Modular & Incremental Proofs of AC-Termination

Xavier Urbain

Laboratoire de Recherche en Informatique (LRI)
CNRS UMR 8623
Bât. 490, Université Paris-Sud, Centre d'Orsay
91405 Orsay Cedex, France
E-mail: urbain@lri.fr

May 14, 2002

Abstract

Recently, the framework of rewriting modules was proposed and provided modular and incremental termination criteria. In this paper, we extend these results to the important case of Associative and Commutative rewriting by means of AC-dependency pairs.

1 Introduction

Either in programming or for automated proof, it is very common to use symbols with an associative and commutative (AC) behaviour. However, a direct expression of AC by means of rules (for instance, $x+y \to y+x$ where $+$ is AC) is unpracticable since these rules do not terminate. In order to preserve the termination property, rewriting over AC equivalence classes (namely rewriting *modulo* AC) is used.

Introducing this new kind of relation complicates termination proofs significantly. In particular, the orders involved in proofs have to be AC-*compatible*. For years the problem of finding suitable orderings has been widely studied [2, 3, 5, 20] but all propositions lead to *simplification orderings* which are far too restrictive in practice. Hence automated termination proof has been a difficult task.

Several recent approaches extend the dependency pairs approach [1] to the AC case [9, 13, 15, 16], which enhances feasibility of automating termination proofs of AC-rewriting.

However, proving termination of a term rewriting system still remains a problem, especially when the system is large, which is indeed the case for most practical applications of rewriting. This problem is due to the fact that a divide and conquer strategy cannot be applied directly. this makes automation of proofs for systems with many rules a difficult task.

Rewriting modules [21, 22] were introduced as a possible answer to this big systems termination problem. From this framework arise powerful criteria both incremental (information that a system is known terminating is taken into account) and modular (irrelevant rules are never considered in proof steps).

Since AC symbols are useful in practice, combining AC techniques with modular methods, both automation compatible, leads to important results.

After giving our notations and some preliminaries, we will firstly describe briefly the AC approach of Kusakari, Marché and Urbain which is the one we consider, and secondly describe the modules framework and show how it extends to the AC case. From our theorems and their corollaries we obtain a new method for proving AC termination in an incremental/modular way, fully suitable for automation.

Our results are implemented in the termination tool of the C*i*ME system [4]. Termination proofs examples through the paper were found by C*i*ME in a completely automated way.

2 Notations & preliminaries

We suppose that the reader is familiar with the basic notions of rewriting and termination, especially with the dependency pairs approach, and refer to the survey of Dershowitz & Jouannaud [8] for details and to Arts & Giesl [1] regarding dependency pairs. We precise our notations hereafter.

The root position in a term is denoted by Λ, symbol at root position in a term t by $\Lambda(t)$, $t|_p$ denotes the subterm of t at position p. We use postfix notation for substitution applications.

A *term rewriting system* (TRS for short) over a signature[1] $\mathcal{F}$ and a set of variables X is a set $R(\mathcal{F})$ of *rewrite rules* $l \to r$. A TRS R defines a rewrite relation $\to_R$ the following way: $s \to_R t$ if there is a position p such that $s|p = l\sigma$ and $t = s[r\sigma]_p$ for a rule $l \to r \in R$ and a substitution σ. We then say that $s|_p$ is a *redex* and that s *reduces to* t *at position* p *with* $l \to r$ or, if the rule of R is not relevant, *with* R, respectively denoted $s \xrightarrow[l \to r]{p} t$ and $s \xrightarrow[R]{p} t$.

A term is *strongly normalizable* (SN) if it cannot reduce indefinitely. A rewrite relation is *strongly normalizing* or *terminates* if any term is SN. In this paper, we will use indifferently 'strongly normalizing' and 'terminating'.

We focus on the notion of $\mathcal{C}_\mathcal{E}$-termination [10, 11, 17]. A terminating TRS R is said to be $\mathcal{C}_\mathcal{E}$-terminating if $R \cup \Pi$ terminates where $\Pi = \{\pi(x,y) \to x \quad (\pi_1),\ \pi(x,y) \to y \quad (\pi_2)\}$ for a fresh symbol π.

$\mathcal{C}_\mathcal{E}$-termination behaves nicely with reference to unions of TRS. It is actually a modular property for unions of

- disjoint TRS (Gramlich, Ohlebusch) [11, 17],

[1] We shall omit the signature if there is no ambiguity.

- finitely branching and constructor sharing systems (Gramlich) [11],
- finitely branching and composable systems (Kurihara & Ohuchi) [12].

$\mathcal{C}_{\mathcal{E}}$-termination is less restrictive than simple termination: whenever adding projections for *all* symbols allows elimination of any constructors 'hiding' a redex, it is impossible with rules of Π only.

In particular:

Proposition 1. (Gramlich [11])

Any simply terminating TRS is $\mathcal{C}_{\mathcal{E}}$-terminating.

$\mathcal{C}_{\mathcal{E}}$-termination is not modular for constructor sharing unions with non-finitely branching TRS, as shown by Ohlebusch [17], hence we restrict our study to the *finitely branching* case, i.e. such that the set of rules that can be applied to a term is always finite. Note that TRS are finite in practice.

Termination is usually proven with the help of *reduction orderings* [6] or quasi-orderings with dependency pairs. We briefly recall what we need. A *term ordering*, also known as *ordering pair* [14] is a pair $(\succeq, \succ)$ of relations over $T(\mathcal{F}, X)$ such that: $\succeq$ is a quasi-ordering, i.e. reflexive and transitive; $\succ \subseteq \succeq - \preceq$ is a strict ordering, i.e. irreflexive and transitive and such that $\succ \subseteq \succeq$, $\succ \cdot \succeq \subseteq \succ$ and $\succeq \cdot \succ \subseteq \succ$. A term ordering is said to be *well-founded* if there is no infinite strictly decreasing sequence $t_1 \succ t_2 \succ \cdots$; *stable* if both $\succ$ and $\succeq$ are stable under substitutions; *weakly* (resp. *strictly*) *monotonic* if for all terms t_1 and t_2, for all $f \in \mathcal{F}$, if $t_1 \succeq$ (resp. $\succ$) t_2 then $f(\ldots, t_1, \ldots) \succeq$ (resp. $\succ$) $f(\ldots, t_2, \ldots)$; A term ordering $(\succeq, \succ)$ is called a *weak (resp. strict) reduction ordering* if it is well-founded, stable and weakly (resp. strictly) monotonic; *a simplification ordering* if it is stable, monotonic and have the subterm property.

