Associative-commutative rewriting via flattening

Jean-Pierre Jouannaud*

LIX, École Polytechnique
91400 Palaiseau, France
Email: jouannaud@lix.polytechnique.fr
http://www.lix.polytechnique.fr/Labo/jouannaud

Abstract. AC-rewriting is simulated by using flattened terms, flattened rewrite rules, extensions and specializations with respect to the AC-operators, therefore allowing us to reduce AC-pattern matching and AC-unification to permutative matching and permutative unification respectively.

1 Introduction

In this paper, we reduce AC-rewriting and checking AC-rewriting for confluence to the problem of rewriting flattened terms modulo permutations and checking permutative-rewriting for confluence. Moreover, we give a simple proof of modularity for AC-rewriting.

2 Preliminaries

We assume given a *signature* (or *vocabulary*) $\mathcal{F}$ of *function symbols*. $T(\mathcal{F}, \mathcal{X})$ denotes the set of *terms* built up from $\mathcal{F}$ and $\mathcal{X}$. We assume familiarity with the basic concepts and notations of term rewriting systems and refer to [1] for supplementary definitions and examples.

Terms are identified with finite labelled trees as usual. *Positions* are strings of positive integers, the root position corresponding to the empty string Λ. We use $\mathcal{P}os(t)$ (resp. $\mathcal{FP}os(t)$) to denote the set of positions (resp. non-variable positions) of t, $t|_p$ for the *subterm* of t at position p, and $t[u]_p$ for the result of replacing $t|_p$ with u at position p in t. This notation is also used to indicate that u is a subterm of t. $\mathcal{V}ar(t)$ denotes the set of variables occuring in t.

Substitutions are written as in $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ where $t_i \neq x_i$. The *domain* of σ is $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_n\}$. We use greek

* Project LogiCal, Pôle Commun de Recherche en Informatique du Plateau de Saclay, CNRS, École Polytechnique, INRIA, Université Paris-Sud.

letters for substitutions and postfix notation for their application. Composition is denoted by juxtaposition. Bijective substitutions are called *variable renamings*.

A *rewrite rule* is a pair of terms, written $l \to r$, such that $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. A *term rewriting system* is a set of rewrite rules $R = \{l_i \to r_i\}_i$. A term t rewrites to a term u at position p with the rule $l \to r \in R$ and the substitution σ, written $t \xrightarrow[l \to r]{p} u$ if $t|_p = l\sigma$ and $u = t[r\sigma]_p$. Such a term t is called *reducible*. An irreducible term is called a *normal form*. A term rewriting system R is *confluent* (resp. *Church-Rosser*) if $t \to^* u$ and $t \to^* v$ (resp. $u \leftrightarrow^* v$) implies $u \to^* s$ and $v \to^* s$ for some s.

The reflexive transitive closure of a relation $\to$, denoted by $\to^*$, is called *derivation*, while its symmetric, reflexive, transitive closure, denoted by $\leftrightarrow^*$. $\leftrightarrow^*_R$ or $=_R$, is called *equationnal theory* generated by the rules in R considered as equations (that is, oriented both ways).

3 AC rewriting

This section collects and combines various techniques found in the literature [6, 8, 3–5, 7, 2]. Assuming that some binary function symbols in $\mathcal{F}_{AC} = \{+_1, \ldots, +_p \mid p \geq 0\} \subseteq \mathcal{F}$ are associative and commutative, we define classicaly AC-rewriting as $s \longrightarrow^p_R t$ iff $s|_p =_{AC} l\sigma$ and $t = s[r\sigma]_p$ for some rule $l \to r \in R$ and substitution σ. The Church-Rosser property becomes $s =_{R \cup AC} t$ iff $s \longrightarrow^*_R u$, $t \longrightarrow^*_R v$ and $u =_{AC} v$ for some terms u, v. We reserve the letters $+$ and $*$ for AC symbols.

3.1 Flattened terms.

It is usual to handle terms with associative-commutative symbols by *flattening*, writing $s \downarrow_{flat}$ for the flattened normal form of the term s. Associative-commutative symbols become varyadic, and cannot have as argument a term headed by the same varyadic symbol. This can be enforced by a simple type system. Our grammar of raw terms is :

$$T \ \to \ \mathcal{X} \mid f(T, \ldots, T) \mid \ +_i (T, \ldots, T)$$

The type system ensuring flattening is based on typing judgements of the form $\vdash s : T_i$, for $i \in [0..p]$, where T_0 denotes the set of flattened terms not headed by a function symbol in $\mathcal{F}_{AC}$, and $T_{i \in [1..p]}$ the set of flattened

terms headed by $+_i$. We use T for the whole set of flattened terms :

