frac{\langle E \mid R \cup \{l \rightarrow r\} \rangle}{\langle E \mid R \cup \{l \rightarrow r'\} \rangle} r \rightsquigarrow_R r'$
$\text{Collapse} \frac{\langle E \mid R \cup \{g \rightarrow d, l \rightarrow r\} \rangle}{\langle E \cup \{l' \approx r\} \mid R \cup \{g \rightarrow d\} \rangle} \begin{cases} l \rightsquigarrow_{g \rightarrow d} l' \\ l \neq g \vee d \prec r \end{cases}$	
$\text{Deduce} \frac{\langle E \mid R \rangle}{\langle E \cup \{s \approx t\} \mid R \rangle} s \approx t \in \text{headCP}(R)$	

Figure 1: Inference rules for ground AC-completion modulo X

Theorem 3. *Given a set E of ground equations, the application of the rules of $\text{AC}(X)$ under a fair strategy terminates and produces a final configuration $\langle \emptyset \mid R \rangle$ such that*

$$\forall s, t \in \mathcal{T}_\Sigma. s =_{E, \text{AC}, X} t \iff s \downarrow_R = t \downarrow_R$$

5 Conclusion and Future works

We have presented a new algorithm $\text{AC}(X)$ which efficiently combines, in the ground case, the AC theory with a Shostak theory X and the free theory of equality. Our combination relies on a tight integration of

the canonizer and the solver of X in the ground AC-completion procedure. Our technique can be viewed as an extension of the Shostak's method to handle AC symbols.

$AC(X)$ has been implemented in the ALT-ERGO theorem prover where it extends its core with built-in AC reasoning. However, in order to fully benefit from $AC(X)$ in the prover, the instantiation lemma mechanism has to be extended to cope both with AC and ground equalities. In the near future, we also plan to extend $AC(X)$ itself to handle the AC theory with unit or idempotence. This will be a first step towards a decision procedure for a part of the finite sets theory. Finally, we shall explore whether the general properties of normalized rewriting can be adapted to our new canonized rewriting relation.

Related Works. AC-completion has been studied for a long time in the rewriting community [10, 14]. A generic framework for combining completion with a generic built-in equational theory E has been proposed in [5]. Normalized completion [12] 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. $AC(X)$ 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 [11] to show the correctness of $AC(X)$. Kapur [6] used ground completion to demystify Shostak's congruence closure algorithm and Bachmair *et al.* [1] 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.

References

- [1] L. Bachmair, A. Tiwari, and L. Vigneron. Abstract congruence closure. *Journal of Automated Reasoning*, 31(2):129–168, 2003.
- [2] E. Contejean. A certified AC matching algorithm. In *15th RTA*, volume 3091 of *LNCS*, pages 70–84, Aachen, Germany, June 2004. Springer.
- [3] N. Dershowitz. Orderings for term rewriting systems. *TCS*, 17(3):279–301, Mar. 1982.
- [4] J.-M. Hullot. Associative commutative pattern matching. In *6th IJCAI, Tokyo*, pages 406–412, Aug. 1979.
- [5] 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.
- [6] D. Kapur. Shostak's congruence closure as completion. In *8th RTA*, volume 1232. Springer-Verlag, 1997.
- [7] 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.
- [8] S. Krstić and S. Conchon. Canonization for disjoint unions of theories. *Information and Computation*, 199(1-2):87–106, May 2005.
- [9] D. S. Lankford. Canonical inference. Memo ATP-32, University of Texas at Austin, Mar. 1975.
- [10] D. S. Lankford and A. M. Ballantyne. Decision procedures for simple equational theories with permutative axioms: Complete sets of permutative reductions. Research Report Memo ATP-37, Department of Mathematics and Computer Science, University of Texas, Austin, Texas, USA, Aug. 1977.
- [11] C. Marché. On ground AC-completion. In *4th RTA*, volume 488 of *LNCS*, Como, Italy, Apr. 1991. Springer.
- [12] C. Marché. Normalized rewriting: an alternative to rewriting modulo a set of equations. *JSC*, 21(3):253–288, 1996.
- [13] G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. on Programming, Languages and Systems*, 1(2):245–257, Oct. 1979.
- [14] G. E. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *J. ACM*, 28(2):233–264, Apr. 1981.
- [15] R. E. Shostak. Deciding combinations of theories. *J. ACM*, 31:1–12, 1984.