Definition 1. A term ordering $(\succeq, \succ)$ over $T(\mathcal{F}, X)$ is said to be π-*expandable* if there is a reduction ordering $(\succeq', \succ')$ over $T(\mathcal{F} \cup \{\pi : 2\}, X)$ such that:

- $(\succeq', \succ')$ restricted to $T(\mathcal{F}, X)$ is exactly $(\succeq, \succ)$;
- $\pi(s,t) \succeq' s$ and $\pi(s,t) \succeq' t$ for all s and t in $T(\mathcal{F}, X)$.

A π-expandable ordering is *strongly* π-*expandable* (respectively *weakly* π-*expandable*) if both it and its associate ordering are strongly monotonic (resp. weakly monotonic). Any simplification ordering is π-expandable [21, 22]. In particular RPO et orderings induced by polynomial intepretations are π-expandable. Lexicographical compositions of π-expandable orderings as well as their use with *Recursive Program Schemes* [1] leads to π-expandables orderings [21, 22].

π-expandable ordering may be used for proving ($\mathcal{C}_{\mathcal{E}}$-)termination.

Proposition 2.

Let $(\succeq, \succ)$ be a strictly monotonic π-expandable ordering. If for each rule $l \to r$ of a term rewriting system R, $l \succ r$, then R is $\mathcal{C}_{\mathcal{E}}$-strongly normalising.

We distinguish between modular and incremental proofs. A modular property $\mathcal{P}$ behaves as follow: If components C_1 and C_2 have property $\mathcal{P}$ so does the union $C_1 \cup C_2$. Proving a property $\mathcal{P}'$ in an incremental manner consists in proving $\mathcal{P}'$ on a component C_1 firstly, then of showing $\mathcal{P}'$ on $C_1 \cup C_2$, possibly using the knowledge that C_1 verifies $\mathcal{P}'$.

2.1 AC-termination with dependency pairs

We will briefly describe in this section a particular extension of the dependency pairs approach to the AC case by Kusakari, Marché and the author [13]. We will deal with AC-extended rewriting on flat terms.

2.1.1 AC-dependency pairs

By *unmarked (dependency) pairs* we denote elements of the union of (Arts & Giesl's) dependency pairs of rules *and* extended rules.

Let $\widehat{\mathcal{F}} = \mathcal{F} \cup \{\widehat{f}, f \text{ defined symbol}\}$ an extended signature where each defined symbol is copied with a mark. For a flat term s over signature $\mathcal{F}$, whose root symbol is a defined one, we denote $\widehat{s}$ the term over $\widehat{\mathcal{F}}$ obtained by replacing the root symbol with its marked copy.

In order to deal with terms headed by AC symbols, we will also have to consider *partially marked* terms. These are terms whose top symbol $\widehat{f}$ is marked, and possibly with non-marked symbols f below $\widehat{f}$ (even if f is AC).

Let t be a term with top symbol f. A term t' is said *partially marked* from t if

- $\Lambda(t') = \widehat{f}$;
- the only marked symbol in t' is $\widehat{f}$;
- if f is not AC, then the only occurrence of $\widehat{f}$ is at top;
- an occurrence of $\widehat{f}$ in t' never occurs below a non-marked symbol;
- when replacing each occurrence of $\widehat{f}$ by f in t', we get a term AC-equivalent to t.

Par(t) denotes the set of partial markings of t.

Let R be a TRS, we denote $\widehat{R}$ the following 'mark erasing' system:

$$\{\widehat{f}(\mathbf{u}) \to f(\mathbf{u}) \ \mid f \text{ defined symbol in } R\}.$$

and we denote by $\xrightarrow{m.e.}$ the rewrite relation by $\widehat{R}$ modulo AC, restricted to pair of terms that are both partially marked from a given term.

Notice that in particular, $\xrightarrow{m.e.}$ cannot erase the mark at top.

We recall now the definition of so-called *DP conditions*.

Definition 2. Let be given an unmarked pair $\langle u, v\rangle$ of R. A set of pairs $\{\dots, \langle u_i, v_i\rangle, \dots\}$ is a *set of marked dependency pairs* for $\langle u, v\rangle$ if each of the following *DP conditions* holds:

1. for all i, there is a substitution σ_i such that $\widehat{v\sigma_i} \xrightarrow{m.e.}^{\star} v_i$

2. for all substitution σ, for all $t \in \mathit{Par}(u\sigma)$, there exists an index i and a substitution σ' such that

$$\sigma =_{AC} \sigma_i\sigma'$$
$$t \xrightarrow{m.e.}^{\star} u_i\sigma'$$

A *set of marked dependency pairs* for R is a union of sets of marked dependency pairs for all unmarked pairs of R.

2.1.2 Proving termination with AC-dependency pairs

Given a set of marked dependency pairs of R, we define a *marked dependency chain* as follows.

Definition 3. A *marked R-chain* of dependency pairs is a sequence $\langle s_1, t_1\rangle$, $\langle s_2, t_2\rangle$,... of marked pairs of R such that for a substitution σ and for each i,

$$t_i\sigma(\xrightarrow[R]{i})^{\star} \cdot \xrightarrow{m.e.}^{\star} s_{i+1}\sigma$$

Theorem 1. (Kusakari, Marché, Urbain [13])
Let R be a TRS, $\widehat{DP}$ a set of marked pairs, assume that $\widehat{DP}$ satisfies the DP conditions. If there is an (non-marked) infinite chain then an infinite marked chain exists.