$$\textbf{Variables:}\quad \frac{x:\sigma\in\Gamma}{\Gamma\vdash x\in T_0}\qquad \textbf{Terms:}\quad \frac{\vdash t\in T_i\quad i\in[0..p]}{\vdash t\in T}$$

$$\textbf{Plain terms:}\quad \frac{f:n\in\mathcal{F}\setminus\mathcal{F}_{AC}\quad \vdash t_1\in T\ \ldots\ \vdash t_n\in T}{\vdash f(t_1,\ldots,t_n)\in T_0}$$

$$\textbf{AC-terms:}\quad \frac{+_i:AC\in\mathcal{F}\quad n\geq 2\quad \forall j\in[1..n]\ \vdash t_j\in T \text{ and } t_j(\epsilon)\neq +_i}{\vdash +_i(t_1,\ldots,t_n)\in T_i}$$

We write $\mathcal{F}_{flat}$ for the varyadic signature, and $T(\mathcal{F}_{flat},\mathcal{X})$ for the set of flattened terms. It is well known that two terms in $T(\mathcal{F},X)$ are AC-equivalent iff their flattened forms are equivalent under the *permutative* congruence $=_P$ generated by the equations in the set

$$\{+_i(x_1,\ldots,x_n)=+_i(x_{\sigma(1),\ldots,\sigma(n)})\mid\sigma\in\Sigma_{n\geq 2}\}$$

where Σ_n is the permutation group of order n. Commutative and other kinds of permutative symbols can be accomodated by adding the associated permutative equations, but keeping the arity fixed. Our results include all such combinations of theories. Permutative theories enjoy many properties: matching is decidable and equivalent terms have the same size, a property that we may use without mentionning.

A flattened term s is a *subterm* of a flattened term t at position p if $t|_p=_P s$ or else $s=+(s_1,\ldots,s_m)$ and $t|_p=_P+(s_1,\ldots,s_m,t_1,\ldots,t_n)$. Given a term t and a position $p\in\mathcal{P}os(t)$, $Flat(t,p)$ shall denote the position $q\in\mathcal{P}os(t{\downarrow_{flat}})$ such that $t|_p{\downarrow_{flat}}$ is a subterm of $t{\downarrow_{flat}}\,|_q$.

3.2 Safe rewriting.

A remaining problem is that flattened terms are not closed under both context application and instantiation, since a variable argument of a varyadic operator can be replaced by a term headed by this very same operator. This makes it impossible to rewrite flattened terms with $R{\downarrow_{flat}}$.

Definition 1. *A pair* $(u[]_p,\gamma)$ *is* safe *with respect to the term* s *if* $u[s\gamma]_p$ *is a flattened term.*

The problem with rewriting on flattened terms is to compute contexts and substitutions which are safe with respect to both the lefthand and

righthand side of rules. For this, we assume that the rewrite system operating on flattened terms is closed under the following inference rules:

$$\textbf{Left-extension:}\quad \frac{+(\boldsymbol{l}) \to g(\boldsymbol{r}) \in R \quad x \notin \mathcal{V}ar(\boldsymbol{l}) \quad g \neq +}{+(\boldsymbol{l}, x) \to +(g(\boldsymbol{r}), x) \in R}$$

$$\textbf{Right-extension:}\quad \frac{f(\boldsymbol{l}) \to +(\boldsymbol{r}) \in R \quad x \notin \mathcal{V}ar(\boldsymbol{l}) \quad f \neq +}{+(f(\boldsymbol{l}), x) \to +(\boldsymbol{r}, x) \in R}$$

$$\textbf{Left-Right-extension:}\quad \frac{+(\boldsymbol{l}) \to +(\boldsymbol{r}) \in R \quad x \notin \mathcal{V}ar(\boldsymbol{l})}{+(\boldsymbol{l}, x) \to +(\boldsymbol{r}, x) \in R}$$

$$\textbf{Specialization:}\quad \frac{l \to r \in R \quad x \text{ below } + \text{ in } l \text{ or } r \quad y \notin \mathcal{V}ar(l)}{l\{x \mapsto +(x,y)\}\downarrow_{flat} \to r\{x \mapsto +(x,y)\}\downarrow_{flat} \in R}$$

where x below $+$ in s iff $+(x, \vec{s})$ is a subterm of s for some vector $\vec{s}$

Note that $l\{x \mapsto +(x,y)\}$ is a raw term that must be possibly flattened (at all places immediately above an occurence of x in l) to become a term. Extension rules appeared first in [8] and specializations in [5]. Combining both ensures safe rewriting. Rules originating from a rule $l \to r$ in R are called its *variants*.