The following corollary gives a termination criterion using marked pairs.

Corollary 1–1. (Kusakari, Marché, Urbain [13])
If there is an AC-compatible weak reduction ordering $\succeq$ such that $l \succeq r$ for each rule of R, $s \succ t$ for each marked dependency pair $\langle s, t\rangle$ of R, and for each defined AC symbol f, $\widehat{f}(x, y) \succeq f(x, y)$ then R is AC-terminating.

2.2 Rewriting modules

In this section, we shall briefly define *rewriting modules* [21, 22]. We will recall some modular and incremental results obtained thanks to the modules framework.

2.2.1 Notion of rewriting modules

Definition 4. Let R_1 be a term rewriting system over a signature $\mathcal{F}_1$. A *module* extending $R_1(\mathcal{F}_1)$ is a pair $[\mathcal{F}_2 \mid R_2]$ such that:

1. $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$ (signatures are disjoint);

2. R_2 is a term rewriting system over $\mathcal{F}_1 \cup \mathcal{F}_2$;

3. for all $l \to r \in R_2$, $\Lambda(l) \in \mathcal{F}_2$.

One can easily see that $R_1 \cup R_2$ is a TRS over $\mathcal{F}_1 \cup \mathcal{F}_2$. We say then that system $R_1 \cup R_2$ over $\mathcal{F}_1 \cup \mathcal{F}_2$ is a *hierarchical extension of* $R_1(\mathcal{F}_1)$ *with module* $[\mathcal{F}_2 \mid R_2]$. We write such an extension:

$$R_1(\mathcal{F}_1) \longleftarrow [\mathcal{F}_2 \mid R_2].$$

A module can extend more than one system and the extension by a module naturally associates 'arrow-wise'. Hence we will write

$$R_1(\mathcal{F}_1) \longleftarrow [\mathcal{F}_2 \mid R_2] \longleftarrow [\mathcal{F}_3 \mid R_3]$$

the extension

$$(R_1(\mathcal{F}_1) \longleftarrow [\mathcal{F}_2 \mid R_2]) \longleftarrow [\mathcal{F}_3 \mid R_3].$$

Remark 1. — For the sake of readability, we may denote

$$[\mathcal{F}_2 \mid R_2] \longleftarrow [\mathcal{F}_3 \mid R_3]$$

the hierarchical extension (with $[\mathcal{F}_3 \mid R_3]$) of the whole hierarchy extended with (that is *headed by*) $[\mathcal{F}_2 \mid R_2]$.

For instance, we may abbreviate "the hierarchy headed by $[\mathcal{F}_2 \mid R_2]$ is extended with $[\mathcal{F}_3 \mid R_3]$", denoted

$$R_1(\mathcal{F}_1) \longleftarrow [\mathcal{F}_2 \mid R_2] \longleftarrow [\mathcal{F}_3 \mid R_3],$$

in "$[\mathcal{F}_3 \mid R_3]$ extends $[\mathcal{F}_2 \mid R_2]$", denoted

$$[\mathcal{F}_2 \mid R_2] \longleftarrow [\mathcal{F}_3 \mid R_3].$$

Two disjoint modules may extend the same base hierarchy.

Definition 5. We say that a module $[\mathcal{F}_2 \mid R_2]$ extends a hierarchy headed by $[\mathcal{F}_0 \mid R_0]$ *independently of* a module $[\mathcal{F}_1 \mid R_1]$ if:

- $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$,
- $[\mathcal{F}_0 \mid R_0] \longleftarrow [\mathcal{F}_1 \mid R_1]$ and
- $[\mathcal{F}_0 \mid R_0] \longleftarrow [\mathcal{F}_2 \mid R_2]$.

We should point out that the only condition we put on extensions is that new rules must have a new symbol at root position of their left-hand side (see Def. 4), thus, modules extensions subsume notions of heavily constrained hierarchical extensions such as *constructor based extensions* [7] and *proper extensions* [18, 19].

Any TRS can be studied as a certain modules hierarchy exploited in its present state i.e., as it is provided. However, it is possible to consider a unique canonical decomposition. A TRS can actually be seen as a hierarchy of *minimal modules*, modules which cannot be split up themselves into a hierarchy of non-empty modules.

2.2.2 Dependency pairs of Modules & relative chains

Definition 6. Let $\widehat{\mathcal{F}}$ be signature $\mathcal{F}$ extended with 'marked copies' $\widehat{f}$ for all symbols f. Marking of a term is as follow: for a term $t \in T(\mathcal{F}, X)$, we denote $\widehat{t}$ term t for which the symbol at Λ has been replaced by its marked copy.

Let $M = [\mathcal{F} \mid R]$ be a module. A *dependency pair of module M* is a pair of terms $\langle \widehat{l}, \widehat{r'} \rangle$ such that there is a rule $l \to r \in R$ for which term r' is a subterm of r with $\Lambda(r') \in \mathcal{F}$.

We denote DP(M) the set of all dependency pairs of a module M.

Since a module can extend more than one hierarchy, dependency chains become *relative* to some relevant set of rules.

Definition 7. Let $M = [\mathcal{F} \mid R]$ be a module and let S be an arbitrary term rewriting system. A *dependency chain of M over S* is a sequence of pairs of DP(M) together with a substitution σ such that for any two successive pairs $\langle s_i, t_i \rangle$ and $\langle s_{i+1}, t_{i+1} \rangle$:

$$t_i\sigma \xrightarrow[S]{\neq\Lambda} s_{i+1}\sigma$$

A dependency chain of a module M over a system S is said to be *minimal* if all proper subterms of instantiated pairs rhs are S strongly normalizable.

2.2.3 Termination with Modules

Dependency pairs of modules and relative dependency chains allow us to define some purely syntactical tests so as to prove termination in an incremental/modular fashion [21, 22].

Theorem 2. (Urbain [21, 22])
Let $[\mathcal{F}_1 \mid R_1] \longleftarrow [\mathcal{F}_2 \mid R_2]$ be a hierarchical extension of $R_1(\mathcal{F}_1)$;

1. if R_1 is $\mathcal{C}_{\mathcal{E}}$-strongly normalising,

2. if there is no infinite dependency chain of $[\mathcal{F}_2 \mid R_2]$ over $R_1 \cup R_2$,

then $R_1 \cup R_2$ is strongly normalising.

By considering the extension of R_1 with $[\mathcal{F}_2 \cup \{\pi : 2\} \mid R_2 \cup \Pi]$ and with the help of Theorem 2 we obtain as a corollary a sufficient condition for composability of $\mathcal{C}_{\mathcal{E}}$-termination.

Corollary 2–1. (Urbain [21,22])
Let $[\mathcal{F}_1 \mid R_1] \longleftarrow [\mathcal{F}_2 \mid R_2]$ be a hierarchical extension of $R_1(\mathcal{F}_1)$;

1. if R_1 is $\mathcal{C}_{\mathcal{E}}$-strongly normalising,

2. if there is no infinite dependency chain of $[\mathcal{F}_2 \mid R_2]$ over $R_1 \cup R_2 \cup \Pi$,

then $R_1 \cup R_2$ is $\mathcal{C}_{\mathcal{E}}$-strongly normalising.

Theorem 3. (Urbain [21,22])
Let $[\mathcal{F}_1 \mid R_1] \longleftarrow [\mathcal{F}_2 \mid R_2]$ be a hierarchical extension of $R_1(\mathcal{F}_1)$ and let $[\mathcal{F}_3 \mid R_3]$ be a module extending R_1 independently of R_2;

1. if $R_1 \cup R_2$ is $\mathcal{C}_{\mathcal{E}}$-strongly normalising,

2. if there is no infinite dependency chain of $[\mathcal{F}_3 \mid R_3]$ over $R_1 \cup R_3 \cup \Pi$,

then $R_1 \cup R_2 \cup R_3$ is $\mathcal{C}_{\mathcal{E}}$-strongly normalising.

Theorem 2 is clearly an incremental result while Theorem 3 is a modular one.
Both proofs of these theorems use a key lemma which is fully proven in earlier work on modules [21, 22]. The point of this paper is to show how an AC-version of this lemma applies to an extension of modules to the AC case.

3 AC rewriting modules

In this section we shall combine the respective powers of AC-dependency pairs and of the modular approach. The modules framework is general enough to allow an extension to the AC-case.

Since there are few interferences of AC-steps over the modular structure of TRS, definitions and proofs are close to those of the standard (non-AC) case. Hence we will go into details mainly when differences arise.

In order to preserve some crucial properties of $\mathcal{C}_{\mathcal{E}}$-termination (the definition of which does not change modulo AC) it is necessary to keep the finitely branching systems restriction.

3.1 AC-modules

The notion of module is extended to AC-extended rewriting over flat terms. Signatures of AC-modules may contain associative and commutative symbols.

Definition 8. Let R_1 be an AC-TRS over signature $\mathcal{F}_1$. A *module extending* R_1 is a pair $[\mathcal{F}_2 \mid R_2]$ such that:

1. $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$;

2. R_2 is an AC-TRS over $\mathcal{F}_1 \cup \mathcal{F}_2$;

3. for each $l \to r$ in R_2, $\Lambda(l) \in \mathcal{F}_2$.

Remark 2. — The important point here is that only one symbol is involved in each AC-step. Hence, there is no permutation of signatures inside terms: AC-steps *introduce no dependency* between modules.

3.2 AC-dependency pairs of modules

Unmarked dependency pairs of a module actually are the relevant extended pairs in the module signature.

Definition 9. Let us consider a hierarchical extension $R_1(\mathcal{F}_1) \longleftarrow [\mathcal{F}_2 \mid R_2]$.

The *set of* AC*-dependency pairs of module* $[\mathcal{F}_2 \mid R_2]$ is the subset of AC-extended pairs of $R_1 \cup R_2$ the defined symbols of which all belong to $\mathcal{F}_2$.

AC-relative dependency chains are defined accordingly.

Definition 10. An AC*-dependency chain of a module* $[\mathcal{F} \mid R]$ *over a TRS* R' (with $R \subseteq R'$) is a sequence of pairs $\dots \langle s_j, t_j \rangle \dots$ ($\langle s_j, t_j \rangle \in \mathrm{DP}([\mathcal{F} \mid R])$) together with a substitution σ such that for all successive pairs $\langle s_i, t_i \rangle, \langle s_{i+1}, t_{i+1} \rangle$: $t_i\sigma \xrightarrow[R']{\neq\Lambda}{}^{\star} s_{i+1}\sigma$ holds.

These definitions are very close to those of the standard case, the existence or not of AC-symbols is enough to remove any ambiguity.

4 Incremental termination proofs for AC-rewriting

By using Remark 2 we are able to state and prove similar results as in the standard (non-AC) case.

In particular, since there is no permutation of symbols from different signatures, marked ones cannot appear in pairs with instantiation.

4.1 Key lemma for the AC case

Lemma 1.

Let S_1 and S_2 be AC-TRS over signature $\mathcal{F}_1$. Let S_3 be an AC-TRS over $\mathcal{F}_1 \cup \mathcal{F}_2$ such that:

- $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$;
- for each $l \to r \in S_3$, $\Lambda(l) \in \mathcal{F}_2$.

If there is an infinite chain of $[\mathcal{F}_1 \mid S_2]$ over $S_1 \cup S_2 \cup S_3$, then there is an infinite chain of $[\mathcal{F}_1 \mid S_2]$ over $S_1 \cup S_3 \cup \Pi$ with the same sequence of pairs but with a new substitution and new rewriting steps.