Let R_{flat} be the closure of $R\downarrow_{flat}$ under the above inference rules. R_{flat} is normally infinite, but only a finite part of it is needed to rewrite a term, since a given flattened term has only finetely many subterms and the permutative theory is size-preserving. AC-rewriting becomes:

Definition 2. *Given two flattened terms t and u, we say that t* rewrites *to s, written $t \xrightarrow[l \to r \in R_{flat}]{p} u$ if $t|_p \leftrightarrow_P^{(\geq p)^*} l\sigma$ and $u = t[r\sigma]_p$.*

Note that both t and u must be flattened terms: rewriting a term t at position p with a rule $l \to r \in R_{flat}$ and a substitution σ requires that the pair $(t[]_p, \sigma)$ is safe with both l and r.

Example 1. Let $R = \{*(x,x) \to +(g(+(x,0)),x)\}$. Then,

$$R_{flat} = \begin{cases} *(x,x,y) \to *(+(g(+(x,0)),x),y) & \dots \\ (+(x,y),+(x,y)) \to +(g(+(x,y,0)),x,y) & \dots \\ (+(x,y),+(x,y),z) \to *(+(g(+(x,y,0)),x,y),z) & \dots \end{cases}$$

$s = *(+(0,0),+(0,0)) \longrightarrow^{\Lambda}_{*(+(x,y),+(x,y)) \to +(g(+(x,y,0)),x,y)} +(g(+(0,0,0)),0,0)$, but s does not rewrite with R, since the result would not be flattened.

3.3 Church-Rosser properties.

The question arises whether the flattened form of a term rewritable modulo AC with R is indeed rewritable with R_{flat}.

Lemma 1. *Assume that $s \longrightarrow^{p}_{R_{AC}} t$ with a rule $l \to r$. Then, there exists a position $p' \in \mathcal{P}os(s{\downarrow_{flat}})$, a variant $l' \to r'$ of $l \to r \in R$ and a substitution σ' such that*

(i) the pair $(s{\downarrow_{flat}}\ []_{p'}, \sigma')$ is safe for both l and r;

(ii) $s{\downarrow_{flat}} =_P s{\downarrow_{flat}}\ [l'\sigma']_P$ and $t{\downarrow_{flat}} =_P s{\downarrow_{flat}}\ [r'\sigma']_P$, hence $s{\downarrow_{flat}} \longrightarrow_{l'\to r'} v =_P t{\downarrow_{flat}}$.

Proof. We define first the position p', then the variant $l' \to r'$ and the substitution σ'. Two steps are needed.

Step 1. There are two cases:

(i) If $s(p) \notin \mathcal{F}_{AC}$ or $l(\Lambda) \neq s(p)$ and $r(\Lambda) \neq s(p)$, then $p' = Flat(s, p)$ and $l' \to r' = l \to r$.

(ii) If $s(p) \in \mathcal{F}_{AC}$ and $l(\Lambda) = s(p)$ or $r(\Lambda) = s(p)$, let $p' = Flat(t, q)$ where q is the smallest position in s such that $\forall q \leq o \leq p\ s(o) = s(p)$. Let now $n = |p| - |q|$. Then $l' \to r'$ is the n-times extension of $l \to r$. It is a Left-extension if $l(\Lambda) = s(p)$, a Right-extension if $r(\Lambda) = s(p)$, and a Left-Right-extension if both $l(\Lambda)$ and $r(\Lambda)$ are equal to $s(p)$.

Step 2. The substitution σ is first flattened, yielding σ'. Then, in case $x\sigma = +(s_1, \ldots, s_n)$ with $+ \in \mathcal{F}_{AC}$ for some variable x such that $+(l_1, \ldots, l_m, x)$ is a subterm of l, then the rule $l' \to r'$ must be specialized n times. This creates n new variables $y_1, \ldots, y_n$ such that $y_i\sigma' = s_i$. This process is applied to each variable x satisfying the above property. The triple $(p', l' \to r', \sigma')$ satisfies properties (i) and (ii) of the lemma. □

Lemma 2. *Let $s, t, u \in T(\mathcal{F}, \mathcal{X})$ such that $s =_{AC} u \longrightarrow_{R_{AC}} t$. Then, $s{\downarrow_{flat}} \longrightarrow_{R_{flat}} v =_P t{\downarrow_{flat}}$.*

Proof. By assumption, $u|_p =_{AC} l\sigma$ for some rule $l \to r$, position $p \in \mathcal{FP}os(t)$ and substitution σ. By lemma 1, $u{\downarrow_{flat}} \longrightarrow^{p'} v' =_P t{\downarrow_{flat}}$ for some variant $l' \to r'$ of $l \to r$, position p' and substitution σ' such that $t{\downarrow_{flat}}\ []_{p'}, \sigma$ is safe for l' and r'.