We use an interpretation of flat AC-terms.

Definition 11. Let us denote $S = S_1 \cup S_2 \cup S_3$; let $>$ be an arbitrary yet *total* ordering over $T_\infty(\widehat{\mathcal{F}_1} \cup \{\pi : 2\} \cup \{\perp : 0\}, X)$.

Interpretation $I^{\mathrm{AC}}(x) : T(\widehat{\mathcal{F}_1} \cup \mathcal{F}_2, X) \to T_\infty(\widehat{\mathcal{F}_1} \cup \{\pi : 2\} \cup \{\perp : 0\}, X)$ is defined as follows:

$$\begin{array}{rcl} I^{\mathrm{AC}}(x) & = & x \text{ if } x \in X, \\ I^{\mathrm{AC}}(f(t_1 \dots t_n)) & = & \left\{ \begin{array}{l} f(I^{\mathrm{AC}}(t_1) \dots I^{\mathrm{AC}}(t_n)) \text{ if } f \in \widehat{\mathcal{F}_1}, \\ \mathrm{Comb}(\mathrm{Red}(f(t_1 \dots t_n))) \text{ if } f \in \mathcal{F}_2, \end{array} \right. \end{array}$$

where

$$\begin{array}{rcl} \mathrm{Red}(t) & = & \{I^{\mathrm{AC}}(t') / \ t \xrightarrow[S_1 \cup S_2 \cup S_3/\mathrm{AC}]{} t'\}, \\ \mathrm{Comb}(\emptyset) & = & \perp, \\ \mathrm{Comb}(\{a\} \cup E) & = & \pi(a, \mathrm{Comb}(E)) \text{ where for each } e \in E, a < e). \end{array}$$

Remark 3. — Set $\mathrm{Red}(t)$ of t' such that t reduces to t' in one step of $S_1 \cup S_2 \cup S_3$ modulo AC is well defined; we choose for each AC-equivalence class a unique inhabitant.

An interpretation of a term t such that $\Lambda(t) \in \mathcal{F}_2$ is a 'comb-like' term whose subterms are interpretations of canonical (because of flattening) inhabitants of AC-equivalence classes of one-step redexes of t (using AC-extended-rewriting over flat terms). Hence it is possible to reach any of these interpretation using an appropriate $\xrightarrow[\pi_2]{}^{\star} \xrightarrow[\pi_1]{}$ reduction.

Proofs of the following lemmata are very similar to those of their non-AC version [21, 22].

Lemma 2.
For all $t \in T(\widehat{\mathcal{F}_1}, X)$ and all substitutions σ,

$$I^{\mathrm{AC}}(t\sigma) = tI^{\mathrm{AC}}(\sigma).$$

Lemma 3.
For all $t_1, \dots, t_n$ in $T(\widehat{\mathcal{F}_1} \cup \mathcal{F}_2, X)$ and for each C context over $\mathcal{F}_1$ with n holes,

$$I^{\mathrm{AC}}(C[t_1, \dots, t_n]) = C[I^{\mathrm{AC}}(t_1), \dots, I^{\mathrm{AC}}(t_n)].$$

Lemma 4.

For each term t strongly normalisable by $S_1 \cup S_2 \cup S_3$, $I^{\text{AC}}(t)$ is finite.

Now, it is sufficient to check that our interpretation behaves the way we expect it to.

Lemma 5.

For all s and t of $T(\widehat{\mathcal{F}_1} \cup \mathcal{F}_2, X)$ and for each $l \to r \in S_1 \cup S_2$,

$$\text{if } s \xrightarrow[l\to r/\text{AC}]{p} t \text{ then } I^{\text{AC}}(s) \xrightarrow[S_1\cup S_2\cup\Pi/\text{AC}]{}^{+} I^{\text{AC}}(t).$$

Moreover, if $p \neq \Lambda$ and $\Lambda(s) \in \widehat{\mathcal{F}_1}$ then $I^{\text{AC}}(s) \xrightarrow[S_1\cup S_2\cup\Pi/\text{AC}]{\neq\Lambda}^{+} I^{\text{AC}}(t)$.

From Remark 2 we know that AC-steps will never switch symbols from $\mathcal{F}_1$ with symbols from $\mathcal{F}_2$. Hence the proof is similar to the standard case [21, 22]. Two cases depending on symbols occurring on the path from Λ à p:

- only symbols from $\mathcal{F}_1$, then applying Lemmata 3 and then 2 and thanks to premises: $I^{\text{AC}}(s) \xrightarrow[S_1\cup S_2/\text{AC}]{} I^{\text{AC}}(t)$;
- there is a smallest (w.r.t. prefix ordering) $p' < p$ such that $\Lambda(s|_{p'}) \in \mathcal{F}_2$, let us suppose $s = C[s_1, \ldots, s', \ldots, s_n]$ where C is a context over $\mathcal{F}_1$, $p = p'q$ and $s|_{p'} = s'$, then by Lemma 3 $I^{\text{AC}}(s) = C[I^{\text{AC}}(s_1), \ldots, I^{\text{AC}}(s'), \ldots, I^{\text{AC}}(s_n)]$.

  From the definition of I^{AC} we may conclude:

  $$C[I^{\text{AC}}(s_1), \ldots, I^{\text{AC}}(s'), \ldots, I^{\text{AC}}(s_n)] \xrightarrow[\Pi/\text{AC}]{}^{+} I^{\text{AC}}(t).$$

  Eventually, the resulting term may have to be flattened.

Similarly (second case), simulation of S_3-steps is possible also.

Lemma 6.

For all s and t in $T(\widehat{\mathcal{F}_1} \cup \mathcal{F}_2, X)$, if $s \xrightarrow[S_3/\text{AC}]{p} t$ then $I(s) \xrightarrow[\Pi/\text{AC}]{}^{+} I(t)$.

Moreover, if $\Lambda(s) \in \widehat{\mathcal{F}_1}$ then $I(s) \xrightarrow[\Pi/\text{AC}]{\neq\Lambda}^{+} I(t)$.

Proof of Lemma 1

Let $\langle u_1, v_1\rangle, \langle u_2, v_2\rangle, \ldots$ be a dependency chain of $[\mathcal{F}_1 \mid S_2]$ over $S_1 \cup S_2 \cup S_3$ with a substitution σ. Let σ' be the substitution such that for each x, $x\sigma' = I(x\sigma)$.

By minimality of the considered chain, substitution σ is strongly normalisable, hence we know by Lemma 4 that terms instantiated by σ' are all finite.

We show that $\langle u_1, v_1\rangle, \langle u_2, v_2\rangle, \ldots$ with σ' is a chain of $[\mathcal{F}_1 \mid S_2]$ over $S_1 \cup S_2 \cup \Pi$.

In that purpose, we have to prove for each i:

$$v_i\sigma' \xrightarrow[S_1\cup S_2\cup\Pi/\mathrm{AC}]{\neq\Lambda}{}^{\star} u_{i+1}\sigma'.$$

We know

$$v_i\sigma \xrightarrow[S_1\cup S_2\cup S_3/\mathrm{AC}]{\neq\Lambda}{}^{\star} \xrightarrow{m.e.}{}^{\star} u_{i+1}\sigma.$$

Let us consider a step $s \xrightarrow[S_1\cup S_2\cup S_3/\mathrm{AC}]{p} t$ from that reduction. Since

$$\Lambda(s) = \Lambda(t) = \Lambda(v_i) = \Lambda(u_{i+1}) \in \widehat{\mathcal{F}_1},$$

by Lemma 6 or Lemma 5 we get

$$I^{\mathrm{AC}}(s) \xrightarrow[S_1\cup S_2\cup\Pi/\mathrm{AC}]{\neq\Lambda}{}^{\star} I^{\mathrm{AC}}(t).$$

We may actually build the sequence by juxtaposition of steps so as to obtain

$$I^{\mathrm{AC}}(v_i\sigma) \xrightarrow[S_1\cup S_2\cup\Pi]{\neq\Lambda}{}^{\star} I^{\mathrm{AC}}(u_{i+1}\sigma).$$

Since $I^{\mathrm{AC}}(v_i\sigma) = v_i\sigma'$ and $I^{\mathrm{AC}}(u_{i+1}\sigma) = u_{i+1}\sigma'$ we conclude using Lemma 2. $\square$

4.2 Main results

With the help of Lemma 1 we may prove the following results regarding (respectively) extensions with one module and extension with several independent modules. Thanks to Remark 2 the proof sketch is similar to those of Theorems 2 and 3.

Theorem 4.
Let $[\mathcal{F}_1 \mid R_1] \longleftarrow [\mathcal{F}_2 \mid R_2]$ be a hierarchical extension of $R_1(\mathcal{F}_1)$ where $\mathcal{F}_1 \cup \mathcal{F}_2$ contains AC-symbols,

1. if R_1 is $\mathcal{C}_\mathcal{E}$-strongly normalising modulo AC,
2. if there is no infinite chain of $[\mathcal{F}_2 \mid R_2]$ over $R_1 \cup R_2/\mathrm{AC}$,

then $R_1 \cup R_2$ is strongly normalising modulo AC.

Theorem 5.
Let $[\mathcal{F}_1 \mid R_1] \longleftarrow [\mathcal{F}_2 \mid R_2]$ be a hierarchical extension of $R_1(\mathcal{F}_1)$, let $[\mathcal{F}_3 \mid R_3]$ be a module extending R_1 independently of R_2,

1. if $R_1 \cup R_2$ is $\mathcal{C}_\mathcal{E}$-strongly normalising modulo AC,
2. if there is no infinite chain of $[\mathcal{F}_3 \mid R_2]$ over $R_1 \cup R_3 \cup \Pi/\text{AC}$,

then $R_1 \cup R_2 \cup R_3$ is $\mathcal{C}_\mathcal{E}$-strongly normalising modulo AC.

In particular, we extend to AC rewriting a previous result by Kurihara & Ohuchi [12].

Corollary 5–1.
$\mathcal{C}_\mathcal{E}$-AC-termination is a modular property for unions of composable TRS.

4.3 Corollaries & practical application

The theorems we get may be used in practical application with the help of AC-compatible π-expandable orderings.

Corollary 4–1.
Let $[\mathcal{F}_1 \mid R_1] \longleftarrow [\mathcal{F}_2 \mid R_2]$ be a hierarchical extension of $R_1(\mathcal{F}_1)$,

1. if R_1 is $\mathcal{C}_\mathcal{E}$-strongly normalising modulo AC,
2. if there is no infinite chain of $[\mathcal{F}_2 \mid R_2]$ over $R_1 \cup R_2 \cup \Pi/\text{AC}$,

then $R_1 \cup R_2$ is $\mathcal{C}_\mathcal{E}$-strongly normalising modulo AC.

Corollary 4–2.
Let $[\mathcal{F}_1 \mid R_1] \longleftarrow [\mathcal{F}_2 \mid R_2]$ be a hierarchical extension of $R_1(\mathcal{F}_1)$,

1. if R_1 is $\mathcal{C}_\mathcal{E}$-strongly normalising modulo AC;
2. if there is a (weak) reduction ordering (resp. weakly π-expandable) AC-compatible $(\succeq, \succ)$ such that:
   - $R_1 \cup R_2 \subseteq \succeq$ and
   - DP($[\mathcal{F}_2 \mid R_2]$) $\subseteq \succ$, for a set of AC-dependency pairs of modules,

then $R_1 \cup R_2$ is strongly normalising (resp. $\mathcal{C}_\mathcal{E}$-strongly normalising) modulo AC.