Since $s{\downarrow_{flat}} =_P u{\downarrow_{flat}}$, there exists a position $q \in \mathcal{P}os(s{\downarrow_{flat}})$ such that $s{\downarrow_{flat}}\ |_q =_P u{\downarrow_{flat}}\ |_{p'}$, and therefore $s{\downarrow_{flat}} \longrightarrow^{q}_{l'\to r'} v =_P v' =_P t{\downarrow_{flat}}$, and we are done. □

Lemma 3. *Let $s, t \in T(\mathcal{F}, \mathcal{X})$. Then $s \longrightarrow^{n}_{R_{AC}} t$ iff $s{\downarrow_{flat}} \longrightarrow^{n}_{R_{flat}} v =_P t{\downarrow_{flat}}$.*

Proof. By induction on the number n of rewrite steps from s to t. The case $n = 0$ is clear. Otherwise, let $s \longrightarrow^{*}_{R_{AC}} u \longrightarrow_{R_{AC}} t$. By the induction hypothesis, $s{\downarrow_{flat}} \longrightarrow^{*}_{R_{flat}} =_P u{\downarrow_{flat}}$. We conclude with Lemma 2. □

We now come to the main new result of this section:

Theorem 1. *Let R be a rewrite system over the signature $\mathcal{F}$ which is Church-Rosser modulo AC, and R_{flat} the associated rewrite system on flattened terms. Let now $s, t \in T(\mathcal{F}, \mathcal{X})$. Then $s \leftrightarrow^*_{R \cup AC}$ iff $s{\downarrow_{flat}} \longrightarrow^*_{R_{flat}} u$, $t{\downarrow_{flat}} \longrightarrow^*_{R_{flat}} v$ and $u \leftrightarrow^*_P v$.*

Proof. The only if part is clear. The if part follows easily from the assumption that R is Church-Rosser modulo AC and Lemma 3. □

4 Modularity of AC-rewriting.

As a corollary, we get a simple proof of modularity of AC-rewriting:

Theorem 2. *The Church-Rosser property of AC-rewriting is modular.*

Proof. Given a rewriting system R over a signature $\mathcal{F}$ containing associative-commutative function symbols, we consider the rewrite system $R_{flat} \cup P^{\rightarrow} \cup P^{\leftarrow}$ over the new signature $\mathcal{F}_{flat}$, where $P^{\rightarrow}$ denotes the rewrite system obtained by orienting the rules in P from left to right, and by $P^{\leftarrow}$ for the converse. By Theorem 1, R is Church-Rosser modulo AC iff R_{flat} is confluent modulo P, which is itself the case iff $R_{flat} \cup P^{\rightarrow} \cup P^{\leftarrow}$ is confluent.

Let now $R1$ and $R2$ be two such Church-Rosser rewriting systems, $P1$ and $P2$ be the respective permutative equations, and R and P be their respective unions. By Toyama's theoremcitetoyama87, $R_{flat} \cup P$ is confluent iff this is the case of $R1_{flat} \cup P1^{\rightarrow} \cup P1^{\leftarrow}$ and $R2_{flat} \cup P2^{\rightarrow} \cup P1^{\leftarrow}$. The result follows. □

5 Conclusion

Our treatment of AC-rewriting does not use AC-pattern matching. However, the extensions and specializations must be precomputed. And indeed, computing them *by need* is the same as using AC-pattern matching with a rule of the original system, making it extremely similar the approach advocated by Steven Ecker in [2].

As a result, AC-unification and AC-matching do not seem necessary for implementations. It would be interesting to put this method into pratice and compare it with traditionnal implementations. **Acknowledgment: we thank Yoshito Toyama for suggesting this short proof of the modularity result for AC-rewriting.**

References

1. **Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.**
2. **Stephen Ecker Efficient rewriting modulo AC.**
3. **Jieh Hsiang and Michaël Rusinowitch. On Word Problems in Equational Theories. In Proc. ICALP, 1987.**
4. **Jean-Pierre Jouannaud and Hélène Kirchner. Completion of a Set of Rules Modulo a Set of Equations. In *Siam Journal of Computing 15:4(1155–1194), 1984.***
5. ***Claude Kirchner, Hélène Kirchner and José Meseguer. Operational Semantics of OBJ-3. In* Proc. ICALP, *1988.***
6. ***Dallas Lankford.and A. M. Ballantyne. Decision procedures for simple equational theories with commutative-associative axioms: Complete sets of commutative-associative reductions. In "Memo ATP-39", Department of Mathematics and Computer Science, University of Texas, Austin, Texas, 1977.***
7. ***Claude Marché. On ground AC-completion. Proceedings RTA, 1991.***
8. ***Gerald E. Peterson and Mark E. Stickel. Complete sets of reductions for some equational theories. In* JACM *28(2):233–264, 1981.***
9. ***Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems.* Journal of the ACM, *34(1):128–143, April 1987.***