Corollary 5–2.
Let $[\mathcal{F}_1 \mid R_1] \longleftarrow [\mathcal{F}_2 \mid R_2]$ be a hierarchical extension of $R_1(\mathcal{F}_1)$, let $[\mathcal{F}_3 \mid R_3]$ be a module extending R_1 independently of R_2,

1. if $R_1 \cup R_2$ is $\mathcal{C}_\mathcal{E}$-strongly normalising modulo AC;
2. if there is a weakly π-expandable ordering AC-compatible $(\succeq, \succ)$ such that:
   - $R_1 \cup R_3 \subseteq \succeq$,
   - DP($[\mathcal{F}_3 \mid R_3]$) $\subseteq \succ$, for a set of marked AC-dependency pairs of modules,

then $R_1 \cup R_2 \cup R_3$ is $\mathcal{C}_\mathcal{E}$-strongly normalising modulo AC.

4.4 Example

The modular/incremental AC approach has been fully implemented in the C*i*ME termination toolbox.

Instead of the usual Peàno's arithmetic example, we chose an example occurring in real programming. Binary arithmetics is really implemented, so that is what we are going to study.

So let us consider the system describing nonnegative rational integers (which clearly $\mathcal{C}_{\mathcal{E}}$-terminates):

$$\begin{array}{ll} \mathcal{F}_{\mathbb{N}} & \{\# : \textit{constant}\,; 1, 0 : \text{(suffix) } \textit{unary}\} \\ R_{\mathbb{N}} & \{\#0 \to \#. \end{array}$$

We shall define some arithmetic on these integers, in particular addition with $[\mathcal{F}_+ \mid R_+]$ where $+$ is associative and commutative.

$$\begin{array}{ll} \mathcal{F}_+ & \{+ : \text{AC}\} \\ R_+ & \left\{ \begin{array}{lcl} x + \# & \to & x \\ x0 + y0 & \to & (x+y)0 \\ x0 + y1 & \to & (x+y)1 \\ x1 + y1 & \to & (x+y+\#1)0 \end{array} \right. \end{array}$$

Termination of $R_{\mathbb{N}} \cup R_+$ is proven using dependency pairs and a polynomial interpretation.

$$\begin{array}{ll} [\![\#]\!] = 0; & \\ [\![0]\!](x) = x+1; & [\![1]\!](x) = x+2; \\ [\![+]\!](x,y) = x+y; & [\![\widehat{+}]\!](x,y) = x+y. \end{array}$$

We may now prove termination of the extension with the module defining multiplication.

$$\begin{array}{ll} \mathcal{F}_\times & \{\times : \text{AC}\} \\ R_\times & \left\{ \begin{array}{lcl} x \times \# & \to & \# \\ x0 \times y & \to & (x \times y)0 \\ x1 \times y & \to & (x \times y)0 + y \end{array} \right. \end{array}$$

Knowing that $R_{\mathbb{N}} \cup R_+$ $\mathcal{C}_{\mathcal{E}}$-terminates, a polynomial interpretation again is sufficient to fulfil remaining constraints:

$$\begin{array}{lll} [\![\#]\!] = 0; & [\![0]\!](x) = x+1; & [\![1]\!](x) = x+1; \\ [\![+]\!](x,y) = x+y; & & \\ [\![\times]\!](x,y) = 2xy + 2(x+y) + 1; & & \\ [\![\widehat{\times}]\!](x,y) = xy + x + y. & & \end{array}$$

Now, we may define bags, then sum and product of elements in a bag of integers.

$$\begin{array}{ll} \mathcal{F}_{\mathsf{Bag}} & \{\emptyset : \textit{constant},\ \{\_\} : \textit{unary},\ \cup : \text{AC}\} \\ R_{\mathsf{Bag}} & \{\ \emptyset \cup x \quad \to \quad x \end{array}$$

This system clearly $\mathcal{C}_\mathcal{E}$-terminates.

$$
\begin{array}{ll}
\mathcal{F}_\Sigma & \{\Sigma : \textit{unary}\} \\
R_\Sigma & \left\{ \begin{array}{lll} \Sigma(\emptyset) & \to & \# \\ \Sigma(\{x\}) & \to & x \\ \Sigma(x \cup y) & \to & \Sigma(x) + \Sigma(y) \end{array} \right.
\end{array}
$$

Since we know that $R_\mathbb{N} \cup R_+$ $\mathcal{C}_\mathcal{E}$-terminates, a polynomial interpretation is enough:

$$
\begin{array}{lll}
[\![\#]\!] = 1; & [\![0]\!](x) = x; & [\![1]\!](x) = 0; \\
[\![+]\!](x, y) = x * y; & [\![\emptyset]\!] = 1; & [\![\{\_\}]\!](x) = x; \\
\multicolumn{3}{l}{[\![\cup]\!](x, y) = 2xy + 2(x + y) + 1;} \\
\multicolumn{3}{l}{[\![\Sigma]\!](x) = x;} \\
\multicolumn{3}{l}{[\![\widehat{\Sigma}]\!](x) = x.}
\end{array}
$$

Eventually, we shall prove $\mathcal{C}_\mathcal{E}$-termination of product similarly:

$$
\begin{array}{ll}
\mathcal{F}_\bigotimes & \{\bigotimes : \textit{unary}\} \\
R_\bigotimes & \left\{ \begin{array}{lll} \bigotimes(\emptyset) & \to & \#1 \\ \bigotimes(\{x\}) & \to & x \\ \bigotimes(x \cup y) & \to & \bigotimes(x) \times \bigotimes(y) \end{array} \right.
\end{array}
$$

with interpretation:

$$
\begin{array}{lll}
[\![\#]\!] = 0; & [\![0]\!](x) = 0; & [\![1]\!](x) = 1; \\
[\![+]\!](x, y) = x + y; & [\![\times]\!](x, y) = xy; & \\
[\![\emptyset]\!] = 1; & [\![\{\_\}]\!](x) = x; & \\
\multicolumn{3}{l}{[\![\cup]\!](x, y) = 2xy + 2(x + y) + 1;} \\
\multicolumn{3}{l}{[\![\bigotimes]\!](x) = x;} \\
\multicolumn{3}{l}{[\![\widehat{\bigotimes}]\!](x) = x.}
\end{array}
$$

As far as we know, this is the first termination proof of this system, with polynomial interpretations only. Other approaches may need for instance lexicographical composition which is not possible to implement efficiently, and ACRPO [20] is of no use here because of rule $x1 + y1 \to (x + y + \#1)0$. Moreover those interpretations were found in a few seconds by C*i*ME in a fully automated way.

5 Conclusion

We combined the respective powers of two recent approaches: a method based on dependency pairs for AC-rewriting [13] and the exploitation of the TRS structure using rewriting modules [21, 22] and $\mathcal{C}_\mathcal{E}$-termination. In so doing, we proposed an extension to the AC case of the modules framework and gave new incremental/modular termination criteria for AC-rewriting: Theorem 4, Theorem 5 and corollaries. It is now possible to perform termination proofs of systems with AC symbols in an incremental/modular fashion, fully compatible with automation. The resulting criteria are implemented in the termination toolbox of the C*i*ME rewriting tool. For instance, its termination expert is able to give an automated termination proof with polynomial interpretation of a AC-TRS describing binary arithmetics without anything else than polynomial interpretations. This is the first proof of this kind as far as we know. For application such as analysis of parallel/communicating processes where systems with numerous rules often occur, our results give users the opportunity to leave the difficult task of finding the proof to a powerful automated tool like C*i*ME.

References

[1] Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

[2] Leo Bachmair and David A. Plaisted. Termination orderings for associative-commutative rewriting systems. *Journal of Symbolic Computation*, 1(4):329–349, December 1985.

[3] Ahlem Ben Cherifa and Pierre Lescanne. An actual implementation of a procedure that mechanically proves termination of rewriting systems based on inequalities between polynomial interpretations. In Jörg H. Siekmann, editor, *8th International Conference on Automated Deduction*, volume 230 of *Lecture Notes in Computer Science*, pages 42–51, Oxford, England, July 1986. Springer-Verlag.

[4] Evelyne Contejean, Claude Marché, Benjamin Monate, and Xavier Urbain. Cime version 2, 2000. Prerelease available at `http://cime.lri.fr/`.

[5] Catherine Delor and Laurence Puel. Extension of the associative path ordering to a chain of associative-commutative symbols. In Claude Kirchner, editor, *5th International Conference on Rewriting Techniques and Applications*, volume 690 of *Lecture Notes in Computer Science*, pages 389–404, Montreal, Canada, June 1993. Springer-Verlag.

[6] Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1):69–115, February 1987.

[7] Nachum Dershowitz. Hierarchical termination. In N. Dershowitz and N. Lindenstrauss, editors, *Proceedings of the Fourth International Workshop on Conditional and Typed Rewriting Systems (Jerusalem, Israel, July 1994)*, volume 968, pages 89–105, Berlin, 1995. Springer-Verlag.

[8] Nachum Dershowitz and Jean-Pierre Jouannaud. Notations for rewriting. *EATCS Bulletin*, 43:162–172, 1990.

[9] Jürgen Giesl and Deepak Kapur. Dependency pairs for equational rewriting. In Aart Middeldorp, editor, *12th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science, pages 93–108, Utrecht, The Netherlands, May 2001. Springer-Verlag.

[10] B. Gramlich. A structural analysis of modular termination of term rewriting systems. Research Report SR-91-15, University Kaiserslautern, 1991.

[11] Bernhard Gramlich. Generalized sufficient conditions for modular termination of rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 5:131–158, 1994.

[12] Masahito Kurihara and Azuma Ohuchi. Decomposable termination of composable term rewriting systems. *IEICE*, E78–D(4):314–320, April 1995.

[13] Keiichirou Kusakari, Claude Marché, and Xavier Urbain. Termination of associative-commutative rewriting using dependency pairs criteria. Research Report 1304, LRI, 2002.

[14] Keiichirou Kusakari, Masaki Nakamura, and Yoshihito Toyama. Argument filtering transformation. In Gopalan Nadathur, editor, *Principles and Practice of Declarative Programming, International Conference PPDP'99*, volume 1702 of *Lecture Notes in Computer Science*, pages 47–61, Paris, 1999. Springer-Verlag.

[15] Keiichirou Kusakari and Yoshihito Toyama. On proving AC-termination by AC-dependency pairs. *IEICE Transactions on Information and Systems*, E84-D(5):604–612, 2001.

[16] Claude Marché and Xavier Urbain. Termination of associative-commutative rewriting by dependency pairs. In Tobias Nipkow, editor, *9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 241–255, Tsukuba, Japan, April 1998. Springer-Verlag.

[17] Enno Ohlebusch. On the modularity of termination of term rewriting systems. *Theoretical Computer Science*, 136:333–360, 1994.

[18] M. R. K. Krishna Rao. Simple termination of hierarchical combinations of term rewriting systems. In *Theoretical Aspects of Computer Software*, volume 789 of *Lecture Notes in Computer Science*, pages 203–223. Springer-Verlag, 1994.

[19] M.R.K. Krishna Rao. Modular proofs for completeness of hierarchical term rewriting systems. *Theoretical Computer Science*, 151:487–512, 1995.

[20] Albert Rubio. A fully syntactic ac-rpo. In Paliath Narendran and Michaël Rusinowitch, editors, *Rewriting Techniques and Applications*, volume 1631

of *Lecture Notes in Computer Science*, pages 133–147, Trento, Italy, July 1999. Springer-Verlag.

[21] Xavier Urbain. *Approche incrémentale des preuves automatiques de terminaison*. Thèse de doctorat, Université Paris-Sud, Orsay, France, October 2001.

[22] Xavier Urbain. Automated incremental termination proofs for hierarchically defined terms rewriting systems. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Automated Reasoning*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 485–498, Siena, Italy, june 2001. Springer-Verlag.